# Fill-in form questions

If this were the real quiz, you would fill in your answers to this on a Google Form, similar to many questions on previous quizzes.

## Longest Common Subsequence

Suppose we are computing the LCS of two strings, using the algorithm from class. We find the following table, although I have omitted the letters from the word that would be printed across the top.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 |
| B | 0 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 |
| L | 0 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 |
| I | 0 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 |
| S | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| H | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| M | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| E | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| N | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 5 |
| T | 0 | 1 | 2 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 |

1. FYI, the word that would be across the top is "ABSTRACTION"

   What is the first letter of "ESTABLISHMENT" that appears anywhere in the word it is being compared to? For example, if 'E' appears in the word that would be printed across the top, write 'E.'

   'S', the second letter

2. What is the last letter of the word that would appear across the top?

   N, although it could also be T.

## Edit Distance

For the next few questions, suppose I am computing edit distance using the algorithm from lecture. However, this time I have omitted both strings from the chart and have instead written the string via indices.

FYI, it was "UNDERSTANDING" and "SUBTRACTION"

Recall that the iterative code is as follows. EditDistance($X_{1\ldots n}$, $Y_{1\ldots m}$)

**for** $j \leftarrow 0 \ldots m$ **do**
  Edit$[0, j] \leftarrow j$
**for** $i \leftarrow 1 \ldots n$ **do**
  Edit$[i, 0] \leftarrow i$
  **for** $j \leftarrow 1 \ldots m$ **do**
    ins $\leftarrow 1+$ Edit$[i, j-1]$
    del $\leftarrow 1+$ Edit$[i-1, j]$
    sub $\leftarrow$ Edit$[i-1, j-1] + (X_i \neq Y_j)$
    Edit$[i, j] \leftarrow$ min(ins, del, sub)

Recall that, despite how that code is written, if $X[i] == Y[j]$, we do not consider that to be a "substitution" despite the variable that would be chosen. We call that situation a "match."

| | | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ | $X_9$ | $X_{10}$ | $X_{11}$ | $X_{12}$ | $X_{13}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| $Y_1$ | 1 | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $Y_2$ | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $Y_3$ | 3 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 8 | 9 | 10 | 11 | 12 |
| $Y_4$ | 4 | 3 | 3 | 3 | 4 | 5 | 6 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $Y_5$ | 5 | 4 | 4 | 4 | 4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $Y_6$ | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 7 | 8 | 9 | 10 | 11 |
| $Y_7$ | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 7 | 7 | 8 | 9 | 10 | 11 |
| $Y_8$ | 8 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 7 | 8 | 8 | 9 | 10 | 11 |
| $Y_9$ | 9 | 8 | 8 | 8 | 8 | 8 | 8 | 7 | 7 | 8 | 9 | 8 | 9 | 10 |
| $Y_{10}$ | 10 | 9 | 9 | 9 | 9 | 9 | 9 | 8 | 8 | 8 | 9 | 9 | 9 | 10 |
| $Y_{11}$ | 11 | 10 | 9 | 10 | 10 | 10 | 10 | 9 | 9 | 8 | 9 | 10 | 9 | 10 |

3. $X_1 == Y_1$? no
4. $X_1 == Y_2$? yes
5. $X_{13} == Y_{11}$? no
6. $X_{12} == Y_{11}$? yes
7. When computing Edit Distance between $X_{1\ldots9}$ and $Y_{1\ldots7}$, was the chosen operation a substitution, a match, or an insertion/deletion?

   This was a substitution.

## Optimal Binary Search Trees

8. Consider the following table, which is the output from running the optimal binary search tree algorithm for some nine-key input. In each entry, the value printed in the upper half of the cell is OPT$[i, j]$, the cost of the optimal binary search tree consisting of keys $i \ldots j$. The value printed in the lower half of the cell is the value of roots$[i, j]$.

   Suppose we have computed the following using the algorithm for the optimal binary search tree. However, we accidentally forgot to record OPT$[1, n]$. What is the missing value for that spot? It is the only one omitted in the following table.

   2.29 – we can find this because we know the sum of probabilities is 1, and the root is 4. So we need OPT[1,3] = 0.8 and OPT[5,7] = .49.

