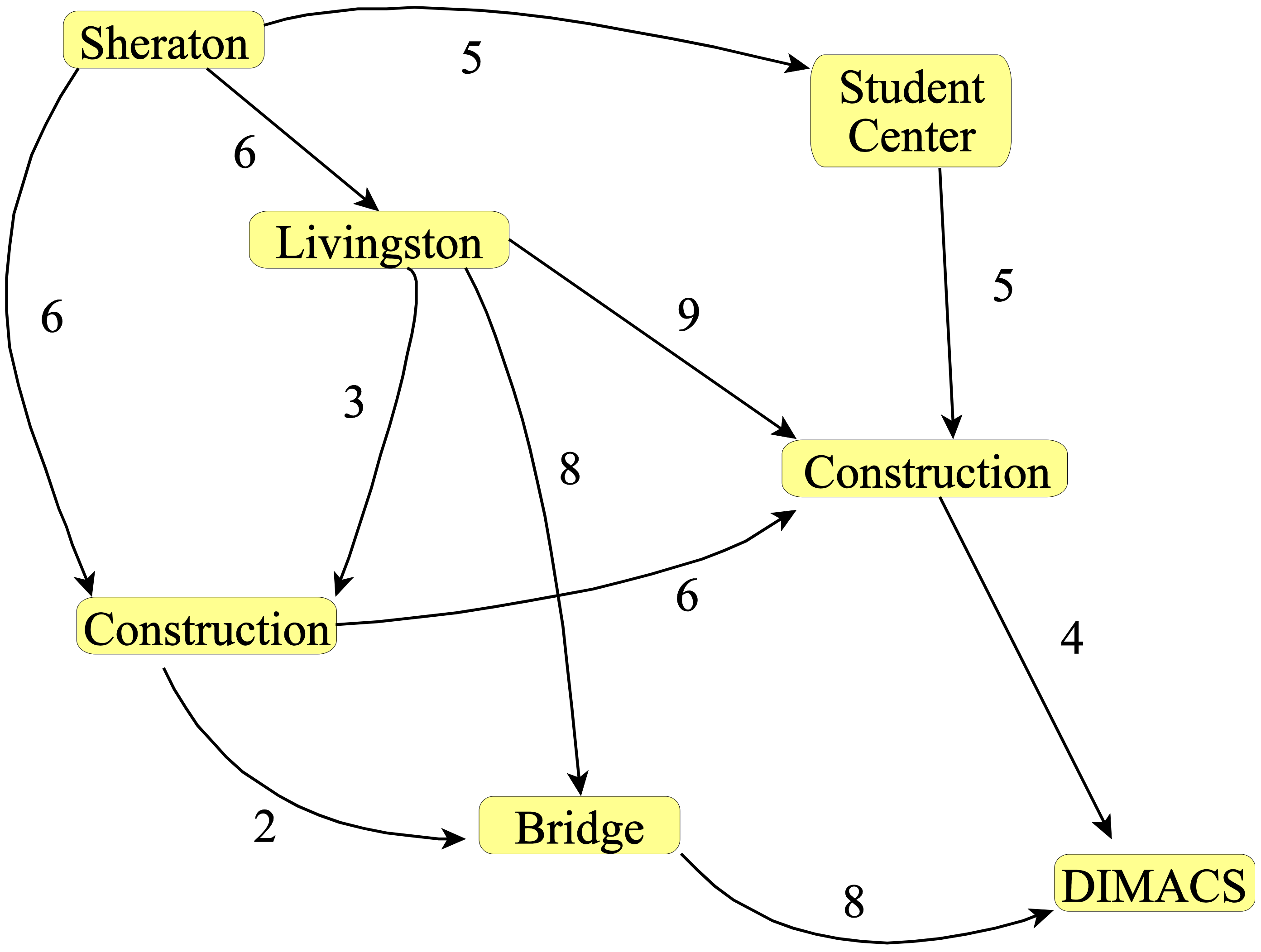
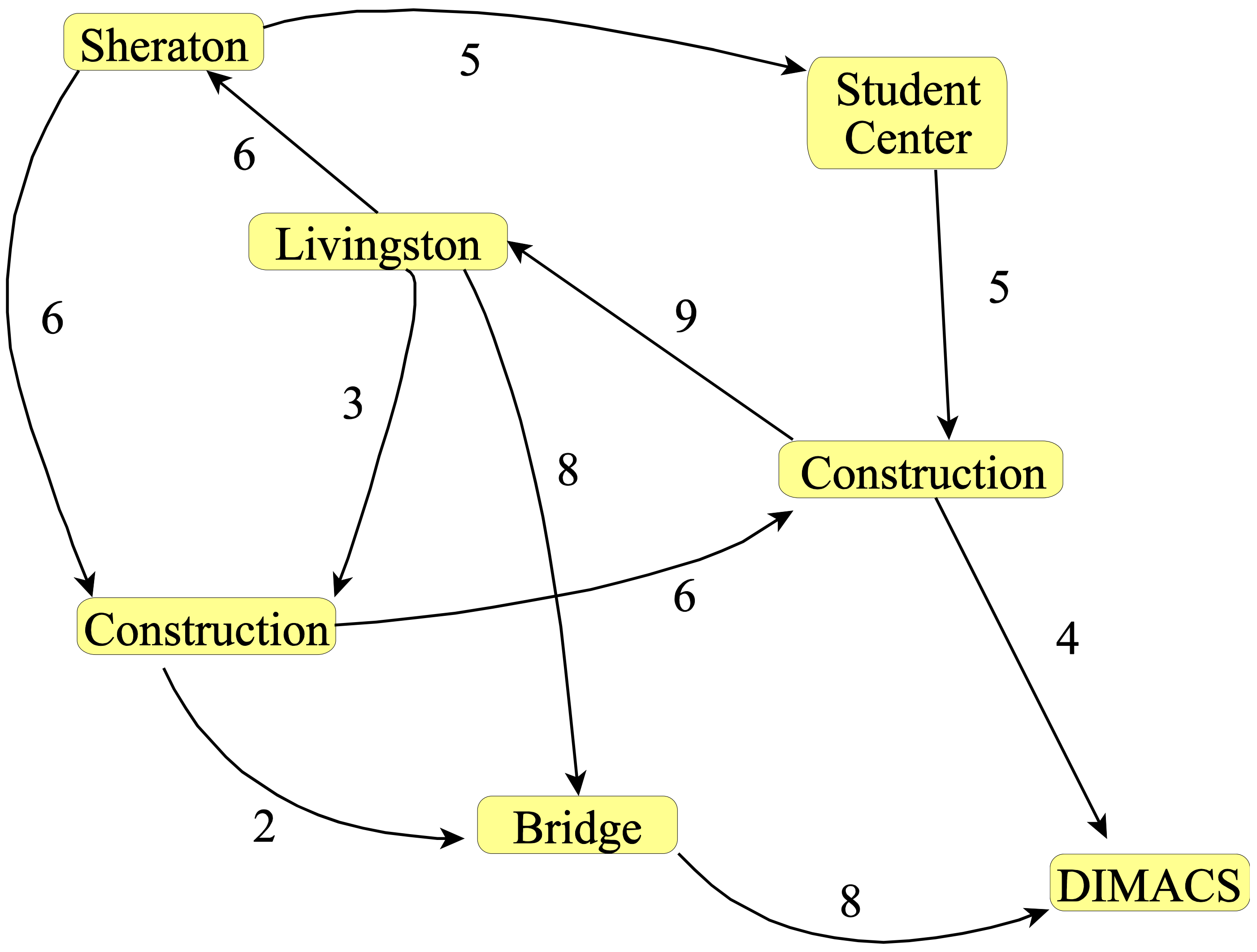
Shortest Path Algorithm



- Works just fine on this graph.
- Length of shortest path = _____

Same Questions, Different Map



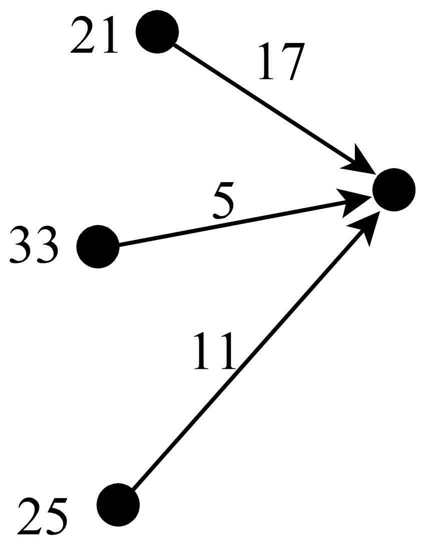
- Why does the algorithm fail on this graph?

Dijkstra on an Acyclic Directed Graph

Given an acyclic, directed graph with weights on the edges, and some start vertex A, let *d*(*v*) denote for each vertex *v* the length of the shortest path from A to *v*.

