Aligning Sequences—Sequentially and Concurrently

Angela B. Shiflet[1], George W. Shiflet[1], Daniel S. Couch[1],
Pietro Hiram Guzzi[2], Mario Cannataro[2]
[1]*Wofford College, Spartanburg, SC, USA*
[2]*University "Magna Græcia" of Catanzaro, Catanzaro, Italy*
shifletab@wofford.edu, shifletgw@wofford.edu,
couchds@email.wofford.edu,
hguzzi@unicz.it, cannataro@unicz.it
© 2016

Introduction

Ruth returned from her vacation to find that her mother had been diagnosed with cancer in both breasts. Unfortunately, the cancer had also spread to her mother's lungs and liver, and aggressive chemotherapy had been begun. The genetic analysis from the breast biopsy revealed that her mother had a mutation for the BRCA1 gene. At first, her mother responded well to treatment, but she died after three months. Her mother was only 48 years old. At the funeral, Ruth learned from her uncle that her mother's older sister also had died from breast cancer very young. Ruth remembered that her mother never wanted to talk about her sister.

During those three months, Ruth spent much time caring for her mother, but after her mother's death, her husband and father encouraged her to seek out genetic testing. Ruth set up an appointment for the testing and also did some of her own research on this gene.

Ruth found that 8% of women in the general population will develop breast cancer by age 70 (ACS 2015)*, and about 5-10% of breast cancers come from inherited, mutated genes* (Schwartz et al 2007). *However, if she carries her mother's mutation, she has a 44-78% chance of developing breast cancer by age 70* (Antoniou et al 2003; Chen 2007). *Furthermore, she would also have an increased risk of ovarian cancer.*

The U.S. National Library of Medicine (National Institutes of Health (NIH)) lists 26 different genes associated with breast cancer (U.S. National Library of Medicine 2015). Of all the breast cancers attributable to **gene mutations**, less than 20% are a result of a mutated *BRCA1* gene (Turnbull and Rahman 2008). Nevertheless, scientists have been very interested in *BRCA1* and the role this gene plays in the development of cancer. *BRCA1* has been localized on the long arm (q arm) of the 17th **chromosome**, position 21 (from base pair 41,196,311- 41,277,499 (81,189 bp)) (U.S. National Library of Medicine 2015). This position is usually described as 17q21 (chromosome 17, long arm, region 2, band 1). Miki et al (1994) reported results for a **cDNA** (DNA synthesized from the messenger RNA for this gene) sequence for this chromosomal region that corresponded to *BRCA1*. From this sequence, they could predict the **amino acid** sequence for the **protein** coded for by the **gene**.

Research has determined that the *BRCA1* gene produces mRNA that can be spliced in several different ways (producing different proteins), implying that it has a complex set of functions. One function seems to be acting as a **tumor suppressor gene**, which inhibits uncontrolled cellular growth. Some *BRCA1* proteins have important roles in the repair of **damaged DNA**. Damaged DNA may produce abnormal protein products, which may not perform their intended function. *BRCA1* products also interact with a bevy of other proteins to regulate other genes, which are involved in such functions as the

control of cell cycle, cell division, embryonic development, signal transduction, etc. (BRCA1 2015, NCBI 2015, UniProt 2015).

To understand how the changes in the nucleotide sequence can lead to cancer, we should step back and review some basic cell and molecular biology.

The human body is made up of about 10 trillion cells, most of which have become specialized for a particular function (e.g., neurons, liver cells, skeletal muscle, etc.). Generally, each cell is composed of an internal, aqueous **cytosol**, surrounded by a lipid/protein barrier called the **plasma membrane**. Within the cytosol are the nucleus and other membrane-bound **organelles**, each with a set of specific functions. The **nucleus** is of particular interest, because it comprises the **nucleic acid**, DNA (deoxyribonucleic acid). DNA, of course, represents the genetic instructions for the cell's physiology, growth, and replication, organized as a sequence of **nucleotides**. DNA's sequence of nucleotides determines the sequence of complementary strands of DNA as well as the sequence of various types of RNA's (ribonucleic acids). One type of RNA (messenger, mRNA) can transduce the nucleotide sequences from DNA into the amino acid sequences of a **protein's**.

Nucleic Acids

DNA and RNA are polymers, or long chains, of molecules called **nucleotides**. Each nucleotide is made up of a sugar (either **deoxyribose** for DNA or **ribose** for RNA), a phosphate, and a nitrogen base (**adenine** (**A**), **guanine** (**G**), **cytosine** (**C**), and **thymine** (**T**) in DNA or **uracil (U)** in RNA). A and G are **purines**, while C, T, and U are **pyrimidines** (Figure 1). DNA is composed of two strands of nucleotides, bound together by hydrogen bonds, whereas RNA is composed of a single nucleotide strand. Phosphodiester linkages connect each nucleotide to its neighbor (Figure 2). A phosphate is attached to the #5 carbon of the sugar and is designated the **5' carbon**. The phosphate then forms a bond with the hydroxyl group (-OH) attached to the #3 carbon of its neighbor, designated the **3' carbon**. Each nucleotide is linked to its neighbors by 5' and 3' ends, and that arrangement yields a nucleotide chain with specific 5'-3' orientation.

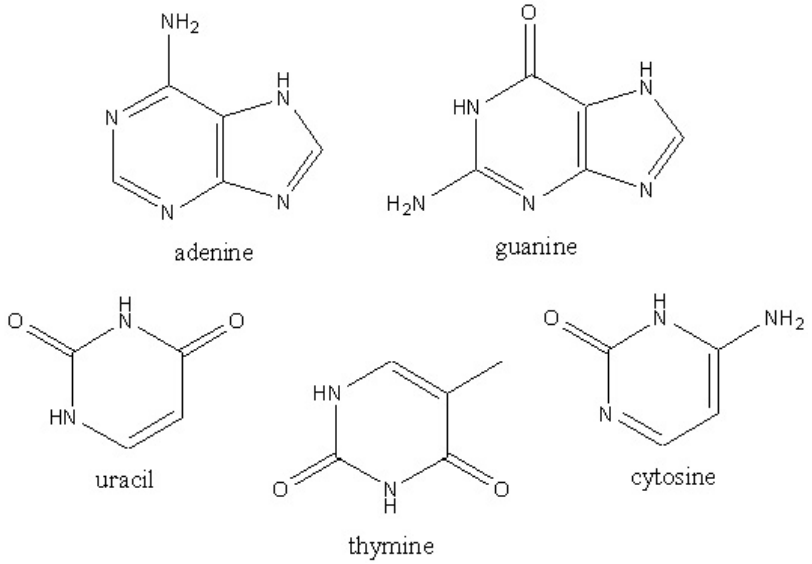

Figure 1 Nitrogen bases (Mrbean427 2008)

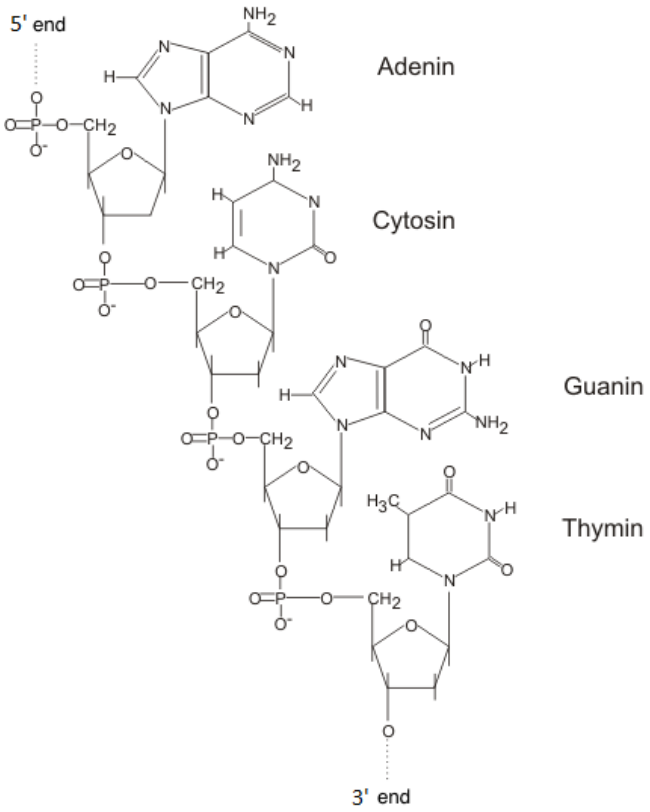

Figure 2 Single strand nucleotide chain (Dna_strand3.png: Boumphreyfr 2011 with slight revision)

In the cell, the nucleic acid **DNA** (**deoxyribonucleic acid**) contains the encoded information for the manufacture of all the proteins a cell needs. However, DNA does not oversee protein synthesis directly, but acts through an intermediary nucleic acid, **mRNA** (**messenger ribonucleic acid**). The mRNA sequences produced from the DNA template subsequently specify the amino acid order of proteins.

In DNA, bases in one strand form hydrogen bonds with the bases in the second strand. Given their particular structures, A in one strand and T in the other strand will bond together, whereas C and G bond together. Each of these pairs of **complementary bases** and is referred to as a **base pair** (**bp**). Barring some error, the reliability of base pairing, allows us to deduce the sequence of bases in the one strand, if we know the sequence of the other strand. For example, suppose one sequence is *s* = ATGAC. Because of the required pairing, A - T and C - G, we know the base pairs must appear as follows:

s:	A	T	G	A	C
	|	|	|	|	|
	T	**A**	**C**	**T**	**G**

RNA is normally a single strand of nucleotides made up of ribose sugars and bases A, C, G, and U (instead of the nitrogen base thymine (T)) (Table 1). Various types of cellular RNA play different roles in the cell. An RNA strand may form loops through base pairing with complementary regions within that one molecule. Also, strands of RNA may form hydrogen bonds with other RNA molecules or with single stranded portions of DNA molecules.

Base	Abbreviation	Complement	DNA	In RNA	Base Type
adenine	A	T in DNA, U in RNA	+	+	purine
guanine	G	C	+	+	purine
cytosine	C	G	+	+	pyrimidine
thymine	T	A	+	-	pyrimidine
uracil	U	A	-	+	pyrimidine

Table 1 Bases in DNA and RNA

Quick Review Question 1 Give the complement of the sequence GTACCT.

Quick Review Question 2 Give the term associated with each of the following:

a. Contains the encoded information that is stored to direct the manufacture of all the proteins a cell needs
b. Compound molecule made of a sugar, a phosphate, and a nitrogen base
c. Type of molecule in DNA and RNA sequences
d. Bases in DNA
e. Bases in RNA
f. Purines
g. Pyrimidines
h. An intermediary nucleic acid in protein synthesis
i. Always bonds with base A in DNA
j. Always bonds with base A in RNA
k. Always bonds with base C in DNA or RNA
l. Always bonds with base T in DNA
m. Always bonds with base U in RNA
n. Always bonds with base G in DNA or RNA
o. Single strand of nucleotides
p. Double strand of nucleotides
q. Pair of complementary bases

Proteins

Because of the numerous roles **proteins** play in cells, they are essential to life. Proteins are components of every cellular membrane, where they have important functions. For instance, they may act as transporters, moving ions or other molecules into or out of the cell or organelle. In the nucleus, some proteins act as **transcription factors**, activating or inhibiting the activity of targeted genes. Proteins may form the **cytoskeleton** of cells

and participate in internal transportation of structures such as vesicles or chromosomes. **Enzymes**, catalysts for chemical reactions in the cell, are mostly proteins.

