

1. Suppose there is a penalty for 1 for each gap and a penalty of 2 for matching two different symbols in a column. What is the NW score of the strings AGTACG and ACATAG?

1. 3
2. 4
3. 5
4. 6

$$\begin{array}{c} A - GTCG \\ ACATA - G \end{array}$$

(2) is the correct answer. We have an upperbound of 8 if we compute the NW score of the original strings. The strings are of the same length so they must have an even NW score because inserting a gap in one of the strings will require we insert a gap in the other. We will have at least one mismatch will we give us a lowerbound of 4.

2. Let  $X = x_1, x_2, \dots, x_m$  and  $Y = y_1, y_2, \dots, y_n$  be two input strings, with each symbol  $x_i$  or  $y_j$  in  $\{A, C, G, T\}$ . How many relevant possibilities are there for the contents of the final column of an optimal alignment?

1. 2
2. 3
3. 4
4.  $m \cdot n$

(2) is the correct answer. We choose for the  $x_m$  character to be a gap or the  $x_m$  character. We again can choose for the  $y_n$  character to be a gap or the  $y_n$  character. Making both  $x_m$  and  $y_n$  gaps would be redundant and we could have a better solution by having either  $x_m$  or  $y_n$  as a gap. Thus we have three options.  $x_m$  and  $y_m$  staying the same.  $x_m$  staying the same and matched with a gap.  $y_n$  staying the same and matched with a gap.

3. Suppose one of the two input strings is empty. What is the NW score of  $X$  and  $Y$ ?

1. 0
2.  $\alpha_{gap} \cdot (\text{length of } X)$

3.  $+\infty$

4. undefined

(2) is the correct answer. We make  $Y$  the “gap” string, which would result in a penalty of the length of  $X$ .

4. Consider the following two search trees that store objects with key 1, 2 and 3: and the search frequency 1:.8, 2:.1, 3:.1. What are the average search times in the two trees, respectively?

1. 1.9 and 1.2

2. 1.9 and 1.3

3. 2 and 1

4. 2 and 3

The average search time for the first tree is  $\sum_{n \in N} p(n) \cdot [\text{depth of } n \text{ in tree} + 1] = .8 * 2 + .1 * 1 + .1 * 3 = 1.9$ . The average search time for the second tree is  $\sum_{n \in N} p(n) \cdot [\text{depth of } n \text{ in tree} + 1] = .8 * 1 + .1 * 2 + .1 * 3 = 1.2$ . Hence (2) is the correct answer.

5. Suppose an optimal binary search tree for the keys  $[1, 2, \dots, n]$  and frequencies  $p_1, p_2, \dots, p_N$  has the key  $r$  as its root, with left subtree  $T_1$  and right subtree  $T_2$ : Of the following four statements, choose the strongest one that you suspect is true.

1. Neither  $T_1$  nor  $T_2$  need be optimal for the keys it contains

2. At least one of  $T_1, T_2$  is optimal for the keys it contains

3. Each of  $T_1, T_2$  is optimal for the keys it contains.

4.  $T_1$  is an optimal binary search tree for the keys  $\{1, 2, \dots, r - 1\}$  and  $T_2$  for the keys  $\{r + 1, r + 2, \dots, n\}$ .

(4) A dynamic programming problem should satisfy the optimal substructure property therefore (1) and (2) are incorrect. (4) is stronger than (3) and is satisfied by the search tree property of a binary search tree.

6. For the sequence alignment input in Quiz 17.1, what are the final array entries of the NW algorithm from section 17.1

7. The *Knapsack* algorithm from section 16.5 and the NW algorithm from section 17.1 both fill in a two-dimensional array using a double for loop. Suppose we reverse the order of the for loop—literally cutting and pasting the second loop in front of the first, without changing the pseudocode in any other way. Are the resulting algorithms well defined and correct?

1. Neither algorithm remains well-defined and correct after reversing the order of the for loops
2. Both algorithms remain well defined and correct after reversing the order of the for loop.
3. The Knapsack algorithm remains well-defined and correct after reversing the order of the for loop, but the NW algorithm does not.
4. The NW algorithm remains well-defined and correct after reversing the order of the for loops, but the Knapsack algorithm does not.

8. The following problems all take as input two strings  $X$  and  $Y$ , with lengths  $m$  and  $n$ , over some alphabet  $\Sigma$ . Which of them can be solved in  $O(m \cdot n)$  time?

1. Consider the variation of sequence alignment in which, instead of single gap-penalty  $\alpha_{gap}$ , you are given two positive numbers  $a$  and  $b$ . The penalty for inserting  $k$  gaps in a row is now defined as  $ak + b$  rather than  $k \cdot \alpha_{gap}$ . The other penalties are defined as before. The goal is to compute the minimum-possible penalty of an alignment under this new cost model.
2. Compute the length of a longest common subsequence of  $X$  and  $Y$ .
3. Assumes that  $X$  and  $Y$  have the same length  $n$ . Determine whether there exists a permutation  $f$ , mapping each  $i \in \{1, 2, \dots, n\}$  to a distinct value  $f(i) \in \{1, 2, 3, \dots, n\}$  such that  $X_i = y_{f(i)}$  for every  $i = 1, 2, \dots, n$ .
4. Compute the length of a longest common substring of  $X$  and  $Y$ .