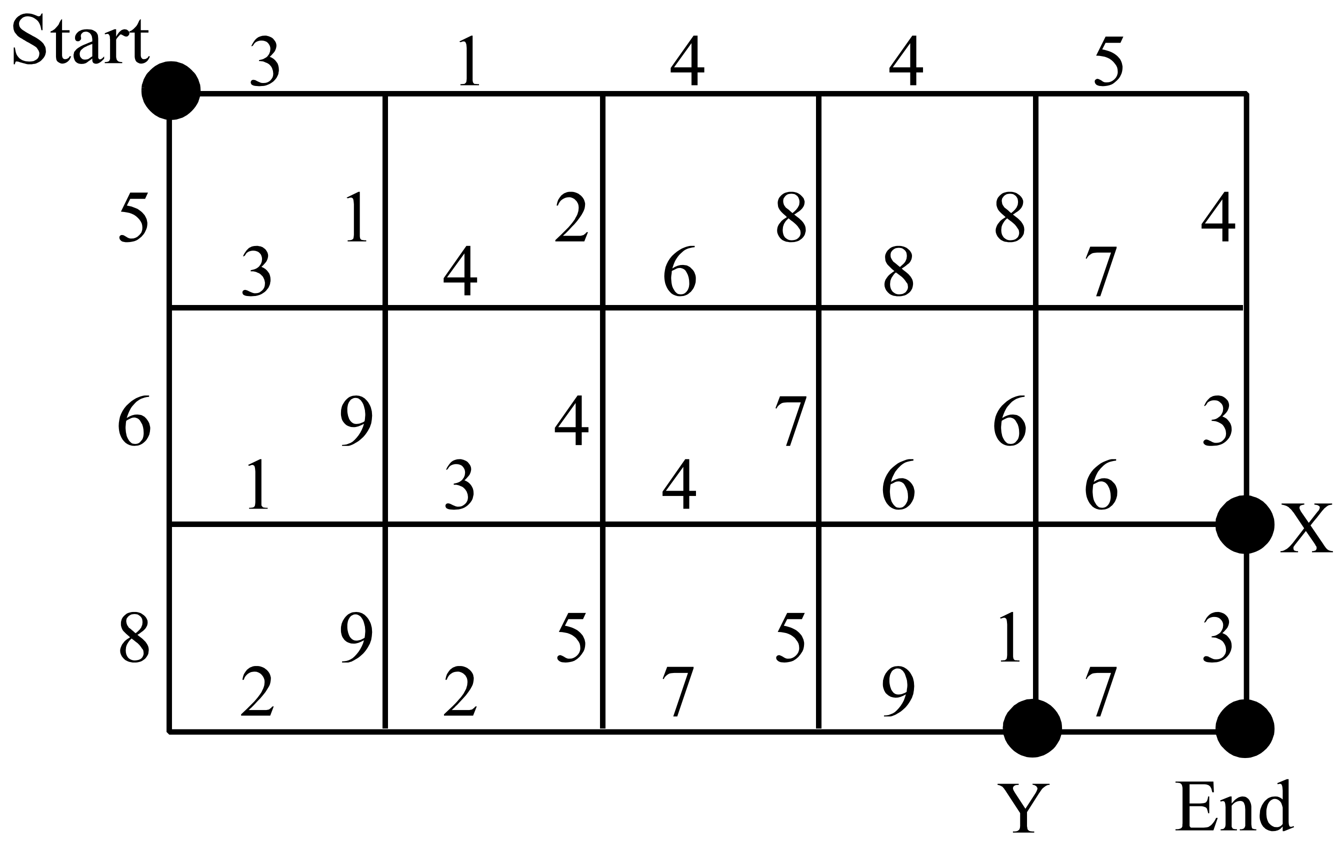
We find all these distances as follows:

- Let *d*(A) = 0
- For all other vertices *v*, examine all arcs (*u*, *v*) directed into *v*, compute the sum:
  $$d(u) + \text{weight}(u, v)$$
  and let *d*(*v*) be the minimum of these values



Dynamic Programming

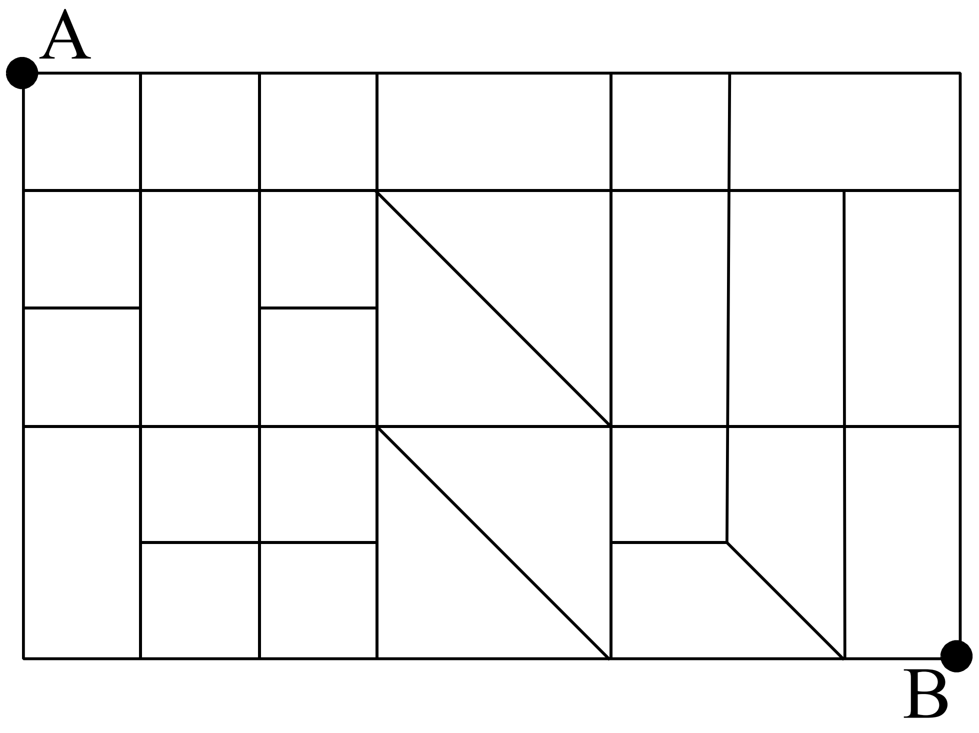
The term "Dynamic programming" refers to the method of finding optimal solutions to a large problem by solving several smaller problems and keeping track of those smaller solutions, usually in order to reuse them.



For example, the shortest path to "End" could be found if we kept track of the shortest paths to "X" and "Y."

Counting Paths

How many paths are there from A to B which move in a generally South/East direction?



Again, a straightforward dynamic programming solution solves this problem relatively quickly.

Global String Alignment

Given two strings, find their best alignment.

Suppose our strings are "`AGCGT`" and "`CAGT`." An alignment consists of writing the strings, in order, so that the letters line up in some way. If the strings have different lengths, as is the case here, then we will need to insert some "gap symbol" in the shorter string. We use "–" as our gap symbol.

`AGCGT` `-CAGT`	`AGCGT` `C-AGT`	`AGCGT` `CA-GT`	`AGCGT` `CAG-T`
`AGCGT` `CAGT-`	`AGCG-T` `C--AGT`	`-AGCGT` `C--AGT`	`AGC-GT` `--CAGT`

Sometimes we will insert gap symbols just to make things line up better, regardless of the lengths.

Which alignment is the best?

A *scoring criterion* is used to evaluate the quality of an alignment.

Gratuitous Use of Gaps

Suppose we wished to align the strings:

`GGAGTCCACCTGTGAAACAATA`, and
`ACGCGCGTCCTCCTGTGACAATT`

Notice here how the insertion of some gaps can help us get a good alignment of a fairly lengthy region of similarity:

```
GGA---GTCCACCTGTGAAACAATA
ACGCGCGTCCTCCTGTGA--CAATT
```

We will thus allow for insertion of gaps wherever it will help us to align our sequences more optimally.

Scoring Criterion

`AGCGT` `-CAGT`	`AGCGT` `C-AGT`	`AGCGT` `CA-GT`	`AGCGT` `CAG-T`
`AGCGT` `CAGT-`	`AGCG-T` `C--AGT`	`-AGCGT` `C--AGT`	`AGC-GT` `--CAGT`

Look at the columns of the alignmnet. Score:

+2 For each alignment of matched letters
–1 For each alignment of mismatched letters
–2 For each alignment of a letter with a gap
$-500 For each alignment of two gaps

Given this scoring system, how do we find the optimal alignment. How do we find all optimal alignments?

Are There Many Possible Alignments?

We wish to align `ABCDEFG` with `TUVWXYZ`.

Our first column can align either:
- A with T
- A with a gap
- a gap with T

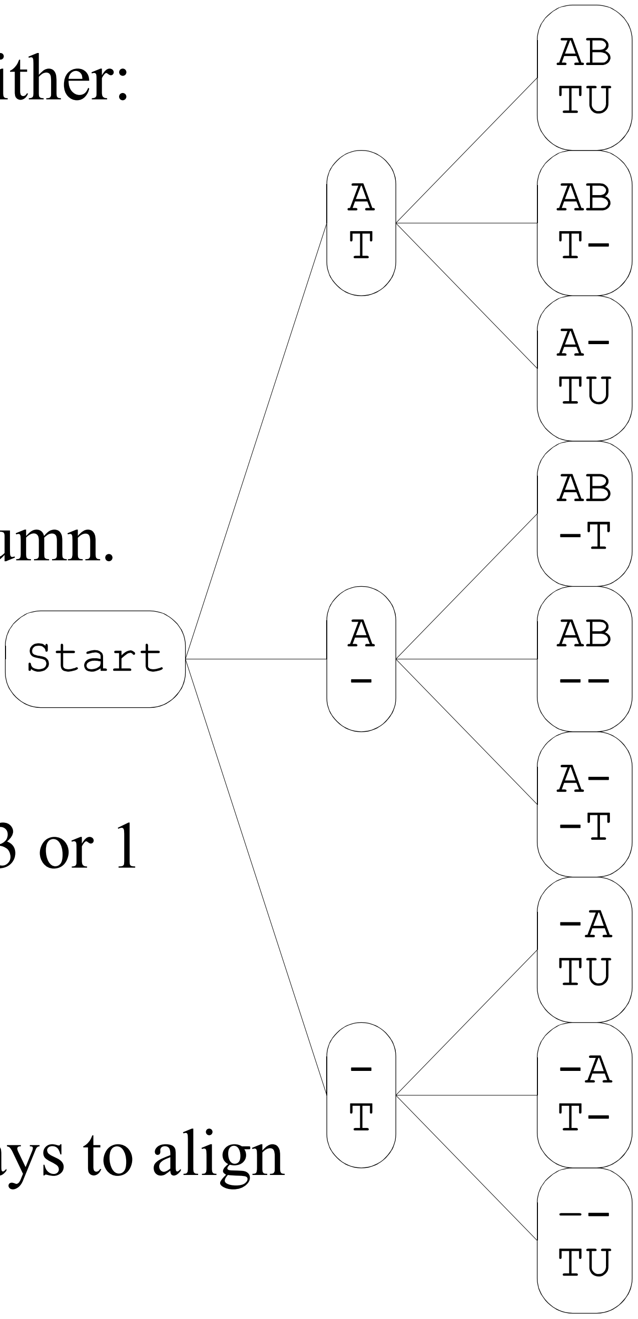
for 3 possible choices.

Similarly for the second column.

And the 3rd through 7th.