A simple protein, or **polypeptide**, is a linear polymer or chain of **amino acids**. Table 2 catalogues the twenty amino acids regularly found in proteins, including both the one-letter and the three-letter codes for each. The central carbon (α carbon) of each amino acid bonds with 4 chemical groups—an **amino group** ($-NH_3^+$), a **carboxyl group** ($-COO^-$), a hydrogen (H), and a variable side-chain (**R-group**) (Figure 3). The R-group is significant, because it determines the chemical nature (acidic, nonpolar, etc.) of each amino acid along the chain. The constituent amino acids of a protein are linked by **peptide bonds**, which form through the interaction of an amino group of one amino acid with the carboxyl group of its neighbor to the left (Figure 4). As these bonds form, water is released, in a condensation reaction. Because one end (**N-terminal**) of a protein has a free amino group and the other (**C-terminal**) has a free carboxyl group, we can assign an orientation to the chain and list the amino acids from the "beginning" (N-terminal) of the chain to the "end" (C-terminal).

One-Letter Code	Three-Letter Code	Name
A	Ala	Alanine
R	Arg	Arginine
N	Asn	Asparagine
D	Asp	Aspartic Acid
C	Cys	Cysteine
Q	Gln	Glutamine
E	Glu	Glutamic Acid
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
L	Leu	Leucine
K	Lys	Lysine
M	Met	Methionine
F	Phe	Phenylalanine
P	Pro	Proline
S	Ser	Serine
T	Thr	Threonine
W	Trp	Tryptophan
Y	Tyr	Tyrosine
V	Val	Valine

Table 2 The twenty commonly occurring amino acids along with their one-letter and three-letter codes. (Note: B is used when one cannot distinguish between D and N because of amino acid analytical processing. Similarly, Z is used when it is ambiguous whether the amino acid is E or Q. X represents an unknown or nonstandard amino acid.)

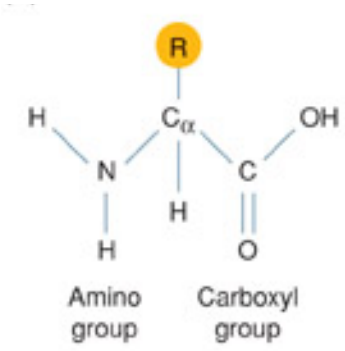

Figure 3 Structure of an amino acid (LLNL 2004)

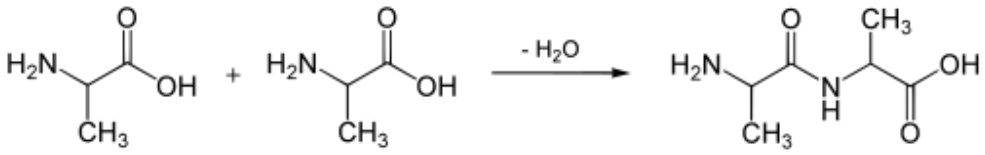

Figure 4 Formation of peptide bond (Yikrazuul 2008)

Quick Review Question 3 Give the term associated with each of the following:

- **a.** Basic building blocks of life
- **b.** Proteins that are catalysts for chemical reactions
- **c.** A simple protein is a linear chain of these
- **d.** Link chains of amino acids
- **e.** Formed through the interaction of an amino group of one amino acid with the carboxyl group of another
- **f.** Amino acid component of a protein
- **g.** Free amino group that is the beginning of the chain of amino acids
- **h.** Free carboxyl group that is the end of the chain of amino acids
- **i.** Linear polymer of amino acids

Connecting DNA Code to Protein Sequence

Almost every cell in the human body contains a set of very long DNA molecules, complexed with proteins, called **chromosomes**. The complete set of chromosomes in a cell, containing the organism's hereditary information, is called the **genome**. For example, a human genome is found in 23 pairs chromosomes (46 total). Each pair of chromosomes is made up of a chromosome from each parent. Stretches of each of these chromosomes that contain information for building a protein or a molecule of RNA are termed **genes**. Gene segments vary greatly in length, but the average gene is composed of about 28,000 base pairs (bp) (B10NUMB3R5 2015). There are contiguous sections of each chromosome that are not part of any gene. Some scientists have concluded that genes that code for proteins compose only a small percentage of the human genetic material. The function of these non-coding portions of the DNA is still being investigated. Some are known to be important for regulating gene expression, and others

may be necessary to match homologues and for structure in the nucleus. There is still much remaining to discover.

In a gene, a sequence of three nucleotides (**triplet**) specifies an amino acid. For example, the sequence TAT or TAC encodes the information for the amino acid tyrosine (Tyr or Y). The **genetic code** represents a correspondence between these triplets and the amino acids they specify. With four base choices, a pair of bases could only encode information for (4)(4) = 16 amino acids. With three bases, (4)(4)(4) = 64 possible triplets exist. Several, such as ACG, ACT, ACC and ACA, encode the same amino acid (threonine), whereas three sequences do not encode any amino acid. These three sequences (TAA, TAG, and TGA) signal a stop.

Synthesis of proteins commences in the nucleus, where enzymes catalyze the production of a molecule of RNA, termed **messenger RNA** or **mRNA**. Each DNA triplet specifies a complementary sequence of three nucleotides, which we call a **codon**, in the RNA. RNA synthesis, called **transcription**, uses one of the strands of DNA as a template, where each nucleotide of the DNA strand base pairs to make the mRNA. The result is a strand of RNA that is complementary to the gene sequence, with U replacing T. The transcribed mRNA is transported into the cytoplasm, where it binds to a **ribosome**. Ribosomes, complexes of ribosomal RNAs (rRNA) and proteins, **translate** the mRNA sequence into a series of amino acids for the protein. The molecular apparatus of the ribosomes translates groups of three nucleotides (**codons**) into specific amino acids, thereby converting a particular sequence of nucleotides into a specific sequence of amino acids.

Quick Review Question 4 Give the number of each of the following:

- **a.** Bases in a triplet
- **b.** Different bases in DNA
- **c.** Commonly occurring amino acids
- **d.** Possible triplets

Quick Review Question 5 Give the term associated with each of the following:

- **a.** Very long DNA molecule
- **b.** Contiguous section of a chromosome that encodes information to build a protein or an RNA molecule
- **c.** A complete set of chromosomes in a cell
- **d.** Sequence of three nucleotides in a gene
- **e.** The process of using genetic code to direct the building of proteins
- **f.** The place in the cell where protein synthesis begins
- **g.** The place in the cell where enzymes catalyze the production of a molecule of RNA
- **h.** A molecule of RNA produced in the nucleus
- **i.** The synthesis of RNA
- **j.** Area surrounding the cell nucleus
- **k.** Small structure on which protein synthesis concludes
- **l.** The final process of protein synthesis
- **m.** Location in the cell of ribosomes
- **n.** Group of three nucleotides

Mutations and Cancer

As long as cells are able to operate normally, with sets of fully functional proteins and other interacting molecular components, growth, replication, and physiological processes proceed like clockwork. Cancerous cells, however, are abnormal in structure and function. What turns normal cells away from regular function to become cancerous? It is alterations to genes that can result in the development of cancer. These alterations are called **mutations**.

Some mutations are found in the egg and/or sperm, and therefore can be transmitted to offspring. Most mutations, however, are acquired during the lifetime of the individual, often the result of exposure to chemicals, radiation, tobacco smoke, and the like. It may take more than one mutation to make a cell cancerous, and in many cases the abnormal cell will die or will be destroyed by the immune system.

Mutations in the *BRCA1* gene have been studied by many scientists, and around 1800 have been described (Couch, Nathanson, and Offit 2014). Some result from intronic defects (introns are the regions of the transcribed mRNA, which are removed by splicing to leave protein coding regions (exons)). In some cases, there are small insertions or deletions of nucleotides within the gene. Missense mutations (where the wrong nucleotide is substituted for the correct one) are also important types of *BRCA1* mutations (Couch, Nathanson, and Offit 2014). Many of these variations are associated with breast and/or other types of cancer. Most mutations for this gene result in the production of an abnormal protein. As we discussed earlier, this gene has multiple functions, and defects in any one gene product may interfere or knock out a critical function. *BRCA1* proteins act to slow down cell division, to promote DNA repair, and to regulate transcription. Consequently, defects in these products can have devastating effects on the structure and function of the cell. Because *BRCA1* is classified as a **tumor suppressor gene**, it usually takes two defects, one in the gene on both number 17 chromosomes, before the effects are noticeable. So, if you have one normal *BRCA1* gene, it may make sufficient quantities of a normal product so that the mutation in the other copy is masked.

Other cancer-associated genes are termed **oncogenes**. These genes are found naturally within the genome or may be inserted by viruses. When they are functioning normally, they are called **proto-oncogenes**. Proto-oncogenes play important roles in regulating the cell cycle. When they are over stimulated by changes in their regulatory mechanisms, they can promote uncontrolled growth of cells.

Quick Review Question 6 Give the term associated with each of the following:

- **a.** Alteration to gene
- **b.** Gene that inhibits tumor growth
- **c.** Gene that plays an important role in regulating the cell cycle
- **d.** Cancer-associated gene found naturally in genome or inserted by virus and that has been over stimulated

Genomics and Bioinformatics

We can take what we have learned from genetics and molecular biology and employ it for genetic mapping and DNA sequencing. As techniques for sequencing improved, the

Human Genome Project was able to utilize the methodologies to sequence and map the entire human genome. The genome was made up of about three billion nucleotides, so the HGP had a daunting job. Although the initial estimate was 100,000 protein coding genes, there turned out to be less than 20,000. In fact, the protein coding portions of the genome makes up less than 2% of the total DNA. There are far more than 20,000 proteins in human cells, but with such processes as DNA rearrangement and alternate splicing of RNA, this lower number of genes is sufficient to generate all of the needed protein products (Ezkurdia et al 2014).

For many years, genes could be mapped to specific regions of a chromosome by determining recombination frequencies. During meiosis (cell division in developing gametes that results in a halving of the chromosome number in each egg or sperm cell), recombination occurs frequently between pairs of chromosomes (called homologues). For instance, it may occur between chromosome 17 from the mother and chromosome 17 from the father. During meiosis, homologous pairs line up, and it is during this time that portions of DNA from one homologue can be exchanged with similar DNA of the other homologue. **Recombination** or **crossing-over** is guided by enzymes, and this process is an important source of new genetic combinations that lead to increased variation in the gametes (Figure 5).

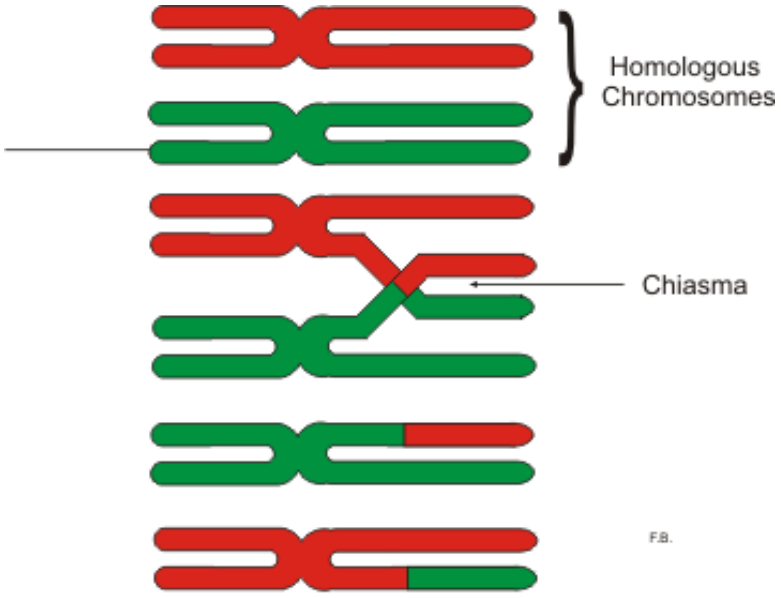

Figure 5 Recombination or crossing-over (Boumphreyfr 2009)

If you have already mapped a particular gene, you can use it as a marker for locating other genes close to it on the chromosome. If two genes are very close together, fewer crossing-over events will occur between the two genes than if they were further apart.