   $0.8 + 0.49 + 1 = 2.29$.

| | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $k_6$ | $k_7$ |
|---|---|---|---|---|---|---|---|
| $k_1$ | 0.17<br>1 | 0.41<br>1 | 0.8<br>2 | 1.29<br>3 | 1.87<br>4 | 2.17<br>4 | <br>4 |
| $k_2$ | | 0.12<br>2 | 0.41<br>3 | 0.83<br>3 | 1.31<br>4 | 1.61<br>4 | 1.73<br>4 |
| $k_3$ | | | 0.17<br>3 | 0.55<br>4 | 0.95<br>4 | 1.25<br>4 | 1.37<br>4 |
| $k_4$ | | | | 0.21<br>4 | 0.61<br>4 | 0.82<br>5 | 0.91<br>5 |
| $k_5$ | | | | | 0.2<br>5 | 0.4<br>5 | 0.49<br>5 |
| $k_6$ | | | | | | 0.1<br>6 | 0.16<br>6 |
| $k_7$ | | | | | | | 0.03<br>7 |

9. Consider the following output for the Optimal BST algorithm, but with a different input from the previous problem. Unfortunately this time, I forgot to write down the root values! What is the root of the overall tree? That is, what should be in roots$[1, n]$?

| | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ |
|---|---|---|---|---|---|
| $k_1$ | 0.43 | 0.61 | 1.06 | 1.51 | 2.06 |
| $k_2$ | | 0.09 | 0.36 | 0.66 | 1.08 |
| $k_3$ | | | 0.18 | 0.48 | 0.81 |
| $k_4$ | | | | 0.15 | 0.45 |
| $k_5$ | | | | | 0.15 |

   It's 3, which we can get by apply the recurrence and testing different valid values for the root.

## Free Response (5 points)

On the real quiz, you would solve this problem, scan your solution (if it isn't already digital), upload your solution to GradeScope, and tag the question.

As a new professor, Shindler needs to set up his lab. However, nothing is free, and Shindler has to pay quarterly rent for the building where he keeps his office. Shindler plans to be at UCI for $n$ quarters (for a hopefully large value of $n$). Each quarter, he has the option to keep his lab in Donald Bren Hall or in Engineering Hall. If he keeps his lab in Donald Bren Hall during quarter $i$, he has to pay $\$d_i$. If he keeps his lab in Engineering Hall that quarter, he has to pay $\$e_i$. If he is in one building during quarter $i$ and the other during quarter $i+1$, he has to pay $\$M$ to move offices.

For example, suppose $n = 4$ and $M = 10$, with the following costs:

| | Quarter 1 | Quarter 2 | Quarter 3 | Quarter 4 |
|---|---|---|---|---|
| Bren | 1 | 3 | 20 | 20 |
| Engineering | 50 | 20 | 2 | 4 |

In this case, the optimal solution would be to stay in Bren Hall for the first two quarters and then move to Engineering Hall for the next two quarters.

Give a dynamic programming solution to this problem; (1) write out the base case and recurrence expressions; (2) give an English specification of the underlying recursive sub-problems and (3) analyze the running time for the iterative version, explaining any important implementation details. You do not need to actually write out the iterative version.

*Hint: You will either have two* $1 \times n$ *tables or one* $2 \times n$ *table.*

**If your entire response to this question is "I do not know," you will get 2 points out of 5 instead. Note that it is possible to earn fewer than 2 points for a bad response to this question. The same option will exist on the real quiz.**

Define OPT$[1, i]$ to be the cost of the optimal solution through the first $i$ quarters, with quarter $i$ being spent in Bren Hall. OPT$[2, i]$ is defined similarly, but for Engineering Hall.

What's the optimal way to end quarter $i$ in Bren Hall? If it's quarter 1, then the cost is $B_1$. Otherwise, it's $B_i$ plus we have a decision to make: did we spend the previous quarter in Bren Hall or in Engineering Hall?

If we spent the previous quarter in Bren Hall, then the optimal cost there is OPT$[1, i-1]$ for the previous quarters plus $d_i$ for this one.. If it was in Engineering Hall, then we spent OPT$[2, i-1]$, plus $d_i$ for this month's rent, plus $M$ for moving.

Thus OPT$[1, i] = \min($OPT$[1, i-1] + d_i$, OPT$[2, i-1] + d_i + M)$

OPT$[2, i]$ (the optimal solution for Engineering Hall) is symmetric.

There are $\mathcal{O}(n)$ cases, each of which take $\mathcal{O}(1)$ to fill in, for a total time of $\mathcal{O}(n)$.