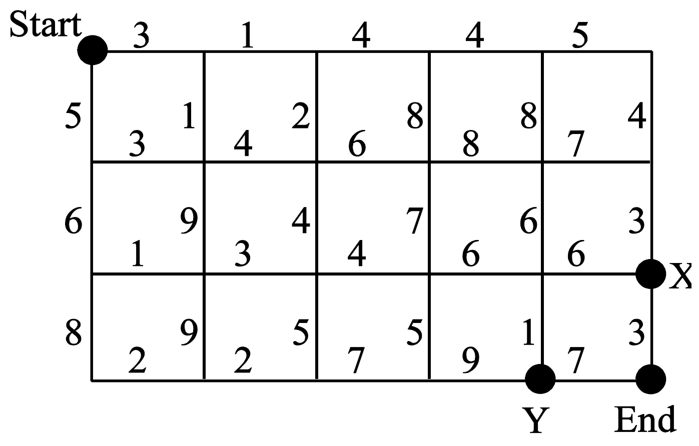
The 8th column could have 3 or 1 choices, depending on what happened before.

Thus, there are at least 3^7 ways to align these strings.



In general, if both strings are of length at least k, then there are at least 3^k ways to align them.

Dynamic Programming to the Rescue



Recall what we said about the figure to the right: If we knew the best paths to X and Y, we could easily find the best path to End.
What information would be analogously helpful in attempting to find an optimal alignment of `AGCGT` and `CAGT`?

What could our last column look like?

Either $\boxed{\begin{array}{c}\texttt{T}\\ \texttt{T}\end{array}}$, $\boxed{\begin{array}{c}\texttt{T}\\ -\end{array}}$ or .$\boxed{\begin{array}{c}-\\ \texttt{T}\end{array}}$.

So what optimal alignments would you like to know?

`AGCG` + `T`
`CAG` `T`
	`AGCG` + `T`
	`CAGT` `–`
		`AGCGT` + `–`
		`CAG` `T`

`AGCG`
`–CAG`
`Cost: –2`
	`–AGCG`
	`CAGT–`
	`Cost: –1`
		`AGCGT`
		`C–AG–`
		`Cost: –4`

How Did We Get Those Costs?

AGCG -CAG Cost: -2	-AGCG CAGT- Cost: -1	AGCGT C-AG- Cost: -4

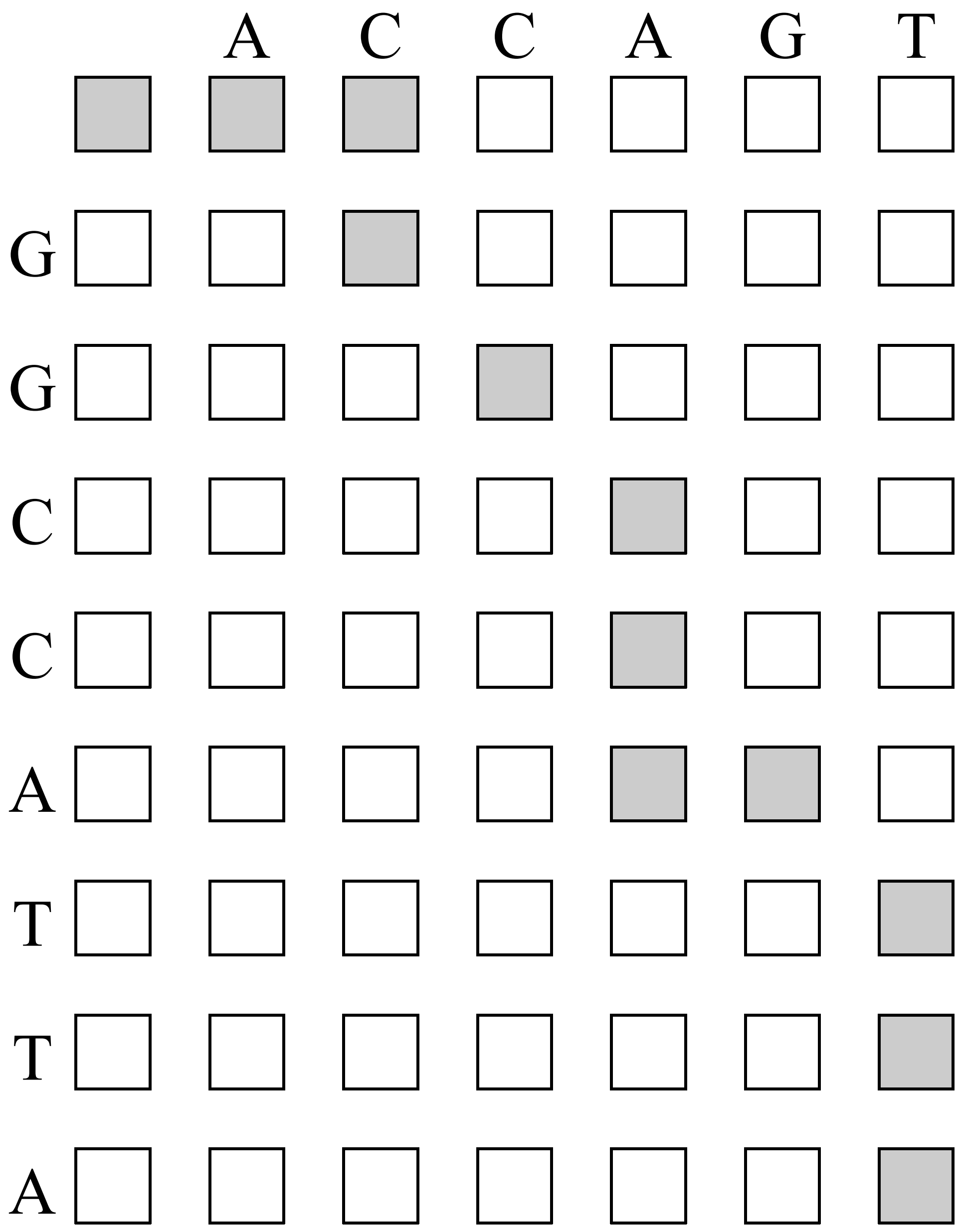
$\begin{array}{l}\texttt{AGC}\\\texttt{CAG}\end{array} + \begin{array}{l}\texttt{G}\\\texttt{T}\end{array}$	$\begin{array}{l}\texttt{AGC}\\\texttt{CAGT}\end{array} + \begin{array}{l}\texttt{G}\\\texttt{-}\end{array}$	$\begin{array}{l}\texttt{AGCG}\\\texttt{CAG}\end{array} + \begin{array}{l}\texttt{-}\\\texttt{T}\end{array}$
-AGC CAG- Cost: 0	-AGC CAGT- Cost: 1	AGCG C-AG Cost: -2

This suggests that, as in the case of our Dijkstra's algorithm, we can "reuse" our solutions to smaller sub-problems to solve the larger problems.

The idea:
Find an optimal alignment between *each* pair of prefixes for each of the two strings.

```
ACGAAGACTTTTTACA
GGCTACGATACGA
```

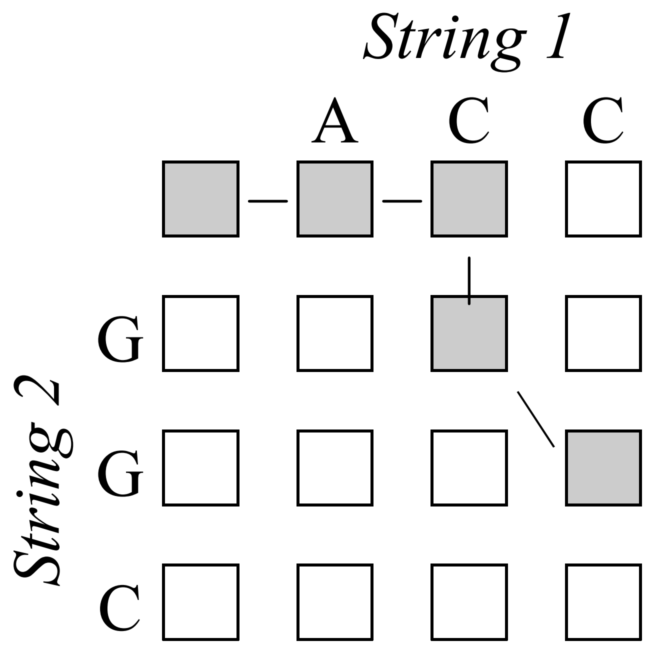
A Nifty Picture



This diagram corresponds to the alignment:

```
AC-CA--GT--
--GGCCA-TTA
```

The Correspondence



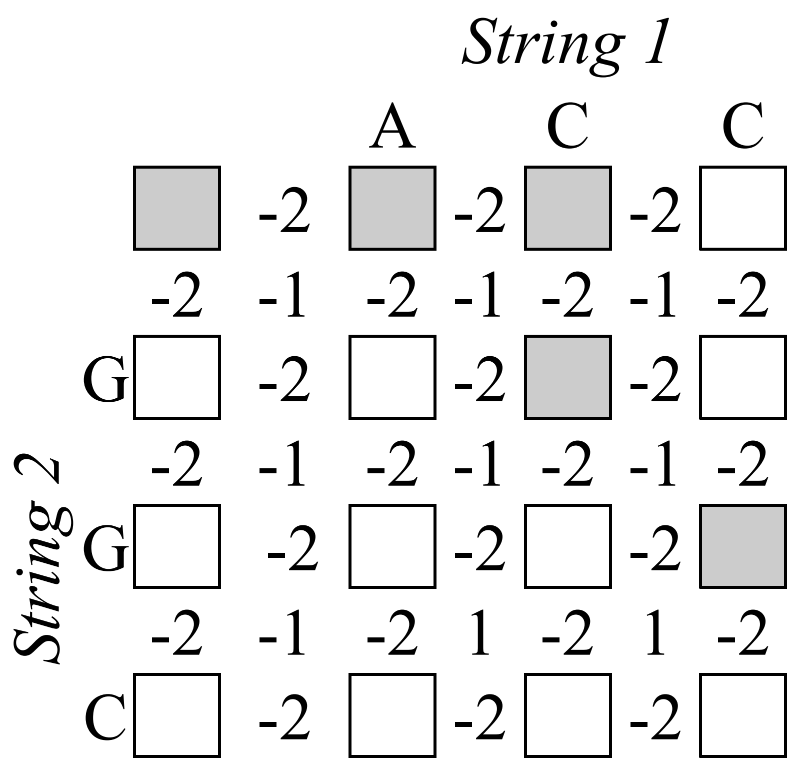
- Each column of the alignment corresponds to a transition from one shaded box to the next, having the letters of the new row and/or new column