With advances in methodologies, smaller, more quickly detectable markers are used for mapping. One type of marker is the SNP (single nucleotide polymorphism), which is the change in only one nucleotide. These polymorphisms are numerous in the human genome and often occur in the noncoding regions of the DNA. What we have from

mapping with these markers is a **genetic or linkage map**. Such a map shows us the linear order of genes on a chromosome, expressed as recombination percentages (map units or centimorgans). One **map unit** equals one centimorgan (cM), which corresponds to 1% recombination between loci or gene positions on the chromosome. Biologists estimate that one map unit covers about 1 million base pairs. A genetic map allows us to link **phenotypes** (expression of genes; observable traits). However, this type of map reveals only an approximate set of relationships. The actual or physical distances between genes do not correspond directly to the distances determined by recombination frequencies.

Consequently, it is necessary to determine physical maps, which may be of differing resolutions. Scientist can determine the physical distances between distinct **landmarks**. There is no need to know the exact DNA sequence or what genes contained in the segment between the landmarks. Early on, commonly used landmarks were **restriction sites**. A restriction site is a short sequence of nucleotides, recognized by enzymes, isolated from various microbes, where the enzyme will make a cut (break bonds) in the nucleic acid. When nucleic acids are treated with such enzymes (singly or in combinations), the digest will contain a collection of fragments. The fragments will form a pattern that can be used to generate a low-resolution, physical map. During the first fifteen years of the century, great strides were made in the sequencing of DNA, and now we have very high-resolution maps, with nucleotide sequences, for most of the human DNA.

Genomics is a young branch of science that incorporates the tools and techniques of molecular biology not only to sequence genomes, but also to unravel unanswered questions regarding the genomic structure, function, and evolution. The information, so gathered, is being stored in enormous databases, allowing the development of a myriad of applications to biology, human health, agricultural practices, etc. Genomics is interrelated with the interdisciplinary field of **bioinformatics**. Bioinformatics, which relies on mathematics, statistics, and computer science, provides tools to compile, organize, and analyze the overwhelming volumes of data that are being generated from genomic studies.

Quick Review Question 7 Give the term associated with each of the following:

- **a.** Portions of DNA from one homologue can be exchanged with similar DNA of the other homologue
- **b.** Map that shows the linear order of genes on a chromosome, expressed as recombination percentages
- **c.** Measure that corresponds to 1% recombination between loci or gene positions on the chromosome
- **d.** A short sequence of nucleotides, recognized by enzymes, isolated from various microbes, where the enzyme will make a cut
- **e.** A young branch of science that incorporates the tools and techniques of molecular biology not only to sequence genomes, but also to unravel unanswered questions regarding the genomic structure, function, and evolution
- **f.** Interdisciplinary field that relies on mathematics, statistics, and computer science and that provides tools to compile, organize, and analyze the overwhelming volumes of data that is being generated from genomic studies

Ruth's Case

Because of the genomic research into breast cancer genes, scientists can determine the specific mutation of the BRCA1 gene that was associated with Ruth's mother's cancer. As we learn more about the way in which this mutation and others lead to cancer's development, we may be able to develop medical treatments that will prevent suffering and death from this disease. If Ruth's test for the mutated gene is positive, we still do not know for certain that she will develop breast cancer. However, armed with the knowledge we have, she can make informed decisions for her own health and lifestyle. Perhaps we'll have some even better options for her children's generation.

Alignment

Using bioinformatics, we can align DNA sequences to identify regions that are similar. Such a similarity might indicate that the two regions have the same function or evolve from a common ancestor in a sequence of mutations. In comparing two sequences, such as ATGAC and ACGC, we can employ a metric, called a **similarity score**, or **score**, to rate various alignments. For a scoring scheme, the highest possible similarity score indicates the best alignment(s). An **alignment** of two DNA sequences has spaces in the sequences so that they are of the same length but so that a space in one sequence is not in the same position as a space in the other sequence. For example, with a dash (-) indicating a space, one alignment of *s* = ATGAC and *t* = ACGC is

s:	A	T	G	A	C
t:	A	-	C	G	C

Instead of stacking the alignments, we can write them as a pair, (ATGAC, A-CGC). Another alignment is (ATGAC--, ---ACGC) or

s:	A	T	G	A	C	-	-
t:	-	-	-	A	C	G	C

Although we can rate the quality of an alignment in many ways, let us define the **score** for an alignment as the total of column scores, where the column scores have the following values:

- +1 for a match
- -1 for a mismatch
- -2 for a space in one of the corresponding positions

For example, with indicated column scores, the alignment

s:	A	T	G	A	C
t:	A	-	C	G	C
column scores:	1	-2	-1	-1	1

has a score of 1 + (-2) + (-1) + (-1) + 1 = -2, while the alignment

s:	A	T	G	A	C	-	-
t:	-	-	-	A	C	G	C
column scores:	-2	-2	-2	1	1	-2	-2

has a score of (-2) + (-2) + (-2) + 1 + 1 + (-2) + (-2) = -8. The **similarity** of two sequences is the maximum of all possible alignment scores.

Because we obtain the same column scores regardless of which sequence is in the first row, the order in which we write the sequences is irrelevant for determining an alignment score. Thus, the similarity of two sequences does not depend on which we write first. For example, the similarity of ATGAC and ACGC is the same as the similarity of ACGC and ATGAC.

Quick Review Question 8 Give the score for the alignment (ATGAC, ACG-C).

Similarity Score

The **Needleman-Wunsch Algorithm** is a technique to determine the similarity and the alignment(s) that yield this highest score (Needleman and Wunsch 1970). The algorithm employs **dynamic programming**, which divides a problem into a collection of smaller problems and uses the solutions to these smaller problems to solve the larger problem. The Needleman-Wunsch Algorithm makes the best decision for **prefixes**, or subsequences from the start of the sequences (the smaller problems), as it iterates over the length of those prefixes. We use the notation $s[i..j]$ to indicate the subsequence from position i to position j, where the first position number is 0 as with arrays in C, C++, Java, and Python. For example, with indexing starting at 0 in s = ATGAC, $s[1..2]$ is TG; and $s[0..2]$ is the prefix ATG. The notation $s[i]$ is base i in sequence s, so that $s[1]$ is T.

We write the developing intermediate similarity scores in a two-dimensional array, or matrix, a. As in Figure 6, the bases of s appear along the left margin of the array, starting at row 1; and the bases of t are on top, starting at column 1. Blanks, which we represent with hyphens, provide the headings for row 0 and column 0. For clarity in Figure 6, we have the indices of s on the extreme left and the indices of the rows of a adjacent to the matrix. Thus, 1 appears to the left of T, which is $s[1]$; and T heads row 2 of matrix a. Similarly, the indices of t appear above the corresponding bases; and the column numbers of a are immediately above the matrix.

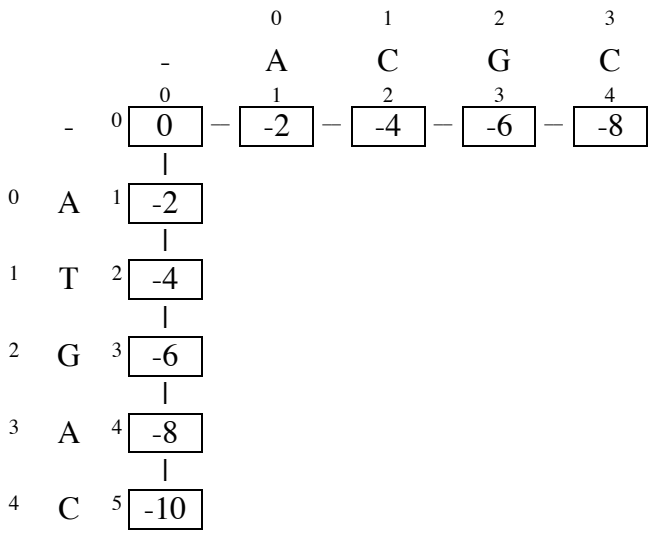

Figure 6 Initial values in similarity matrix with *s* = ATGAC (left) and *t* = ACGC (above)

Array position *a*[*i*][*j*] is the similarity score for prefixes *s*[0..*i* - 1] and *t*[0..*j* - 1]. For example, *a*[3][2] gives the similarity score for *s*[0..2] (ATG in our example) and *t*[0..1] (AC here). In row 0, we write the on-going scores for matching all spaces with prefixes of *t*. For example, aligning a space with the prefix *t*[0..0] = A costs -2; two spaces corresponding to *t*[0..1] = AT has a cumulative value of (-2) + (-2) = -4; three spaces and ACG, adds -2 to the previous score to obtain (-4) + (-2) = -6; and four spaces with *t* yields (-6) + (-2) = -8. Symmetrically, in column 0, we place the on-going scores for matching prefixes of *s* with all spaces. We draw line segments between subsequent values to indicate the derivation steps.

We give a general formula for score *a*[*i*][*j*] below, but first we illustrate the process with a particular example. To determine *a*[3][2], we only must know the row 3 heading (G), column 2 heading (C), and the scores in positions above (*a*[2][2]), to the left (*a*[3][1]), and diagonally left above (*a*[2][1]) the desired position. Figure 7 illustrates this situation. We obtain the best alignment of ATG and AC in at least one of three ways:

- **Diagonal:** Starting with the best alignment of AT and A, prefixes of *s* and *t*, respectively, consider the impact of the match (+1 value), or in this case, the mismatch (-1 value) of G onto the prefix of *s* and C onto the prefix of *t*. Thus, add +1 or -1 to the diagonal value.

- **Above:** Starting with the best alignment of AT and AC, prefixes of *s* and *t*, respectively, consider the impact (-2 value) of adding G to the prefix of *s* and blank to the prefix of *t*. Thus, add -2 to the value above the position.

- **Left:** Starting with the best alignment of ATG and A, prefixes of *s* and *t*, respectively, consider the impact (-2 value) of adding blank to the prefix of *s* and C to the prefix of *t*. Thus, also add -2 to the value to the left of the position.

The value in *a*[3][2] is the maximum of the three scores.

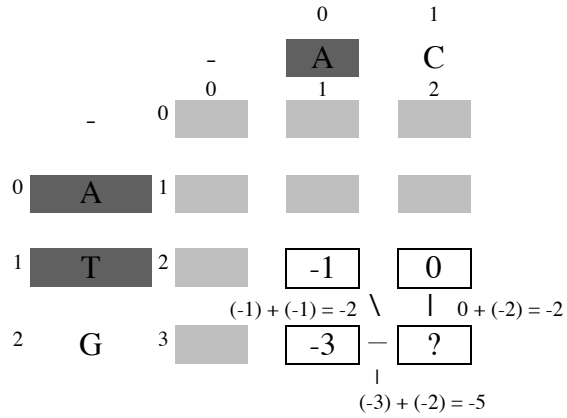

Figure 7 Determine *a*[3][2] from *a*[2][2], *a*[3][1], and *a*[2][1]

The score -1 in *a*[2][1] indicates the best score for aligning prefix sequences *s*[0..1] = AT and *t*[0..0] = A. Going diagonally to *a*[3][2], we have the score for attaching G to the end of AT and C to the end of A. The alignment situation is as follows:

Best alignment of AT and A	G
	C

Because G and C mismatch, this alignment of AT**G** and A**C** has a value of (-1) + (-1) = -2, which is the sum of the previous score and the mismatch value, as Figure 8 illustrates.