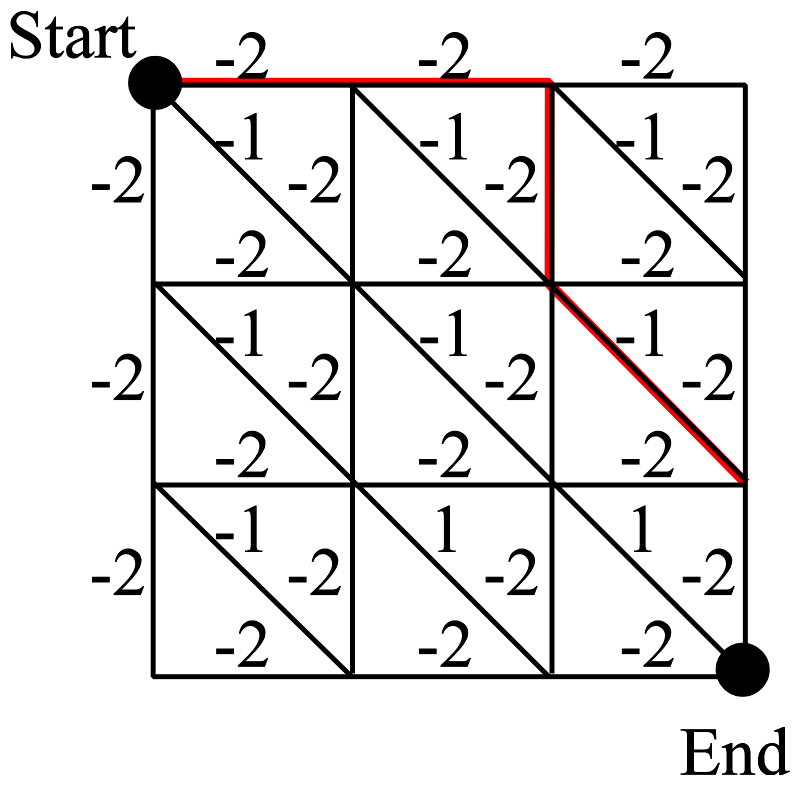
- Moving horizontally corresponds to aligning a letter in string 1 with a gap in string 2

- Moving vertically corresponds to aligning a letter in string 2 with a gap in string 1

- Moving diagonally corresponds to aligning the two characters, one from each string.

The Cost of Each Step





An Optimal Alignment is a Best Path

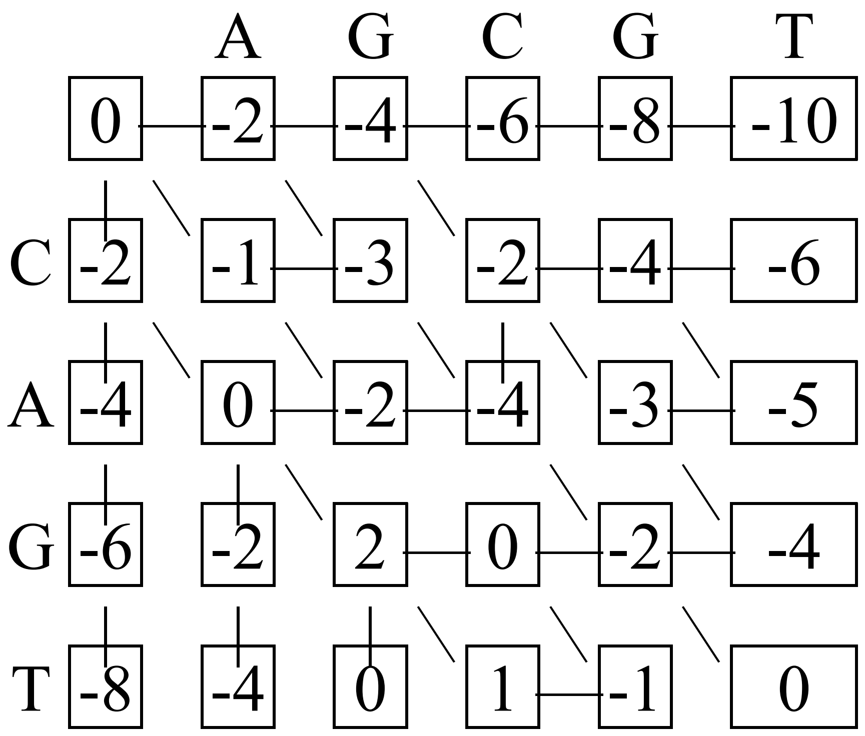
So we can find an optimal alignment by finding the best path in this graph

Which can be done by dynamic programming, determining the "distance" to each cell by considering the distances to its neighbors. Let's try this together:

Match	+2
Mismatch	−1
Gap	−2

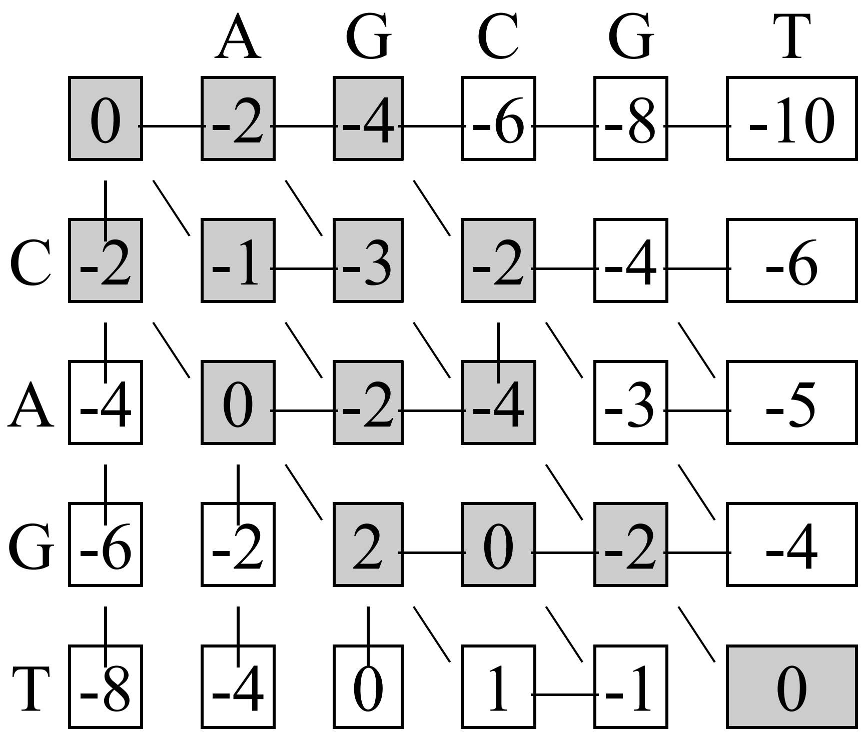
		A	G	C	G	T
C						
A						
G						
T						

Reconstructing the Alignment



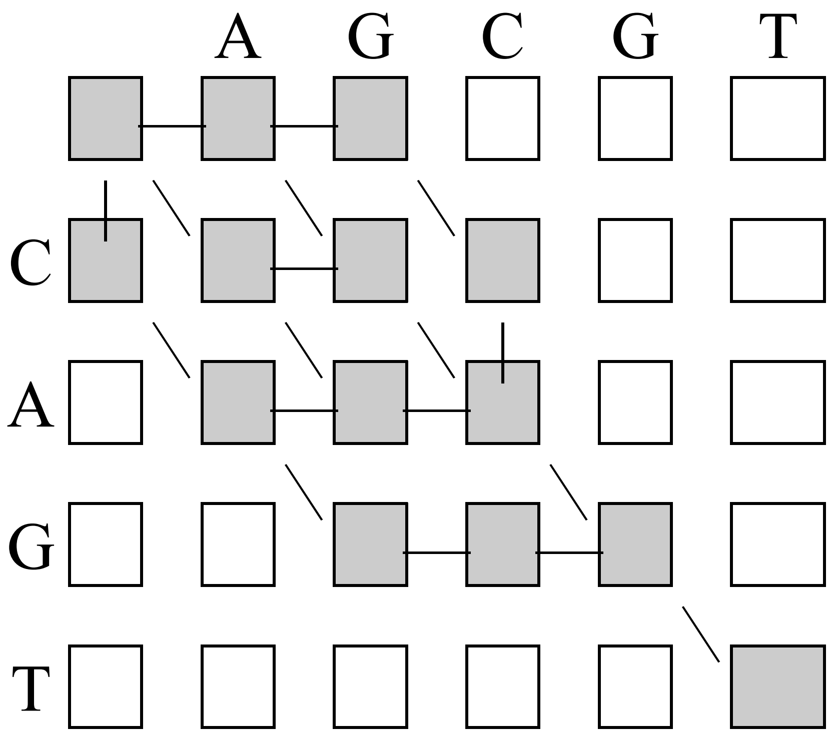
When each cell is optimized, record all (1, 2 or 3) cells from which that optimum can be achieved.

Reconstructing the Alignment

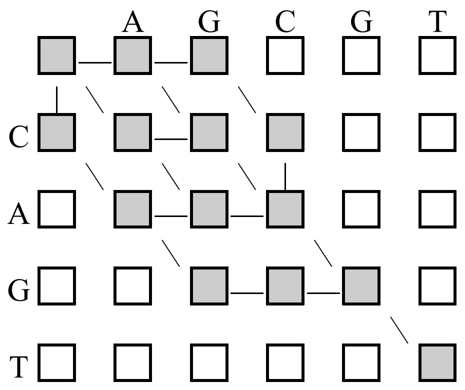


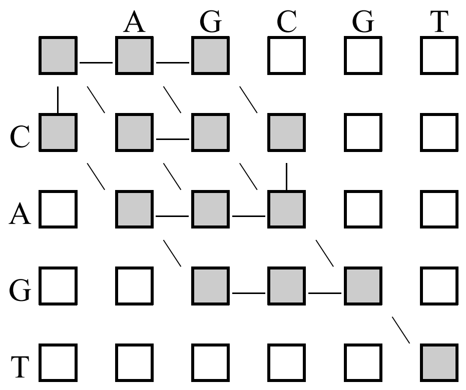
- Shade the bottom-right cell
- For each other cell, shade it if there is a directed edge *from* that cell *to* a shaded cell
- These shaded cells form some subgraph of the original lattice
- (These should be figured bottom-to-top, right-to-left)
- All paths in this subgraph correspond to optimal alignments

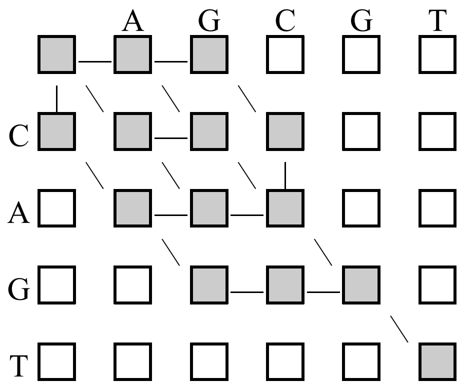
Counting the Optimal Alignments

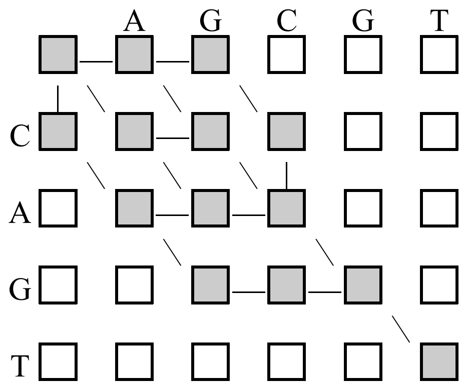


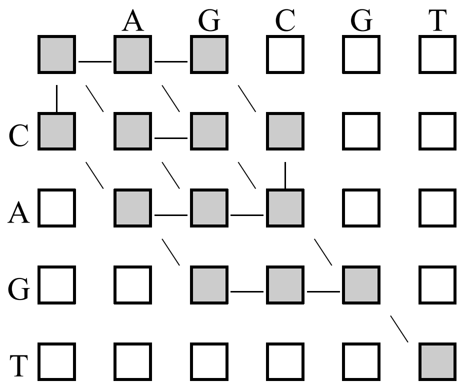
Building the Optimal Alignments

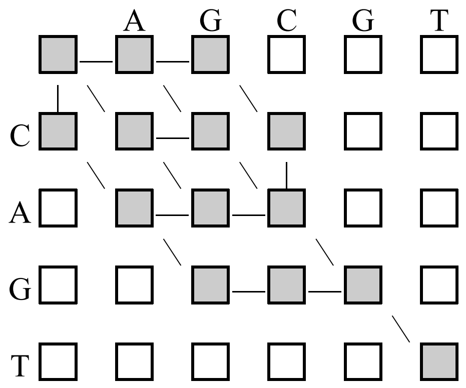






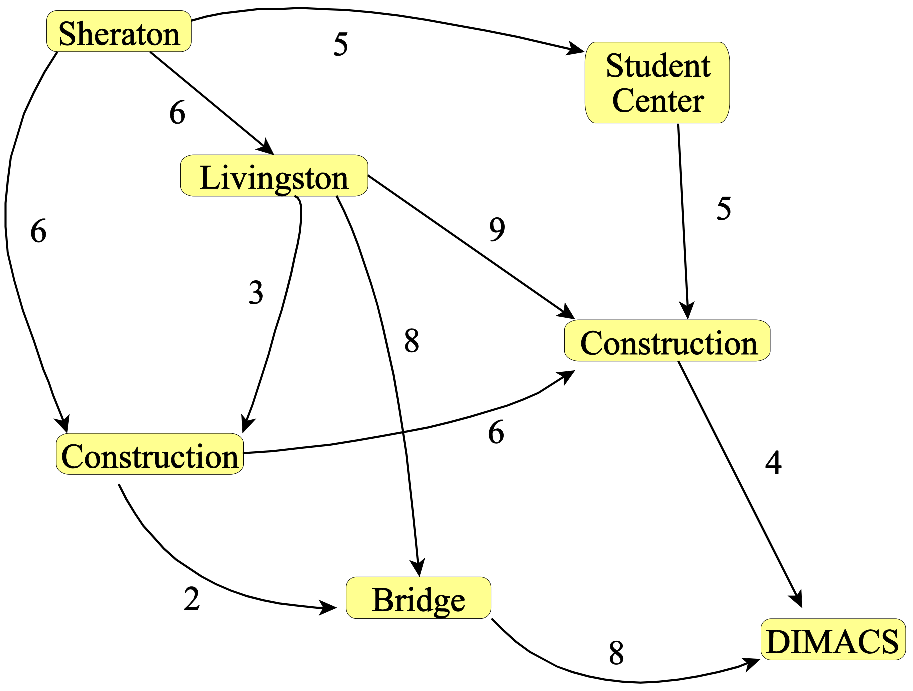


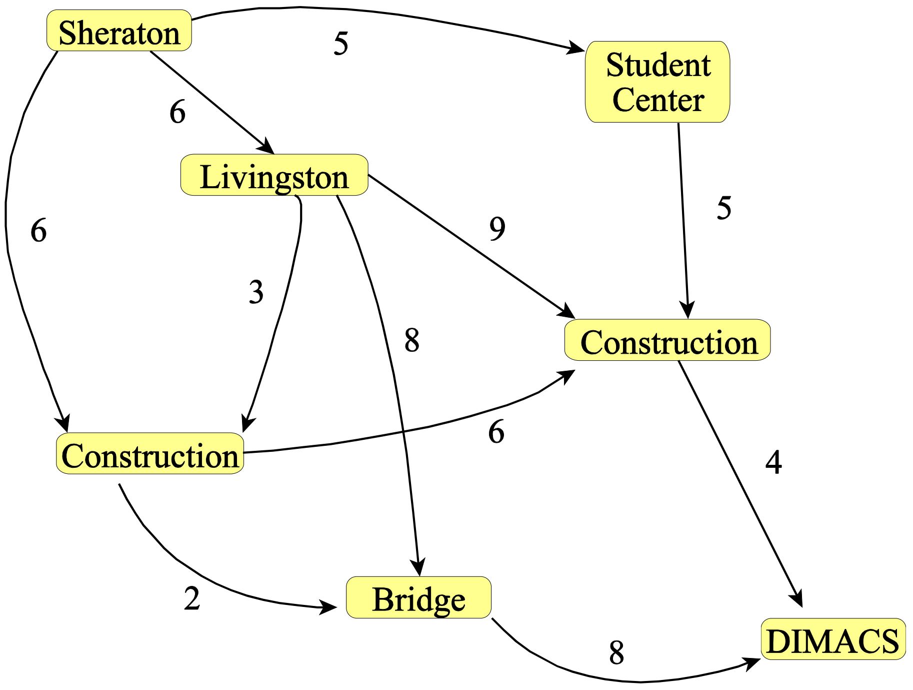




Handout #1 — Shortest Path from the Hotel

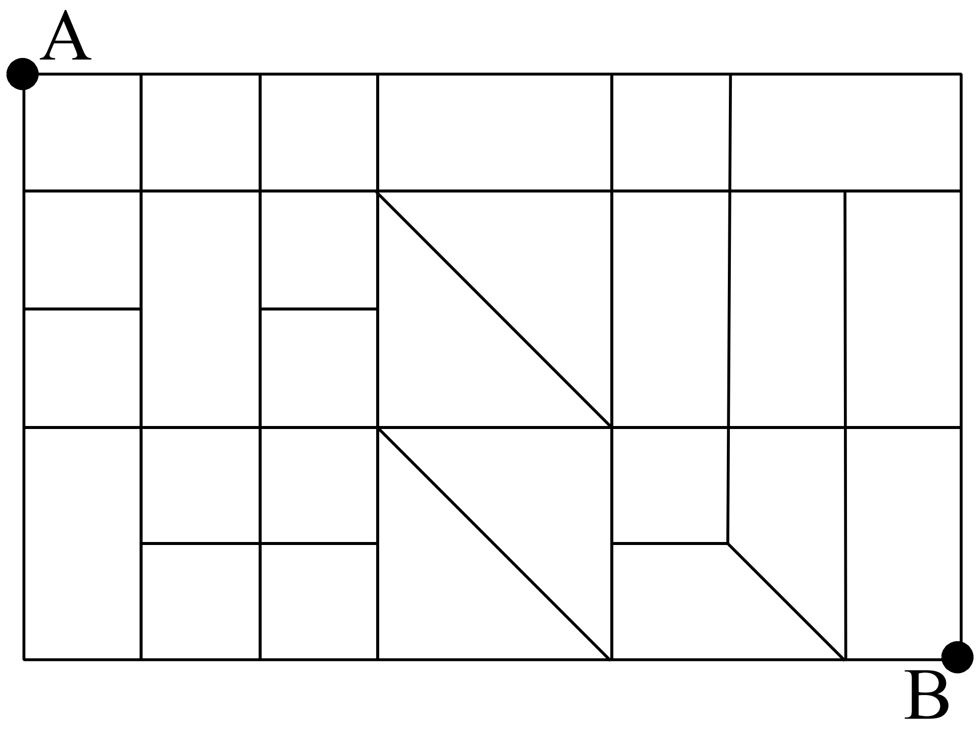
What is the length of the shortest path from the hotel to DIMACS? Use whatever method you wish to find the answer. (Two copies of the graph are provided.)