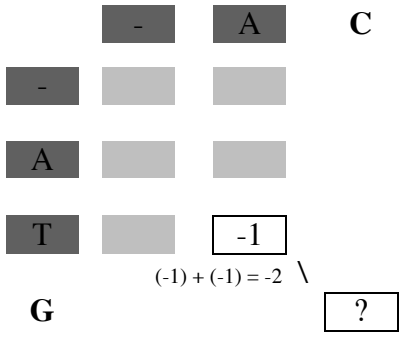

Figure 8 Intermediate score from mismatch of G and C

From Figure 7, *a*[3][2], *a*[2][2] = 0 is the similarity for prefixes sequences *s*[0..1] = AT and *t*[0..1] = AC. Thus, when extending the prefix of *s* to *s*[0..2] = ATG, we match a space with G from *s*, as follows:

Best alignment of AT and AC	G
	-

Because matching with a space costs -2, this arrangement has a total value of 0 + (-2) = -2, as Figure 9 shows.

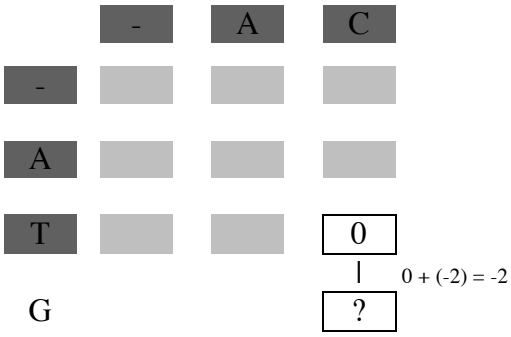

Figure 9 Intermediate score from alignment of of G and a space

Finally, we consider *a*[3][1] = -3, which is to the left of *a*[3][2]. To go from the best alignment of the prefix ATG of *s* and A of *t*, we match a space in *s* to C in *t*, as follows:

Best alignment of ATG and A	-
	C

The resulting value, *a*[3][1] + (-2) = (-3) + (-2) = -5, is not as good as the -2 values obtained from above and on the diagonal. The array element *a*[3][2] is the maximum of -2, -2, and -5, which is -2.

To summarize, *a*[3][2] is the maximum of the following values:

- Diagonal value (*a*[2][1]) plus -1 for mismatch of G and C (or diagonal value plus +1 if the letters had matched)
- Above value (*a*[2][2]) plus (-2)
- Left value (*a*[3][1]) plus (-2)

The general formula is as follows:

$$a[i][j] = \max \begin{cases} a[i-1][j-1] + p(s[i], t[j]) \\ a[i-1][j] + (-2) \\ a[i][j-1] + (-2) \end{cases} \quad \text{where}$$

$$p(s[i], t[j]) = \begin{cases} +1, & \text{if } s[i] = t[j] \\ -1, & \text{if } s[i] \neq t[j] \end{cases}$$

Figure 10 illustrates derivation of the value $a[i][j]$.

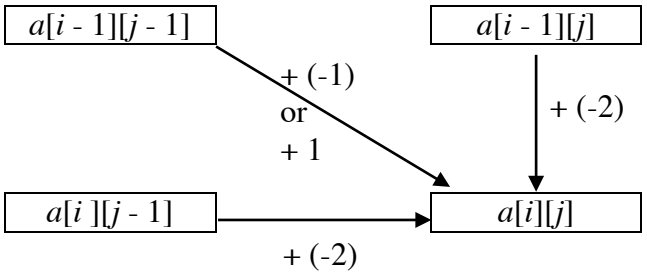

Figure 10 Derivation of the value $a[i][j]$

A line segment is drawn to $a[3][2]$ from any of the three positions that yields this maximum. In this example, we draw a line segment between $a[2][1]$ and $a[3][2]$ and between $a[2][2]$ and $a[3][2]$.

Quick Review Question 9 Suppose $a[2][2] = a[2][3] = 0$, $a[3][2] = -2$, and row 3 and column 3 headings read "G," as in the following matrix:

		0	1	2	G 3
	0				
	1				
	2			0	0
G	3			-2	

For Parts a-c, determine the score from each position to $a[3][3]$.

- **a.** Diagonal, $a[2][2]$
- **b.** Above, $a[2][3]$
- **c.** Left, $a[3][2]$
- **d.** The score of $a[3][3]$ is derived from which direction?

Quick Review Question 10 Suppose the value above is -5; the left position is 1; and the diagonal position is 0. The row label is G, and the column label is C, as in the following partial matrix:

	C	
	0	-5
G	1	

Compute the possible values from the directions in Parts a-c below and the new value in the array of similarity values.

a. Above
b. Left
c. Diagonal
d. What is the matrix value?
e. We draw line segment(s) from which position(s)?

Similarity Matrix

To derive the entire array, we initialize the first row and column as Figure 6 indicates. We then proceed row by row, from left to right, determining scores by the above formula. Having obtained values to the left, above, and on the diagonal, we can use the formula to evaluate *a*[*i*][*j*]. The answer to Quick Review Question 4 develops the complete array.

Quick Review Question 11 Fill in the array of values for the similarity matrix below, indicating the direction(s) from which you derive each value. Employ the following scoring:

- +1 for a match
- -1 for a mismatch
- -2 for a space in one of the corresponding positions

		-	A	C	G	C
		0	1	2	3	4
-	0	0	-2	-4	-6	-8
A	1	-2				
T	2	-4				
G	3	-6				
A	4	-8				
C	5	-10				

Figure 11 contains the entire similarity matrix for Quick Review Question 4. The value in the bottom, right corner, 0, is the similarity of ATGAC and ACGC. Following the line segments from that corner backward to *a*[0][0], we see how best to align the sequences:

- A diagonal line segment indicates a match or mismatch of the pair of bases that head the row and column. For example, the diagonal between *a*[4][3] and *a*[5][4] indicates that it is best to attach the Cs onto the prefixes aligned with similarity value *a*[4][3]. The diagonal between *a*[1][1] and *a*[2][2] says to align T and C onto the prefixes aligned with similarity value *a*[1][1].

- A vertical line segment matches the row heading in *s* with a space. For example, the horizontal line segment between *a*[3][3] and *a*[4][3] indicates to attach *A* to the prefix of *s* and a space to the prefix of *t* in the developing alignment.

- A horizontal line segment matches a space with the column heading in *t*.

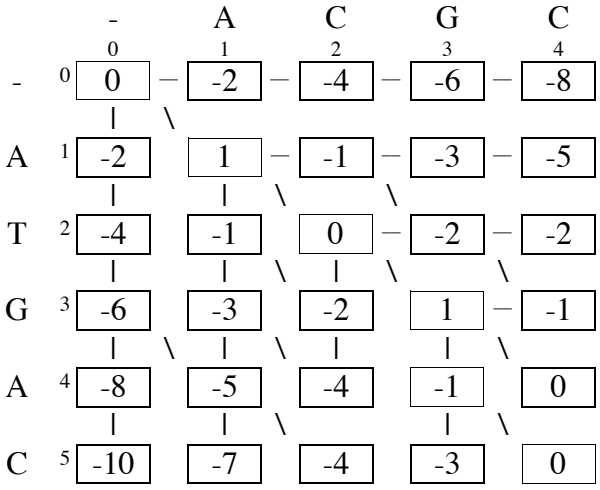

Figure 11 Array of similarity values for ATGAC and ACGC

Figure 12 displays the similarity matrix of Figure 11 with the path from *a*[5][4] to *a*[0][0] indicating the best alignment. Using Figure 12, Figure 13 gives the optimum alignment of ATGAC and ACGC. Sometimes, more than one alignment yields the same similarity.

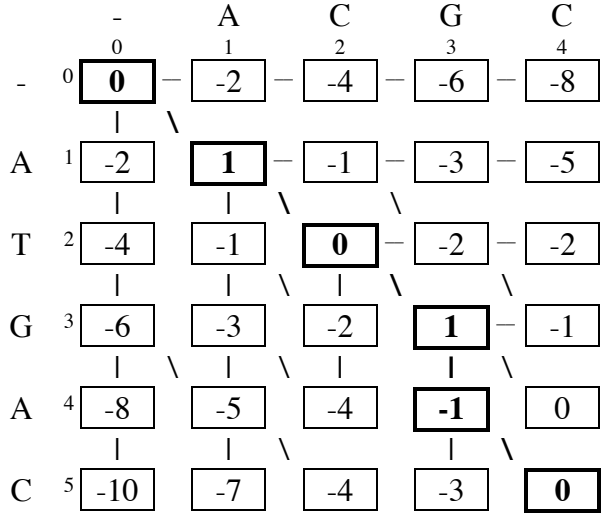

Figure 12 From Figure 11, path indicating best alignment of ATGAC and ACGC in bold

s:	A	T	G	A	C
t:	A	C	G	-	C

Figure 13 Optimal alignment of ATGAC and ACGC obtained from Figure 12

Needleman-Wunsch Algorithm

Bases match, bases mismatch, and a base's association with a space can carry different weights than +1, -1, and -2, respectively. Thus, the Needleman-Wunsch Algorithm (*score*), which determines the similarity array for two sequences, has input parameters for *matchValue*, *mismatchValue*, and *spacePenalty*, as well as for sequences *s* and *t*. The output parameter is the array (*a*) of similarity values. The Needleman-Wunsch Algorithm follows:

***score*(*s*, *t*, *matchValue*, *mismatchValue*, *spacePenalty*, *a*)**
Procedure to determine array of similarity values for two sequences of bases

Pre:
s and *t* are non-empty sequences of bases.
matchValue is the numeric value for two paired bases in an alignment agreeing.
mismatchValue is the numeric value for two paired bases in an alignment disagreeing.
spacePenalty is the numeric value for a base being paired with a space in an alignment.
matchValue > *mismatchValue*
matchValue > *spacePenalty*
a is an *m*-by-*n* array, where *m* is the length of *s* plus 1 and *n* is the length of *t* plus 1.

Post:
a is an array of similarity values for prefixes of *s* and *t*. The value in the bottom, right corner is the similarity of *s* and *t*.

Algorithm:
m ← length of *s*
n ← length of *t*
for *i* from 0 through *m*, do the following:
    a[*i*][0] ← *i* * *spacePenalty*
for *j* from 0 through *n*, do the following:
    a[0][*j*] ← *j* * *spacePenalty*
for *i* from 1 through *m*, do the following:
    for *j* from 1 through *n*, do the following:
        if (*s*[*i* - 1] equals *t*[*j* - 1])
            diagonalValue ← *matchValue*
        else
            diagonalValue ← *mismatchValue*
        a[*i*][*j*] ← max(*a*[*i* - 1][*j* - 1] + *diagonalValue*,

$a[i - 1][j] + spacePenalty,$
$a[i][j - 1] + spacePenalty)$

Quick Review Question 12 The questions below refer to the example in Quick Review Question 4 and Figure 11 as they relate to the *score* algorithm.