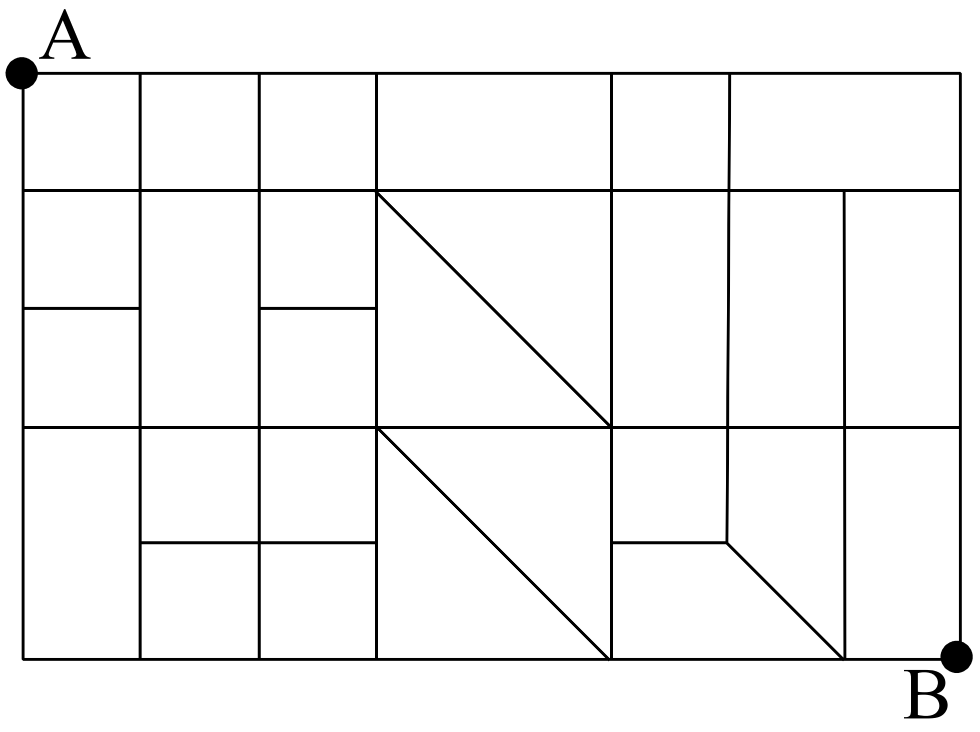




Handout #2 — Counting Paths

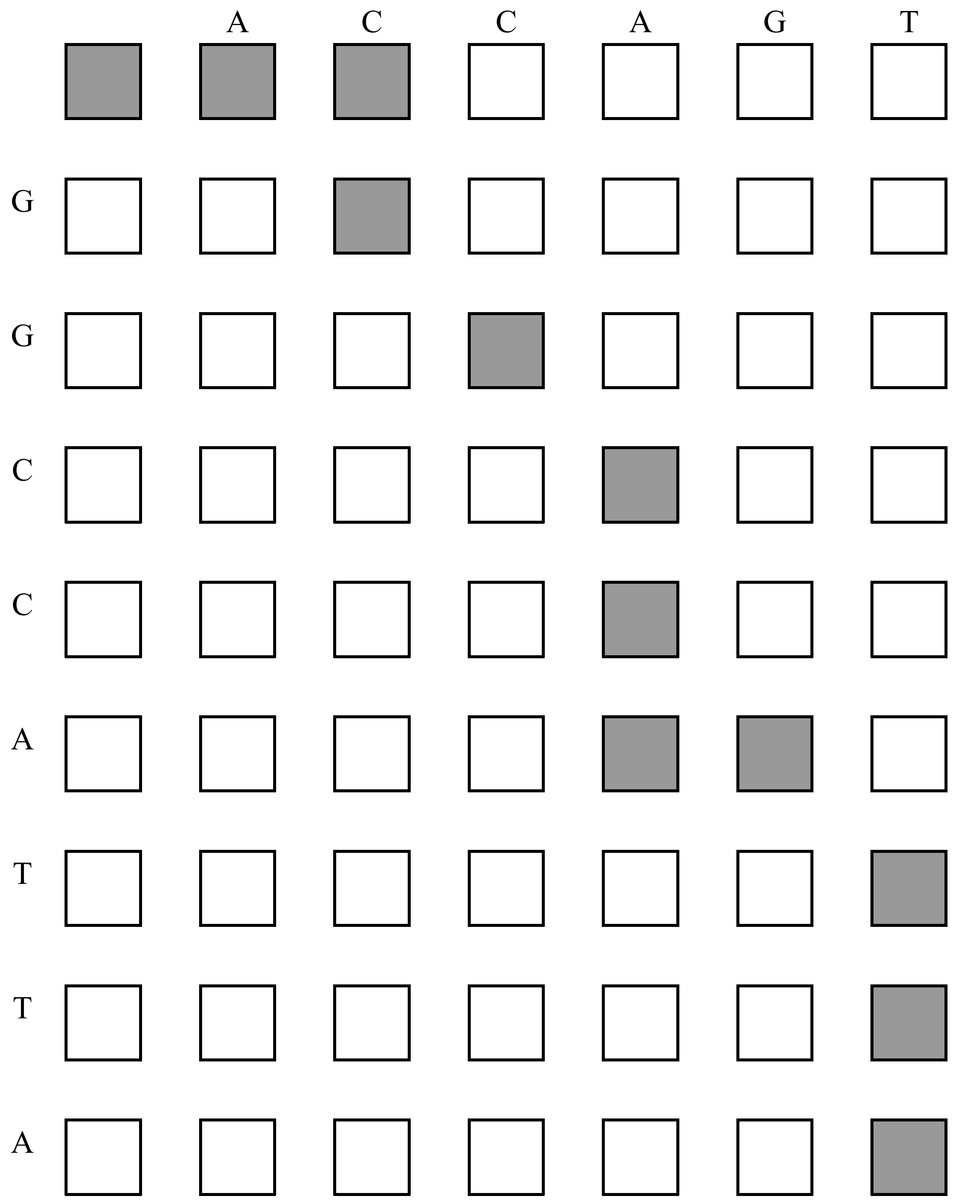
How many ways are there to walk from A to B on this graph, if you are not allowed to "backtrack." That is, all steps should be toward either the East, the South or the Southeast.





Handout #3 — The Correspondence Between Paths and Alignments

What alignment is indicated by the shaded boxes below?



Handout #4 — Finding the Optimal Alignment via Dynamic Programming

Use dynamic programming to find the best alignment (that with the highest score) between the two indicated strings.

Match	+2
Mismatch	−1
Gap	−2

		A	G	C	G	T
C						
A						
G						
T						

Handout #5 — Enumerating the Optimal Alignments from our Table

Here is the completed dynamic programming matrix for these two strings. (Two copies are proided.) How many optimal alignments are there? What are all of the optimal alignments?

Match	+2
Mismatch	−1
Gap	−2

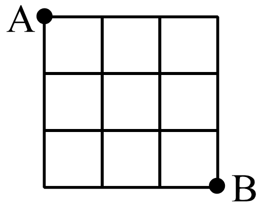
$$\begin{tikzcd}
 & & A & G & C & G & T \\
 & \boxed{0} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] \arrow[dr, dash] & \boxed{-4} \arrow[r, dash] \arrow[dr, dash] & \boxed{-6} \arrow[r, dash] & \boxed{-8} \arrow[r, dash] & \boxed{-10} \\
C & \boxed{-2} \arrow[d, dash] \arrow[dr, dash] & \boxed{-1} \arrow[r, dash] \arrow[dr, dash] & \boxed{-3} \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{-4} \arrow[r, dash] \arrow[dr, dash] & \boxed{-6} \\
A & \boxed{-4} \arrow[d, dash] & \boxed{0} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] & \boxed{-4} \arrow[dr, dash] & \boxed{-3} \arrow[r, dash] \arrow[dr, dash] & \boxed{-5} \\
G & \boxed{-6} \arrow[d, dash] & \boxed{-2} \arrow[d, dash] & \boxed{2} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{0} \arrow[r, dash] \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] \arrow[dr, dash] & \boxed{-4} \\
T & \boxed{-8} & \boxed{-4} & \boxed{0} & \boxed{1} \arrow[r, dash] & \boxed{-1} & \boxed{0}
\end{tikzcd}$$

$$\begin{tikzcd}
 & & A & G & C & G & T \\
 & \boxed{0} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] \arrow[dr, dash] & \boxed{-4} \arrow[r, dash] \arrow[dr, dash] & \boxed{-6} \arrow[r, dash] & \boxed{-8} \arrow[r, dash] & \boxed{-10} \\
C & \boxed{-2} \arrow[d, dash] \arrow[dr, dash] & \boxed{-1} \arrow[r, dash] \arrow[dr, dash] & \boxed{-3} \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{-4} \arrow[r, dash] \arrow[dr, dash] & \boxed{-6} \\
A & \boxed{-4} \arrow[d, dash] & \boxed{0} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] & \boxed{-4} \arrow[dr, dash] & \boxed{-3} \arrow[r, dash] \arrow[dr, dash] & \boxed{-5} \\
G & \boxed{-6} \arrow[d, dash] & \boxed{-2} \arrow[d, dash] & \boxed{2} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{0} \arrow[r, dash] \arrow[dr, dash] & \boxed{-2} \arrow[r, dash] \arrow[dr, dash] & \boxed{-4} \\
T & \boxed{-8} & \boxed{-4} & \boxed{0} & \boxed{1} \arrow[r, dash] & \boxed{-1} & \boxed{0}
\end{tikzcd}$$

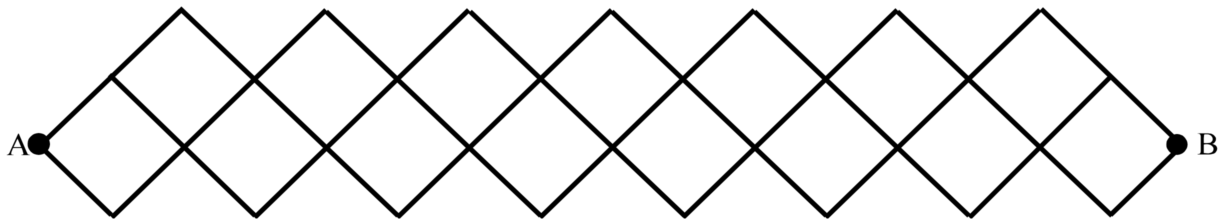
Exercises — Introduction to Dynamic Programming

Quick Concepts

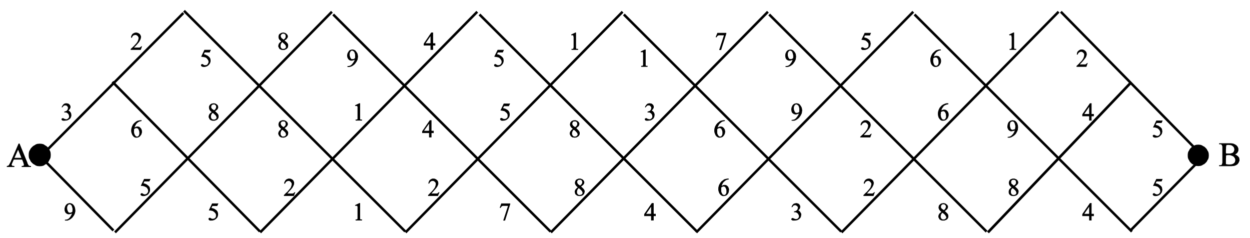
1. How many ways are there to walk from A to B on the grid to the right, without backtracking?



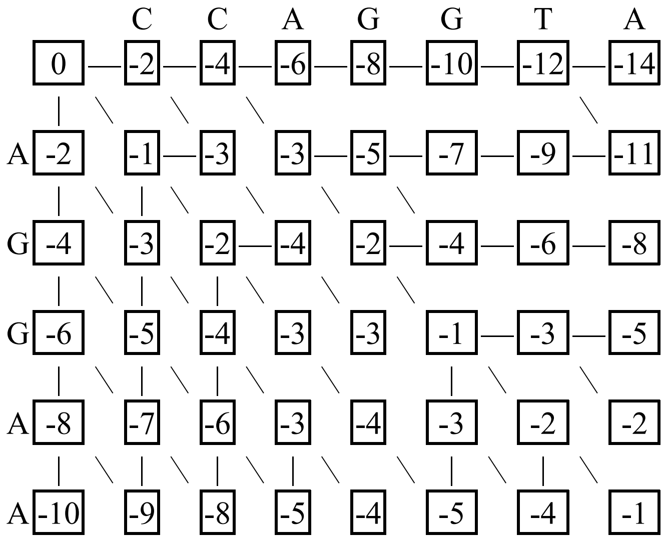
2. Same question on the grid below.



3. The numbers on the edges of the graph below represent distances. What is the length of the shortest path from A to B? How many routes achieve that length? (The take-away lesson is that even though there are *many* paths from A to B, dynamic programming finds the best one rather quickly!)

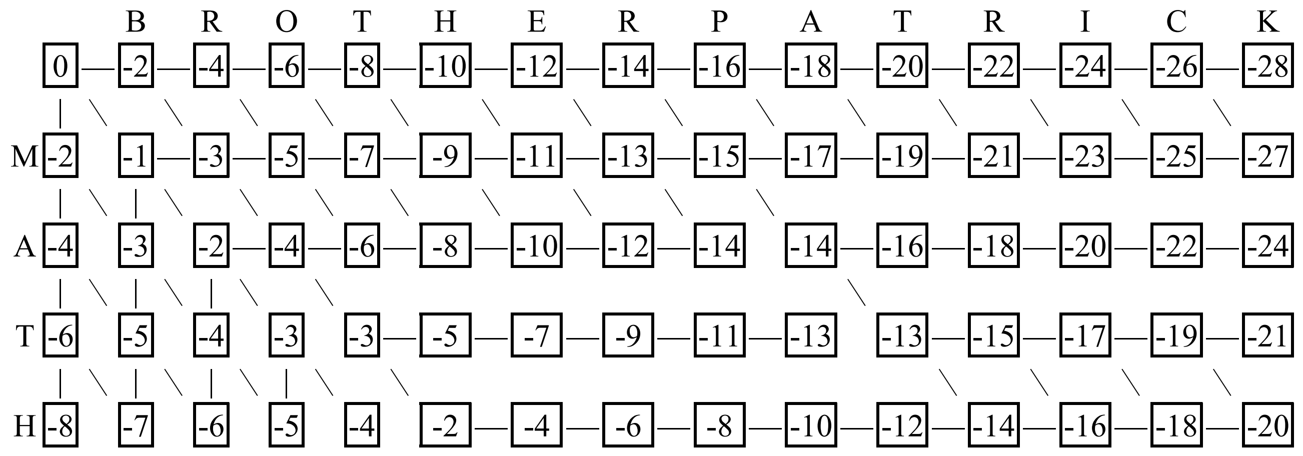


4. What optimal alignment corresponds to the following scoring matrix:


C C A G G T A
0 -2 -4 -6 -8 -10 -12 -14
A -2 -1 -3 -3 -5 -7 -9 -11
G -4 -3 -2 -4 -2 -4 -6 -8
G -6 -5 -4 -3 -3 -1 -3 -5
A -8 -7 -6 -3 -4 -3 -2 -2
A -10 -9 -8 -5 -4 -5 -4 -1

Presentation Problems

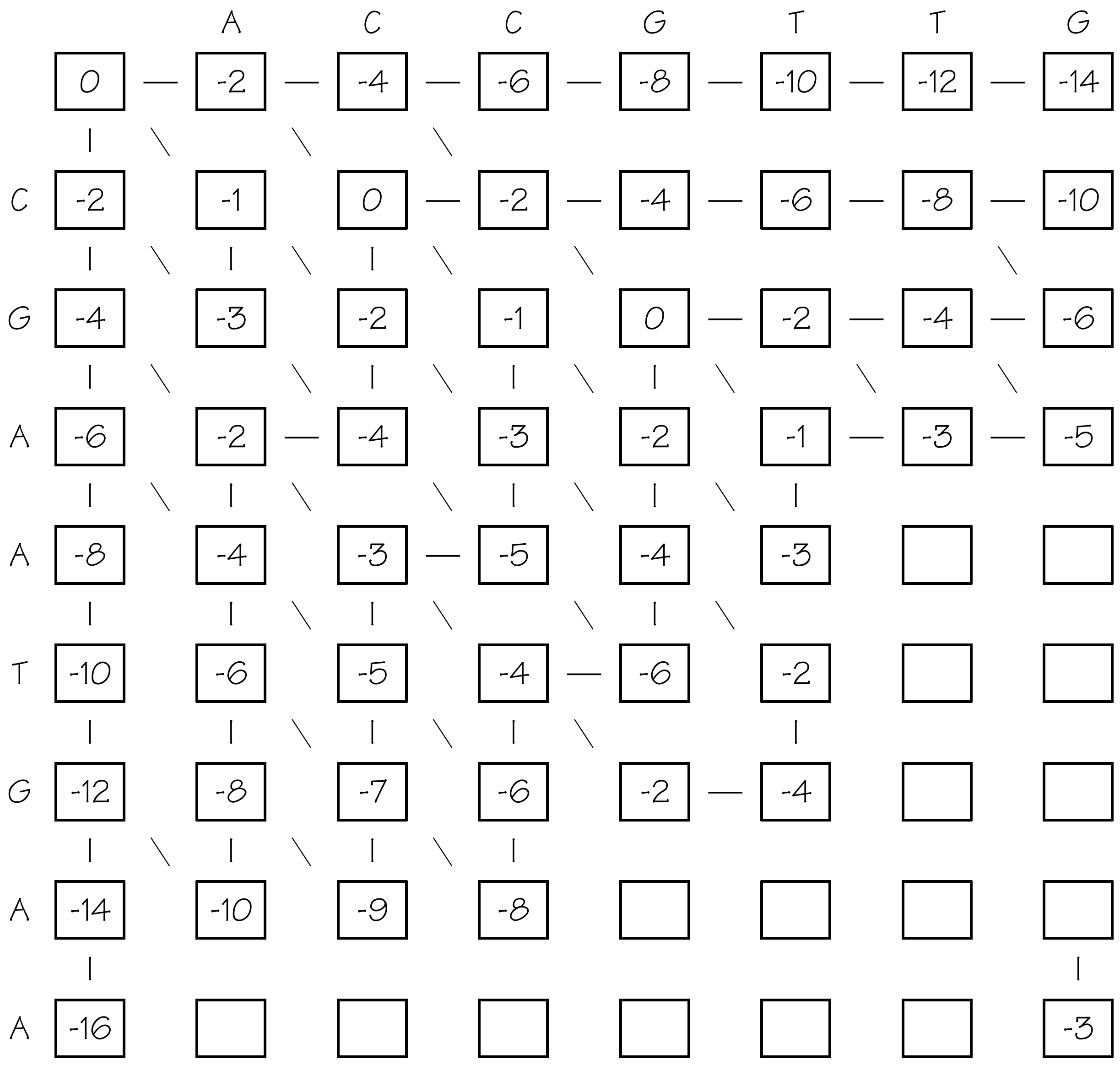
5. How many optimal alignments are indicated by the following scoring matrix?



$$\begin{array}{c|ccccccccccccccc}
 & & B & R & O & T & H & E & R & P & A & T & R & I & C & K \\
\hline
 & 0 & -2 & -4 & -6 & -8 & -10 & -12 & -14 & -16 & -18 & -20 & -22 & -24 & -26 & -28 \\
M & -2 & -1 & -3 & -5 & -7 & -9 & -11 & -13 & -15 & -17 & -19 & -21 & -23 & -25 & -27 \\
A & -4 & -3 & -2 & -4 & -6 & -8 & -10 & -12 & -14 & -14 & -16 & -18 & -20 & -22 & -24 \\
T & -6 & -5 & -4 & -3 & -3 & -5 & -7 & -9 & -11 & -13 & -13 & -15 & -17 & -19 & -21 \\
H & -8 & -7 & -6 & -5 & -4 & -2 & -4 & -6 & -8 & -10 & -12 & -14 & -16 & -18 & -20
\end{array}$$