- **a.** Give the value of s.
- **b.** Give the value of t.
- **c.** Give the value of *matchValue*.
- **d.** Give the value of *mismatchValue*.
- **e.** Give the value of *spacePenalty*.
- **f.** Give the value of m.
- **g.** Give the value of n.
- **h.** Indicate the component to which the first loop assigns values: column 0, diagonal, last column, last row, or row 0
- **i.** Give the value that the second loop assigns to $a[0][3]$.
- **j.** Indicate how the third loop goes through the matrix: a column at a time, a diagonal at a time, or a row at a time
- **k.** When i equals 4, give the value of $s[i - 1]$.
- **l.** When j equals 5, give the value of $t[j - 1]$.
- **m.** When i equals 4 and j equals 5, give the value of *diagonalValue*.
- **n.** When i equals 4 and j equals 5, referring to Figure 11, give the value of $a[i - 1][j - 1]$ + *diagonalValue*.
- **o.** When i equals 4 and j equals 5, referring to Figure 11, give the value of $a[i - 1][j]$ + *spacePenalty*.
- **p.** When i equals 4 and j equals 5, referring to Figure 11, give the value of $a[i][j - 1]$+ *spacePenalty*.
- **q.** When i equals 4 and j equals 5, give the value of $a[i][j]$.

The algorithm below uses the array of similarity values to determine the optimal alignment of s and t in a method that is comparable to following line segments from the bottom, right to the top, left in the similarity array. The algorithm employs an initialization function, *align*, that calls a recursive function, *recAlign*, to develop the alignment. The function *align* has input parameters of this array and the original strings and output parameters of the aligned strings with a dash indicating a space. In *recAlign*, the plus sign (+) between a string and a character indicates concatenation, and *null* is the empty string.

***align*(*s*, *t*, *a*, *spacePenalty*, *sAlignment*, *tAlignment*)**

Algorithm to determine an optimal alignment of two sequences of bases

Pre:

s and t are sequences of bases.
a is the array of similarity values for prefixes of s and t, as completed by *score*.
spacePenalty is the numeric value for a base being paired with a space in an alignment.

Post:

sAlignment and *tAlignment* are sequences in the optimal alignment for sequences *s* and *t*, respectively, with a dash indicating a space.

Algorithm:

sLength ← length of *s*
tLength ← length of *t*
recAlign(*s*, *t*, *a*, *spacePenalty*, *sAlignment*, *tAlignment*, *sLength*, *tLength*)

***recAlign*(*s*, *t*, *a*, *spacePenalty*, *sAlignment*, *tAlignment*, *i*, *j*)**

Algorithm to determine optimal alignment of two prefix sequences of bases

Pre:

s and *t* are sequences of bases.
a is the array of similarity values for prefixes of *s* and *t*, as completed by *score*.
i is an index that is less than or equal to the length of *s*.
j is an index that is less than or equal to the length of *t*.
spacePenalty is the numeric value for a base being paired with a space in an alignment.

Post:

sAlignment and *tAlignment* are sequences in the optimal alignment for prefixes *s*[0..*i* - 1] and *t*[0..*j* - 1], respectively, with a dash indicating a space.

Algorithm:

```
if (i equals 0 and j equals 0)
    sAlignment ← null
    tAlignment ← null
else
    if (i > 0 and a[i][j ] equals a[i - 1][j] + spacePenalty)          // above
        recAlign(s, t, a, matchValue, mismatchValue, spacePenalty,
                sAlignment, tAlignment, i - 1, j)
        sAlignment ← sAlignment + s[i - 1]
        tAlignment ← tAlignment + '-'
    else if (j > 0 and a[i][j ] equals a[i][j - 1] + spacePenalty)     // left
        recAlign(s, t, a, matchValue, mismatchValue, spacePenalty,
                sAlignment, tAlignment, i, j - 1)
        sAlignment ←   sAlignment + '-'
        tAlignment ←   tAlignment + t[j - 1]
    else                                                               // diagonal
        recAlign(s, t, a, matchValue, mismatchValue, spacePenalty,
                sAlignment, tAlignment, i - 1, j - 1)
        sAlignment ← sAlignment + s[i - 1]
        tAlignment ← tAlignment + t[j - 1]
```

Quick Review Question 13 The questions below refer to the example in Quick Review Question 5 and Figure 12 as they relate to the *align* and *recAlign* algorithms.

- **a.** Give the value of *sLength*.
- **b.** Give the value of *tLength*.
- **c.** Give the value of *sAlignment* after completion of *align*.
- **d.** Give the value of *tAlignment* after completion of *align*.
- **e.** Referring to Figure 12, for *i* equals 4 and *j* equals 3, give the value of *a*[*i*][*j*].
- **f.** Referring to Figure 12, for *i* equals 4 and *j* equals 3, give the value of *a*[*i* - 1][*j*] + *spacePenalty*
- **g.** For *i* equals 4 and *j* equals 3, complete the arguments in the call to *recAlign*.

```
recAlign("ATGAC", "ACGC", a, -2, sAlignment, tAlignment, ___,   ____);
```

- **h.** After the call to *recAlign* in Part g when *i* equals 4 and *j* equals 3, the value of *sAlignment* is ATG. Give the value subsequently assigned to *sAlignment*.
- **i.** After the call to *recAlign* in Part g when *i* equals 4 and *j* equals 3, the value of *tAlignment* is ACG. Give the value subsequently assigned to *tAlignment*.
- **j.** For *i* and *j* equal 1, referring to Figure 12, give the value of *a*[*i*][*j*].
- **k.** For *i* and *j* equal 1, referring to Figure 12, give the value of *a*[*i* - 1][*j*] + *spacePenalty*.
- **l.** For *i* and *j* equal 1, referring to Figure 12, give the value of *a*[*i*][*j* - 1] + *spacePenalty*.
- **m.** For *i* and *j* equal 1, complete the arguments in the call to *recAlign*.

```
recAlign("ATGAC", "ACGC", a, -2, sAlignment, tAlignment, ___,   ____);
```

- **n.** Indicate the value of *sAlignment* after the call to *recAlign* in Part m: 0, "A", "AT", or *null*
- **o.** After the call to *recAlign* in Part m when *i* and *j* equal 1, the value of *sAlignment* is *null*. Give the value subsequently assigned to *sAlignment*.
- **p.** After the call to *recAlign* in Part m when *i* and *j* equal 1, the value of *tAlignment* is *null*. Give the value subsequently assigned to *tAlignment*.

The procedure *align* gives one optimal alignment, but not every one. When a choice exists, the algorithm gives priority to going vertically up, next to going to the left, and finally to going on the diagonal. An advantage of having the diagonal be the last option is that *recAlign* then does not need values for *matchValue* and *mismatchValue*. By altering the *if-else* statement, we can change this precedence. For some sequence pairs, such variations yield different optimal alignments.

Why Parallel?

Aligning two sequences requires a great deal of computing power. For instance, if each sequence is of length $n = 10{,}000$, then the effort for creation of the similarity matrix is on the order of, or proportional to, $n^2 = 100{,}000{,}000$. In general, we use big-oh notation, here $O(n^2)$, to indicate the effort in terms of the size, n. Thus, because $(2n)^2 = 4n^2$, going from a sequence of length n to a sequence double the length, $2n$, should take

proportionally four times as long. For a C implementation of the Needleman-Wunsch Algorithm, Table 3 lists timings of the sequential program for an increasing number of nucleotides. Note that the runtime to match sequences of length 10,000 is approximately four times the process for sequences of length 5,000. We can make a similar observation going from sequences of length 10,000 to those of length 20,000 and from 20,000 to 40,000. Figure 14 graphs the data in Table 3 with the superimposed plot of *runtime* = 1.62585×10^{-8} *nucleotides*^{2}. Thus, the graph illustrates that the runtime is proportional (constant of proportionality 1.62585×10^{-8}) to the square of the number of nucleotides. Moreover, when we are trying to match a sequence to multiple sequences in a database, the real challenge of the task becomes evident. Thus, it is advantageous to have multiple processes working on tasks in parallel, or simultaneously. We can have different processes aligning the search sequence to different database sequences, and/or we can have a parallel alignment algorithm to operate on each pair of sequences.

# Nucleotides	Runtime (s)
10	0.004
50	0.004
100	0.004
500	0.008
1,000	0.017
5,000	0.307
10,000	1.176
12,000	2.127
16,000	3.837
20,000	5.982
24,000	8.520
30,000	14.560
35,000	21.116
37,500	24.310
40,000	27.340

Table 3 Runtime (s) versus number of nucleotides for a sequential C implementation of the Needleman-Wunsch Algorithm

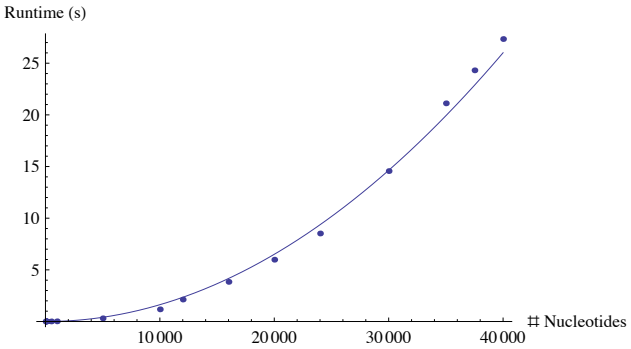

Figure 14 Graph of the data in Table 3 and *runtime* = 1.62585×10^{-8} *nucleotides*^{2}

However, splitting the work of determining the similarity matrix is not obvious for an innately sequential task—calculating the elements row by row from left to right. To determine one matrix element, we must know the values of three other elements: the value to the left, above, and diagonally above. In the following section, we consider the pipeline algorithm that splits the work so that, at least to some degree, the calculations can be done in parallel.

Pipeline Algorithm

For simplicity, let us assume that the number of processes equals the number of rows, n, in the similarity matrix and that Process j is responsible for making the calculations on row j. The algorithm *score* indicates the calculation of Process j's first element, or its element in column 0, in the similarity matrix, a, as follows:

$$a[0][j] \leftarrow j * spacePenalty$$

Instead of having one process looping sequentially through the calculation of each element in the first column (column 0), the processes can simultaneously compute their first column elements without having to communicate with the other processes. After the evaluation, Process j, for $j = 0, 1, \ldots, n - 2$, can send $a[0][j]$ to Process $(j + 1)$.

For an example, let us return to the sequences s = ATGAC and t = ACGC. With *spacePenalty* = -2, for the first step six processes calculate simultaneously the first-row elements, as the following indicates:

Step 1:

	-	A	C	G	C	
-	**0**					Process 0
A	**-2**					Process 1
T	**-4**					Process 2
G	**-6**					Process 3
A	**-8**					Process 4
C	**-10**					Process 5

The six values are evaluated in the time it would take the sequential algorithm to calculate one element. Afterwards, Processes 0 – 4 send their values (0, -2, -4, -6, -8, respectively) to Processes 1 – 5, respectively, for the calculations involving diagonal elements.