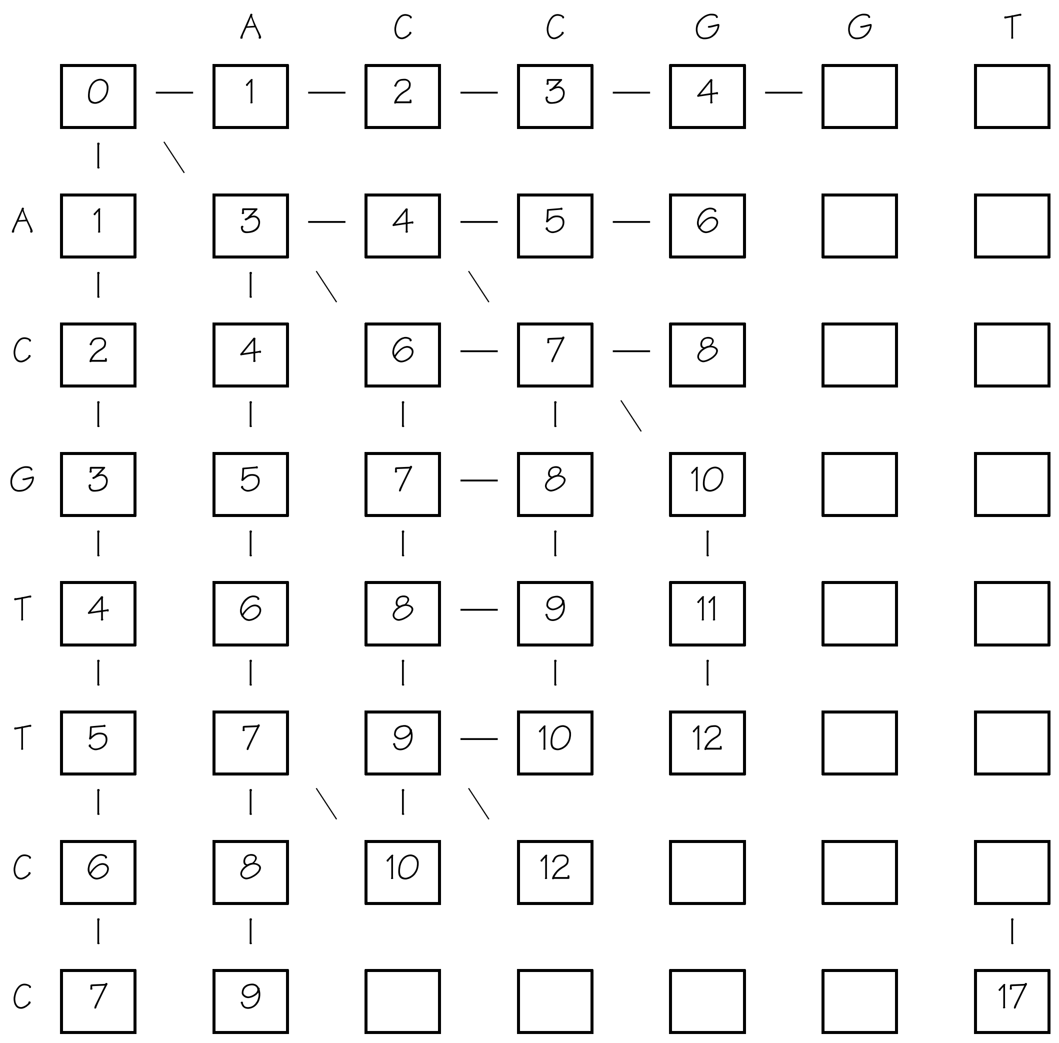
6. Go to http://bsw-uiuc.net/ and log in. This may require you to create a free account, if you don't already have one. You may wish to have Kathy Gabric's notes to BSW handy. Then:
   a. Create a new session called "Homework 1"
   b. Select the "nucleic tools" tab
   c. Type "GRK3" into the edit box in the "Ndjinn" box, select the "GenBank Primate Sequences" database and click "Ndjinn" to search for sequences related to the GRK3 protein.
   d. This should find one sequence with accession number 22477867. Import that sequence into your session.
   e. View the sequence and some information about that sequence. You can do this in the "Nucleic Tools" section (you're probably there after the import above) select the sequence (it'll be at the bottom of the page) and then click the "View Record" tool button.
   f. What are the first 3 letters of the amino acid sequence of this gene's protein product?
   g. What are the 17th-, 16th- and 15th-to-last letters of the amino acid sequence of this gene's protein product?
   h. Go to www.pubmed.com to find out what Dr. Kelsoe has discovered about mutations of the GRK3 gene.
   i. Back in the "Nucleic Tools" section of the Biology Student Workbench, select your GRK3 sequence (at the bottom of the page), and, next to the "BLASTN" tool, select parts 1 and 2 of the GenBank Primate Sequences databases. Then click the "BLASTN" tool to find sequences in other primates' genomes which are similar to the GRK3 gene in humans.
   j. The first few matches you get will just be other entries of exactly this gene, or some portion thereof. What is the first sequence which represents a genuinely different portion of the human genome, or the genome of some other species? And what *is* "*Pan Troglodytes*"?

7. Here is the start of an alignment between "`ACCGTTG`" and "`CGAATGAA`" with match score 2, mismatch penalty -1 and gap penalty -2. Feel free to finish it.



8. Give all optimal alignments between "ACCGTTG" and "CGAATGAA" with match score 2, mismatch penalty -1 and gap penalty -2.

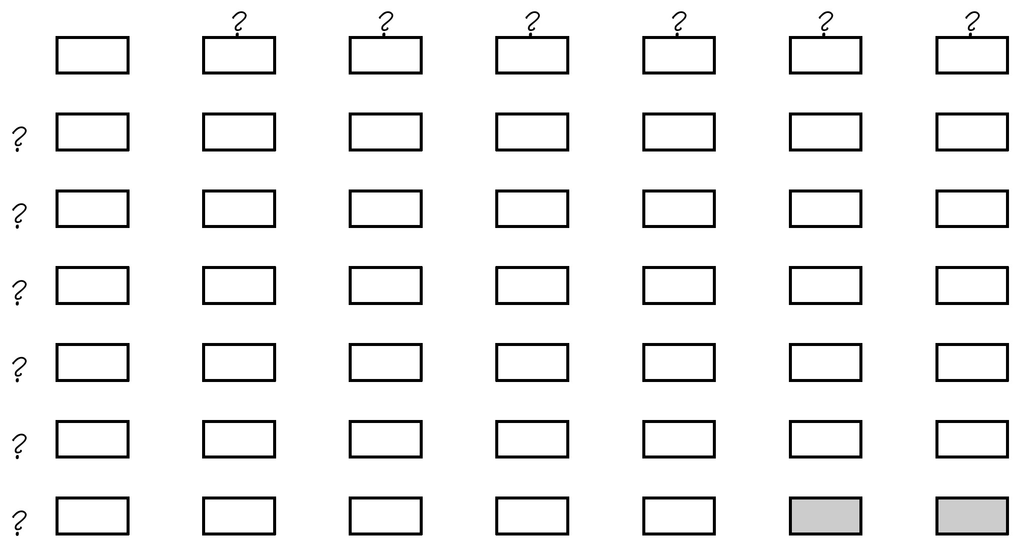
9. Two DNA sequences derived from a common ancestor in an environment in which deletions were much more likely than point mutations. To reflect this in an alignment, a researcher assigns a match score of +3, a mismatch score of -1 and a gap "penalty" of +1. Here is the resulting scoring matrix. You might enjoy finishing it.



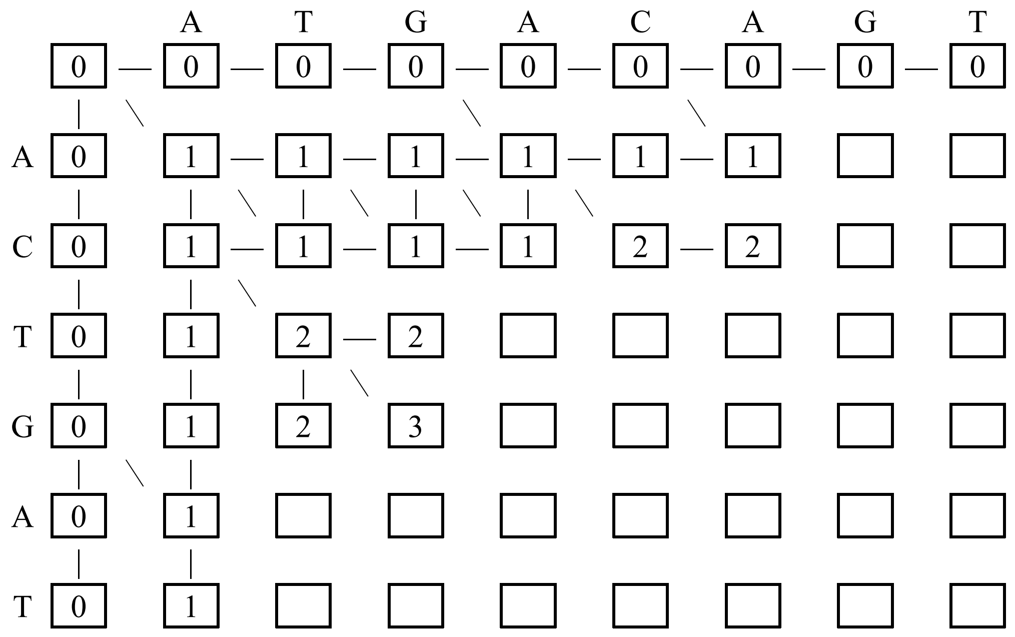
		A	C	C	G	G	T
	0	1	2	3	4		
A	1	3	4	5	6		
C	2	4	6	7	8		
G	3	5	7	8	10		
T	4	6	8	9	11		
T	5	7	9	10	12		
C	6	8	10	12			
C	7	9					17

10. Prove that under the (very artificial) scoring system given in the previous problem, an optimal alignment of any two strings will never align two mismatched bases. In fact, what relationship between the gap penalty and the mismatch penalty will guarantee this behavior?

11. Under an alignment of two nucleotide strings with match score +1, mismatch penalty -1 and gap penalty -2, how large could the difference be between the values in the two boxes shaded below? Could they possibly be the same?



12. In the *Longest Common Subsequence* problem, two strings are given and one wishes to find the length of the longest (not-necessarily contiguous) common subsequence in those strings. For example, the strings ATGACAGT and ACTGAT have "ACGT" as a common substring, but there is a longer one. Devise a dynamic programming solution to the *Longest Common Subsequence* problem for these two strings. To help you get started, I've begun a matrix below.



A T G A C A G T
0 0 0 0 0 0 0 0 0
A 0 1 1 1 1 1 1
C 0 1 1 1 1 2 2
T 0 1 2 2
G 0 1 2 3
A 0 1
T 0 1