Moreover, as in the *score* function, Process 0 can compute ith element in row 0 by just knowing the value of *spacePenalty*, as follows:

$$a[i][0] \leftarrow i * spacePenalty$$

However, instead of looping through all of the m elements, immediately after the calculation of row 0's second element, $a[0][1]$, Process 0 can communicate this value to

Process 1. For example, with *spacePenalty* = -2, Process 0 calculates its next element (*a*[0][1] = 1 * -2 = -2) in boldface as follows and sends its this value to Process 1:

Step 2:

	-	A	C	G	C	
-	0	**-2**				Process 0

After this communication, two calculations can occur simultaneously: Process 0 can evaluate *a*[0][2] as 2 * *spacePenalty* = 2 * -2 = -4; and knowing the diagonal (*a*[0][0] = 0), left (*a*[1][0] = -2), and above (*a*[0][1] = -2) elements, Process 1 can evaluate *a*[1][1] as 1, the maximum of 0 + 1, -2 + -2, and -2 + -2. These simultaneous calculations are in boldface in the following step:

Step 3:

	-	A	C	G	C	
-	0	-2	**-4**			Process 0
A	-2	**1**				Process 1

Then, at the same time, Process 0 sends *a*[0][2] = -4 to Process 1, and Process 1 sends *a*[1][1] = 1 to Process 2. Now, Processes 0, 1, and 2 can be occupied simultaneously calculating their next elements. Process 0 calculates 3 * -2 = -6; while Process 1 uses -2, 1, and -4 to calculate *a*[1][2] = -1; and Process 2 employs -2, -4, and 1 in its evaluation of *a*[2][1] = -1, as follows:

Step 4:

	-	A	C	G	C	
-	0	-2	-4	**-6**		Process 0
A	-2	1	**-1**			Process 1
T	-4	**-1**				Process 2

Processes 0, 1, and 2 now pass these boldface values to Processes 1, 2, and 3, respectively. Then, the first four processes can simultaneously evaluate their next elements, as Step 5 below indicates in boldface. With each step, an additional process was drafted to work.

Step 5:

	-	A	C	G	C	
-	0	-2	-4	-6	**-8**	Process 0
A	-2	1	-1	**-3**		Process 1
T	-4	-1	**0**			Process 2
G	-6	**-3**				Process 3

After passing these boldface values to the next processes down, Process 0 is out of work. However, Processes 1 through 4 can calculate their next values (in boldface below), with Processes 1 through 3 sending the results to Processes 2 through 4, respectively:

Step 6:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	**-5**	Process 1
T	-4	-1	0	**-2**		Process 2
G	-6	-3	**-2**			Process 3
A	-8	**-5**				Process 4

With Processes 0 and 1 having nothing to do, Processes 2 – 5 evaluate their next elements, which are in boldface in the following, and Processes 2 – 4 communicate their results to the subsequent processes:

Step 7:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	**-2**	Process 2
G	-6	-3	-2	**1**		Process 3
A	-8	-5	**-4**			Process 4
C	-10	**-7**				Process 5

Now, only Processes 3, 4, and 5 have work to do, as the following indicates:

Step 8:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	-2	Process 2
G	-6	-3	-2	1	**-1**	Process 3
A	-8	-5	-4	**-1**		Process 4
C	-10	-7	**-4**			Process 5

After the appropriate communication, only Processes 4 and 5 have elements to calculate:

Step 9:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	-2	Process 2
G	-6	-3	-2	1	-1	Process 3
A	-8	-5	-4	-1	**0**	Process 4
C	-10	-7	-4	**-3**		Process 5

Process 5 has one last element, $a[5][4]$, to calculate for completion of the entire similarity matrix:

Step 10:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	-2	Process 2
G	-6	-3	-2	1	-1	Process 3
A	-8	-5	-4	-1	0	Process 4
C	-10	-7	-4	-3	**0**	Process 5

Looking back at this example, we only needed 10 steps in using the parallel pipeline algorithm to perform the calculations of this 6-by-5 matrix: 5 steps to march across, keeping Process 0 occupied with each step, and 5 steps to complete each of the rows below row 0. By contrast, the sequential Needleman-Wunsch Algorithm requires 6 * 5 = 30 steps to do the calculations, evaluating one element at a time.

Quick Review Question 14 Indicate the number of calculation steps using sequential Needleman-Wunsch Algorithm and a pipeline algorithm for each of the following sizes of matrices:

a. 10-by-6
b. 300-by-333
c. 5000-by-5000
d. 30,200-by-30,190

To reiterate, Figures 15 and 16 depict the progress of this pipelining system for sequences of length $n = 11$ and $m = 16$. The processes calculate the darker elements without information from any other process. Shading indicates the pipeline order of the calculations. After communication of the value above, a process can start computation of its element in darker outline. Thus, calculation of values on this **anti-diagonal** can proceed in parallel. Figure 15 shows one of the earlier phases with only three processes being busy, while Figure 16 depicts a stage where all processes are occupied (Chen et al 2006).

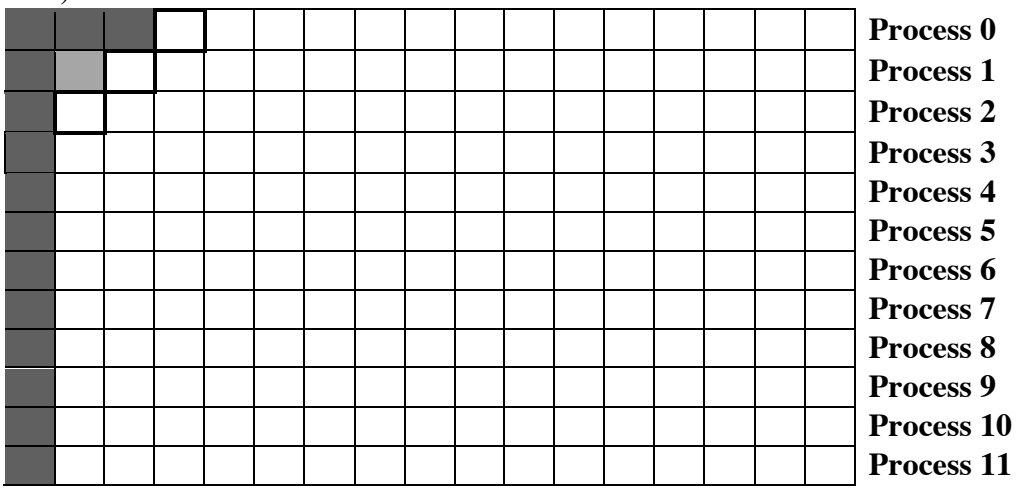

Figure 15 Pipelining the similarity matrix for sequences of length $n = 11$ and $m = 16$

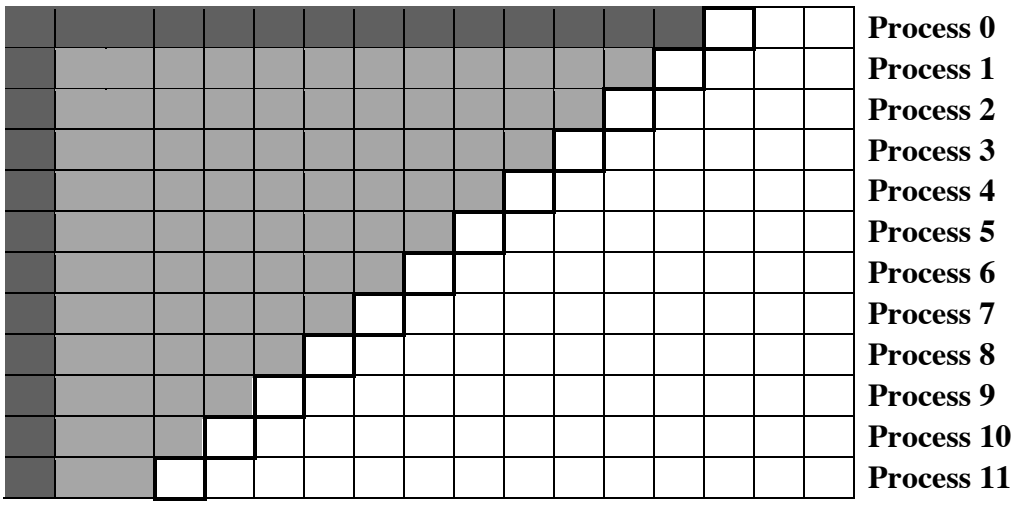

Figure 16 Pipelining the similarity matrix for sequences of length $n = 11$ and $m = 16$

Quick Review Question 15 Suppose that two sequences are each of length $n = m = 30$. Assume we are using the pipeline algorithm using the configuration above with the number of processes equaling the number of rows and each process calculating the value of one similarity matrix (a) element before communicating that value.

- **a.** How many processes should we use?
- **b.** Which process computes $a[0][20]$, and to which process does it send the value?
- **c.** Which process computes $a[20][0]$, and to which process does it send the value?
- **d.** Which process computes $a[30][10]$, and to which process does it send the value?
- **e.** What is the size of the similarity matrix?
- **f.** How many steps are required to calculate (fill in) the matrix sequentially?
- **g.** How many simultaneous steps are required to calculate (fill in) the matrix using the described pipeline method?
- **h.** What is the complexity of this version of the pipeline algorithm?
- **i.** How many simultaneous communications are required?

As indicated in the two previous quick review questions, using this version of the pipeline algorithm, the number of steps to calculate the similarity matrix for sequences of length n is on the order of, or proportional to, n, written with big-oh notation as $O(n)$. The processes determine the elements of the first column simultaneously, in approximately the same amount of time that it takes a single process to evaluate one element. Then, Process 0 marches across the first row of n elements with other processes performing evaluations, too. Afterwards, calculations in the remaining n rows are completed in n steps. Thus, the complexity for calculation using the pipeline algorithm, where a process communicates a value immediately after its evaluation, is as follows: $O(1) + O(n) + O(n) = O(n)$. This complexity is a great improvement over the sequential program, which has complexity $O(n^2)$.

However, one disadvantage of this version of the pipeline algorithm is the amount of communication, which is O(*n*), too. For smaller sequences, the pipeline version with 32 processes actually takes longer (Table 4). However, starting with sequences of length 12,000 nucleotides, we do observe speedup. For example, for sequences of length 37,500, the sequential algorithm takes approximately 24.310 s / 16.510 s = 1.472 times longer than the pipeline algorithm. Ideally, employing 32 processes, we would hope for a speedup of 32, but communication limits this speedup potential.

# Nucleotides	Sequential Runtime (s)	Pipeline Runtime (s), 32 Processes	Speedup
10	0.004	0.0162	0.247
50	0.004	0.0084	0.476
100	0.004	0.0067	0.593
500	0.008	0.0095	0.842
1,000	0.017	0.0157	1.083
5,000	0.307	0.3210	0.956
10,000	1.176	1.2310	0.955
12,000	2.127	1.7980	1.183
16,000	3.837	3.1730	1.209
20,000	5.982	4.7770	1.252
24,000	8.520	7.0690	1.205
30,000	14.560	11.034	1.320
35,000	21.116	14.925	1.415
37,500	24.310	16.510	1.472
40,000	27.340	19.240	1.421

Table 4 Runtime (s) versus number of nucleotides for a sequential and pipeline C implementations of the Needleman-Wunsch Algorithm along with speedup

Block-and-Band Version of the Pipeline Algorithm

To reduce communication, we can have each process calculate a **block** of several column values before communicating the results to the next process. For example, with sequences *s* = ATGAC and *t* = ACGC and *spacePenalty* = -2, we start with the same initial step of calculating and communicating the first column elements:

Step 1:

	-	A	C	G	C	
-	**0**					Process 0
A	**-2**					Process 1
T	**-4**					Process 2
G	**-6**					Process 3
A	**-8**					Process 4
C	**-10**					Process 5

Then, sequentially, Process 0 computes the two-element block *a*[0][1] = 1 * -2 and *a*[0][2] = 2 * -2 = -4:

Steps 2:

	-	A	C	G	C	
-	0	**-2**	**-4**			Process 0

After communicating the results to Process 1, Process 0 can work on evaluating sequentially its next block, at the same time that Process 1 is calculating sequentially its own block of two elements:

Step 3:

	-	A	C	G	C	
-	0	-2	-4	**-6**	**-8**	Process 0
A	-2	**1**	**-1**			Process 1

Processes 0 and 1 communicate these blocks to Processes 1 and 2, respectively. Computation continues with the latter processes evaluating their two-element blocks concurrently:

Step 4:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	**-3**	**-5**	Process 1
T	-4	**-1**	**0**			Process 2

After communication, Process 1 no longer participates as Processes 2 and 3 continue:

Step 5:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	**-2**	**-2**	Process 2
G	-6	**-3**	**-2**			Process 3

Afterwards, in the same fashion at the next step, two processes concurrently calculate their block elements in sequence and pass their blocks to the subsequent processes:

Step 6:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	-2	Process 2
G	-6	-3	-2	**1**	**-1**	Process 3
A	-8	**-5**	**-4**			Process 4

Then, Process 3's row is complete, and Processes 4 and 5 evaluate the following boldface blocks:

Step 7:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	-2	Process 2
G	-6	-3	-2	1	-1	Process 3
A	-8	-5	-4	**-1**	**0**	Process 4
C	-10	**-7**	**-4**			Process 5

After Process 5 sends its block to Process 5, the latter can complete the similarity matrix:

Step 8:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	Process 1
T	-4	-1	0	-2	-2	Process 2
G	-6	-3	-2	1	-1	Process 3
A	-8	-5	-4	-1	0	Process 4
C	-10	-7	-4	**-3**	**0**	Process 5

Although a process must calculate the elements of a block sequentially, multiple processes, here two, can evaluate their blocks concurrently. This example required seven communication steps instead of nine in the case of single-element blocks. Moreover, parallel processing commands to communicate a block of elements are faster than two separate sends of single elements.

Figure 17 depicts communication of blocks, which are in color, of size 2, for sequences of length $n = 11$ and $m = 16$ (Chen et al 2006).

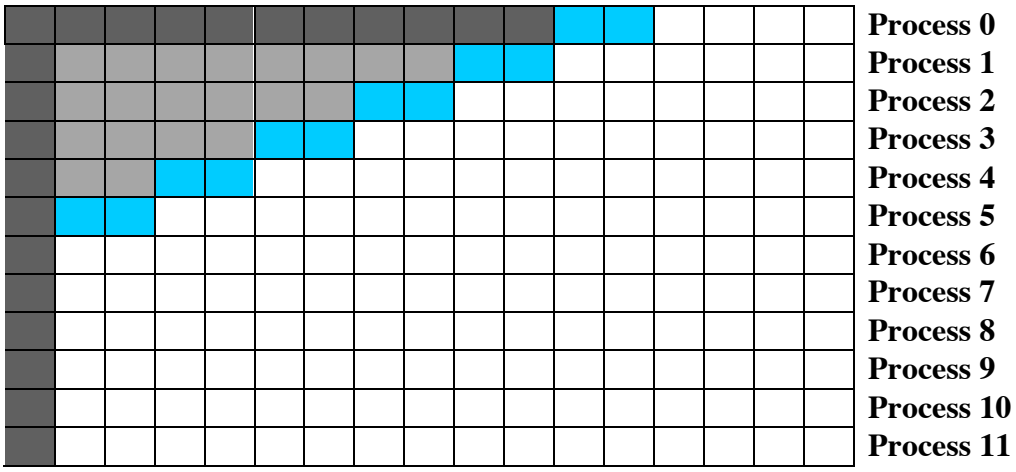

Figure 17 Pipelining a similarity matrix with block size of 2

Quick Review Question 16 For sequences of length $n = 17$ and $m = 15$, give the number of

- **a.** calculation steps using the sequential Needleman-Wunsch Algorithm
- **b.** calculation steps using the pipeline algorithm
- **c.** communication steps using the pipeline algorithm
- **d.** sequential computations within a block of size 3
- **e.** communication steps using a block of size 3
- **f.** sequential computations within a block of size 5
- **g.** communication steps using a block of size 5

Besides having a process calculate multiple column elements, we can make a process responsible for a **band**, or several rows, of elements. Figure 18 shows an allocation, where each process is responsible for a band size of 3 rows, where the block size, or number of columns, is 2. Thus, when possible, Process j calculates the values in a 3×2 submatrix and then communicates the two values in the last row of the submatrix to the next process, Process $(j + 1)$. Upon receiving the two elements, Process $(j + 1)$ proceeds sequentially row-by-row from left to right to calculate another 3×2 submatrix. Evaluation of the 3×2 submatrices proceeds in parallel (Chen et al 2006).

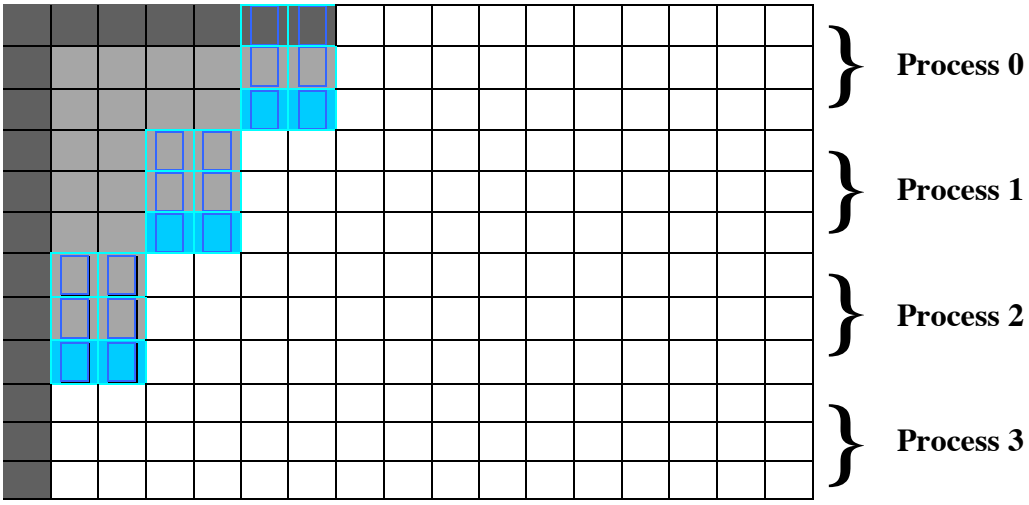

Figure 18 Pipelining a similarity matrix with a band size of 3 and a block size of 2

Returning to the sequences s = ATGAC and t = ACGC with *spacePenalty* = -2, suppose we employ a band and block, each of size 2. Because the similarity matrix is of size 6×5 and each band incorporates 2 rows, the algorithm employs $6 / 2 = 3$ processes. One form of the initial step, which is below, has each process calculating the two elements of its band in the first column (column 0). Then, Processes 0 and 1 send their bottom elements, -2 and -6, to Processes 1 and 2, respectively.

Step 1:

	-	A	C	G	C	
-	**0**					Process 0
A	**-2**					
T	**-4**					Process 1
G	**-6**					
A	**-8**					Process 2
C	**-10**					

Process 0 continues by sequentially calculating the four elements in the 2 × 2 boldface submatrix below. Then, the process sends the last row of the submatrix with elements 1 and -1 to Process 1.

Step 2:

	-	A	C	G	C	
-	0	**-2**	**-4**			Process 0
A	-2	**1**	**-1**			

With the appropriate information, Process 1 now has enough information to calculate the elements of another 2 × 2 submatrix sequentially at the same time that Process 0 evaluates its last submatrix (see boldface below). Then, Process 0 sends the elements of the last row of its submatrix, -3 and -5, to Process 1 while Process 1 communicates its last row with values -3 and -2 to Process 2.

Step 3:

	-	A	C	G	C	
-	0	-2	-4	**-6**	**-8**	Process 0
A	-2	1	-1	**-3**	**-5**	
T	-4	**-1**	**0**			Process 1
G	-6	**-3**	**-2**			

Process 0 is now idle, while Processes 1 and 2 evaluate their next submatrices, in boldface as follows:

Step 4:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	
T	-4	-1	0	**-2**	**-2**	Process 1
G	-6	-3	-2	**1**	**-1**	
A	-8	**-5**	**-4**			Process 2
C	-10	**-7**	**-4**			

Only Process 1 needs to communicate the last row with 1 and -1 to Process 2 so that the latter can evaluate the last submatrix, as follows in boldface:

Step 5:

	-	A	C	G	C	
-	0	-2	-4	-6	-8	Process 0
A	-2	1	-1	-3	-5	
T	-4	-1	0	-2	-2	Process 1
G	-6	-3	-2	1	-1	
A	-8	-5	-4	**-1**	**0**	Process 2
C	-10	-7	-4	**-3**	**0**	

Although sequential processing occurs, there is less communication with the block-and-band version of the pipeline algorithm. In the 2-by-2 version above, only four simultaneous sends of single elements or two-element sequences occur. In contrast, the version with 2-element blocks with one-element bands required seven communication steps, and the original pipeline version with single-element blocks had nine communication steps.

Quick Review Question 17 For sequences of length $n = 17$ and $m = 15$, give the number of

- **a.** processes for bands of size 6 and blocks of size 3
- **b.** sequential computations within such a submatrix
- **c.** elements communicated from such a submatrix
- **d.** communication steps for bands of size 6 and blocks of size 3
- **e.** processes for bands of size 3 and blocks of size 5
- **f.** sequential computations within such a submatrix
- **g.** elements communicated from such a submatrix
- **h.** communication steps for bands of size 3 and blocks of size 5

Quick Review Question 18 Suppose that two sequences are each of length $n = m = 30$. Assume we are using the pipeline algorithm with a block and band size of 6.

- **a.** How many processes are required?
- **b.** Which process computes $a[20][10]$, and to which process, if any, does it send the value?
- **c.** How many elements are in one of Process 2's submatrices?
- **d.** How many values at a time does Process 2 communicate from a submatrix?
- **e.** How many steps are required to calculate (fill in) the matrix sequentially?
- **f.** One communication of a contiguous sequence of 6 elements is faster than 6 individual sends. How many simultaneous contiguous sequence communications are required?

By running the parallel program for different sizes of blocks and bands and timing the results, we can determine an optimal balance of computation and communication. Of course, we should perform each experiment numerous times, say 100, and average the individual times to obtain more reliable results.

Exercises

1. Consider the two sequences ATGAC and ACGC.
- **a.** List every possible alignment in which ATGAC has no blanks.
- **b.** List every possible alignment in which ATGAC has at least one blank and ACGC has only blank(s) at the end.

2.
- **a.** Trace through the *score* algorithm for the strings and similarity array in Quick Review Question 4 and Figure 11.
- **b.** Trace through the *align* and *recAlign* algorithms for the strings and similarity array in Quick Review Question 5 and Figure 12.

3.
- **a.** Develop the similarity array for strings s = CGTA and t = GCATG using the scoring of +1, -1, -2 from the example in this module.
- **b.** What is the similarity score?
- **c.** Give the optimal alignment that *align* returns.
- **d.** Give all optimal alignments for these strings.

4. Repeat Exercise 1 for strings s = CGTA and t = GCATG.

5. Repeat Exercise 1 for strings s = ATGAC and t = ACGC using the scoring that a mismatch or space costs 1, while a match has score 2.

6. What is the complexity of *score?*

7. What is the complexity of *recAlign?*

8. Change the priority of the *recAlign* algorithm so that when a choice exists, the algorithm gives priority to going vertically up, next to going on the diagonal, and finally to going to the left. Because priority is given to going up, the resulting alignment is called the "upmost alignment." Give the additional parameters that *align* and *recAlign* need.

9. Change the priority of the *recAlign* algorithm so that when a choice exists, the algorithm gives priority to going to the left, next to going on the diagonal, and finally to going vertically up. Because priority is given to going to the left, the resulting alignment is called the "down most alignment." Give the additional parameters that *align* and *recAlign* need.

Projects

1.
- **a.** Develop a program using the Needleman-Wunsch Algorithm that returns the similarity and an optimal alignment for pairs of sequences.
- **b.** Time the generation of the similarity matrix for different lengths of sequences as in Table 3 and Figure 14.

2. Complete Project 1, except modify *align* so that the procedure returns all optimal alignments. Hint: Use a stack and backtracking.

3. Write a program to decide on the fastest method for reading two sequences:
- The root reads and broadcasts both sequences.
- One process reads and broadcasts one sequence, while another process reads and broadcasts the other sequence.
- Even processes read sequence 1 and then sequence 2, while the odd processes read the sequences in the reverse order.
- Each process reads both sequences.

For timings, do each experiment at least 100 times, averaging the results.

4. **a.** Implement the first version of the pipeline algorithm, where each process calculates on matrix element and then communicates the result.
b. For sequences of size at least 5,000, calculate and graph the speedup of the calculation of the similarity value for an increasing number of processes. Run the timing experiments multiple times, averaging the results.

5. **a.** Implement the second version of the pipeline algorithm, where a process calculates a submatrix of size *band* × *block* and then communicates the results on the last submatrix row.
b. For large sequences and a given number of processes, determine advantageous band and block sizes by fixing one size and increasing the other.

Acknowledgements

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

References

ACS (American Cancer Society). 2015. http://www.cancer.org/ (accessed Sept. 14, 2015)

Antoniou A, Pharoah PD, Narod S, et al. Average risks of breast and ovarian cancer associated with BRCA1 or BRCA2 mutations detected in case series unselected for family history: a combined analysis of 22 studies. 2003. *Am J Hum Genet*. May 2003;72(5):1117-30.

B10NUMB3R5, The Database of Useful Biological Numbers. 2015. http://bionumbers.hms.harvard.edu/bionumber.aspx?&id=105336&ver=2 (Accessed Sept. 14, 2015)

Boumphreyfr (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], via Wikimedia Commons. 2009. https://commons.wikimedia.org/wiki/File:Meiosis_crossover.png (accessed Sept. 14, 2015)

BRCA1, Genetics Home Reference. 2015. http://ghr.nlm.nih.gov/gene/BRCA1 (accessed Sept. 14, 2015)

Chen S. and G. Parmigiani. Meta-analysis of BRCA1 and BRCA2 pene- trance. J Clin Oncol. Apr 10 2007;25(11):1329-33.

Chen, Yang, Songnian Yu, Ming Le. 2006. "Parallel Sequence Alignment Algorithm for Clustering System" in *International Federation for Information Processing (IFIP)*, Volume 207, *Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management*, eds. K. Wang, Kovacs G., Wozny M., Fang M., (Boston: Springer), pp. 311-321.

Couch, Fergus J., Katherine L. Nathanson, and Kenneth Offit. "Two decades after BRCA: setting paradigms in personalized cancer care and prevention." *Science (New York, NY)* 343, no. 6178 (2014): 1466-1470.

Dna_strand3.png: Boumphreyfr derivative work: Vojtech.dostal (Dna_strand3.png) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], via Wikimedia Commons. 2011. https://commons.wikimedia.org/wiki/File:Dna_strand3_cs.png (accessed Sept. 14, 2015)

Ezkurdia, Iakes, David Juan, Jose Manuel Rodriguez, Adam Frankish, Mark Diekhans, Jennifer Harrow, Jesus Vazquez, Alfonso Valencia, and Michael L. Tress. "Multiple evidence strands suggest that there may be as few as 19 000 human protein-coding genes." *Human molecular genetics* 23, no. 22 (2014): 5866-5878.

LLNL (Lawrence Livermore National Laboratory). 2004. "The Art of Protein Structure Prediction," UCRL-52000-04-12. https://str.llnl.gov/str/December04/Fidelis.html (accessed July 24, 2015)

Miki, Y., J. Swensen, D. Shattuck-Eidens, P. A. Futreal, K. Harshman, S. Tavtigian, Q. Liu, C. Cochran, L. M. Bennett, W. Ding, R. Bell, J. Rosenthal, and 33 others. 1994. "A strong candidate for the breast and ovarian cancer susceptibility gene BRCA1." *Science* 266: 66-71. [PubMed: 7545954]

Mrbean427 (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], via Wikimedia Commons. 2008. https://commons.wikimedia.org/wiki/File:Nitrogenous_bases.jpg (accessed Sept. 14, 2015)

NCBI, "BRCA1 breast cancer 1, early onset [*Homo sapiens* (human)]" 2015. http://www.ncbi.nlm.nih.gov/gene/672 (accessed Sept. 14, 2015)

Needleman, S. B. and C. D. Wunsch. 1970. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *J. Molecular Biology*, vol. 48, pp. 443-453.

Schwartz GF, Hughes KS, Lynch HT, et al. Proceedings of the international consensus conference on breast cancer risk, genetics, & risk management, April, 2007. *Cancer*. Nov 15 2008;113(10):2627-37.

Turnbull C, Rahman N. Genetic predisposition to breast cancer: past, present, and future. *Annu Rev Genomics Hum Genet*. 2008;9:321-45.

UniProt, "UniProtKB - P38398 (BRCA1_HUMAN)." 2015. http://www.uniprot.org/uniprot/P38398#section_comments (accessed Sept. 14, 2015)

The U.S. National Library of Medicine (National Institutes of Health (NIH)) list 26 different genes associated with breast cancer. http://ghr.nlm.nih.gov/condition/breast-cancer/show/Related+Gene(s) (accessed Sept. 14, 2015)

Yikrazuul (Own work) [Public domain], via Wikimedia Commons. 2008. https://commons.wikimedia.org/wiki/File:Peptide_bond_formation.svg (accessed Sept. 15, 2015)

Answers to Quick Review Questions

1. CATGGA

2.

- **a.** DNA
- **b.** nucleotide
- **c.** nucleotide
- **d.** A, G, C, T
- **e.** A, G, C, U
- **f.** A, G
- **g.** C, T, U
- **h.** mRNA
- **i.** T
- **j.** U
- **k.** G
- **l.** A
- **m.** A
- **n.** C
- **o.** RNA
- **p.** DNA
- **q.** base pair (bp)

3.

- **a.** proteins
- **b.** enzymes
- **c.** amino acids
- **d.** peptide bonds
- **e.** peptide bonds
- **f.** residue
- **g.** N-terminal
- **h.** C-terminal
- **i.** polypeptide

4.

- **a.** 3
- **b.** 4
- **c.** 20
- **d.** 64

5.

- **a.** chromosome
- **b.** gene

c. genome
d. triplet
e. protein synthesis
f. nucleus
g. nucleus
h. mRNA
i. transcription
j. cytoplasm
k. ribosome
l. translation
m. cytoplasm
n. codon

6.

a. mutation
b. tumor suppressor gene
c. oncogene
d. proto-oncogene
e. oncogene

7.

a. recombination or crossing-over
b. genetic or linkage map
c. map unit
d. restriction site
e. genomics
f. bioinformatics

8. $0 = 1 + (-1) + 1 + (-2) + 1$

9. **a.** 1; because the third row and column headings agree, we have $a[2][2] + 1 = 0 + 1 = 1$.
b. –2 because from above, $a[2][3] + (-2) = 0 + (-2) = -2$.
c. -4 because from the left, $a[3][2] + (-2) = (-2) + (-2) = -4$.
d. Diagonal. $a[3][3]$ is the maximum of 1, -2, and -4, which comes from the diagonal. Thus, we draw a line segment between $a[2][2]$ and $a[3][3]$.

10. **a.** -7 because (-5) + (-2) = -7.
b. -1 because 1 + (-2) = -1.
c. -1 because 1 + (-2) = -1.
d. -1
e. Left and diagonal

11. Figure 11

12. **a.** ATGAC
b. ACGC
c. 1

d. -1
e. -2
f. 6
g. 5
h. column 0
i. -6
j. a row at a time
k. G
l. C
m. -1
n. -1
o. -4
p. -1
q. -1

13. **a.** 5
b. 4
c. ATGAC
d. ACG-C
e. -1
f. -1
g. 3, 3
h. ATGA
i. ACG-
j. 1
k. -4 Thus, *a*[*i*][*j*] does not equal *a*[*i* - 1][*j*] + *spacePenalty*.
l. -4 Thus, *a*[*i*][*j*] does not equal *a*[*i*][*j* - 1] + *spacePenalty*.
m. 0, 0
n. *null*
o. A
p. A

14. **a.** 60 = 10 * 6; 15 = 6 + 9
b. 99,900 = 300 * 333; 632 = 333 + 299
c. 25,000,000 = 5000 * 5000; 9999 = 5000 + 4999
d. 91,1738,000 = 30,200 * 30,190; 60,389 = 30,190 + 30,199

15. **a.** 31
b. Process 0, Process 1
c. Process 20, Process 21
d. Process 30, Process 31
e. 31 × 31
f. 31 · 31 = 961
g. 61 = 1 (first column) + 30 (as Process 0 travels through the first row) + 30 (for the other processes to finish calculating their rows)
h. O(*n*)

i. 60 = 1 (first column) + 30 (as Process 0 travels through the first row) + 29 (for the other processes to finish communicating). Recall that the last process does not communicate its values.

16. **a.** 288 = 18 * 16
b. 33 = 18 + 15
c. 32; the last row does not communicate its calculations
d. 3
e. 22 = 1 (for the first column elements) + 5 (for the 5 blocks of size 3 across the first row) + 16 (to complete the other rows); the last row does not communicate its blocks
f. 5
g. 20 = 1 (for the first column elements) + 3 (for the 3 blocks of size 5 across the first row) + 16 (to complete the other rows) ; the last row does not communicate its blocks

17. **a.** 3 = 18 / 6
b. 18 = 6 * 3
c. 3
d. 7 = 1 (first column) + 5 (as Process 0 travels through the first band) + 1 (for the other processes to finish communicating). Recall that the last process does not communicate its values.
e. 6 = 18 / 3
f. 15 = 3 * 5
g. 5
h. 8 = 1 (first column) + 3 (as Process 0 travels through the first band) + 4 (for the other processes to finish communicating). Recall that the last process does not communicate its values.

18. **a.** 6 because the matrix has 31 rows
b. Process 3, no communication
c. 36
d. 6
e. 367 = 1 (first column) + $5 \cdot 36$ (as Process 0 travels through the first row with 6×6 submatrix sizes) + $5 \cdot 36$ (for the other processes, except the last process, to finish calculating their submatrices) + 6 (for the last process to finish calculating its last submatrix of 6 elements)
f. 10 = 1 (first column) + 5 (as Process 0 travels through the first row with block sizes of 6) + 4 (for the other processes to finish communicating). Recall that the last process does not communicate its values.