CS 344
Fall 2020

**Problem Set 4**
**Due:** Dec 14, 2020, 11:59 p.m.

---

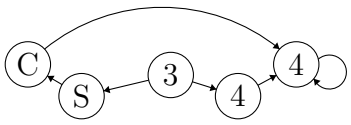
**Name**: (Yuting Chen) **NetID**: (yc1071)

**Honor Code**: Students may discuss and work on homework problems in groups, which is encouraged. However, each student must write down their solutions independently to show they understand the solution well enough to reconstruct it by themselves. Students should clearly mention the names of the other students who offered discussions. We check all submissions for plagiarism. We take the honor code seriously and expect students to do the same.

**Instruction for Submission**: This homework has a total of 40 points + 5 bonus points, it will be rescaled to 10 points + 1.25 bonus points as the eventual score. We have provided the homework4.tex file for you, please write your answer to each question in this latex file directly after the corresponding question, and submit the complied PDF file only. You should name your PDF file as "Firstname-Lastname-NetID.pdf".

**Late Policy**: The homework is due on 12/14 (Monday) at 11:59pm. We will release the solutions of the homework on Canvas on 12/18 (Friday) 11:59pm. If your homework is submitted to Canvas before 12/14 11:59pm, there will no late penalty. If you submit to Canvas after 12/14 11:59pm and before 12/18 11:59pm (i.e., before we release the solution), your score will be penalized by $0.9^k$, where $k$ is the number of days of late submission. For example, if you submitted on 12/17, and your original score is 80, then your final score will be $80 * 0.9^3 = 58.32$ for $17 - 14 = 3$ days of late submission. If you submit to Canvas after 12/18 11:59pm (i.e., after we release the solution), then you will earn no score for the homework.

**Drawing graphs:** You might try `http://madebyevan.com/fsm/` which allows you to draw graphs with your mouse and convert it into LaTeX code:




You can also draw by hand and insert a picture.

**Make best use of picture in Latex**: If you think some part of your answer is too difficult to type using Latex, you may write that part on paper and scan it as a picture, and then insert that part into Latex as a picture, and finally Latex will compile the picture into the final PDF output. This will make things easier. For instructions of how to insert pictures in Latex, you may refer to the Latex file of our homework 1, which includes several examples of inserting pictures in Latex.

---

Discussion Group (People with whom you discussed ideas used in your answers if any):

I acknowledge and accept the Honor Code. Please type your initials below:
**Signed**: (YC)

---

1. **Longest Common Subsequence** (20 points)

   Consider the two sequences `penguin` and `chicken`, using the LCS algorithm we learnt in class, find the longest common subsequence of these two words.

   (a) (10 points) Work out the 2-dimensional dynamic programming table $T(i, j)$ and provide the length of the LCS.

   **[You are expected to work out every element of the dynamic programming table.]**

(a)

| | | p 1 | e 2 | n 3 | g 4 | u 5 | i 6 | n 7 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| h | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| i | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| c | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| k | 5 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| e | 6 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| n | 7 | 0 | 0 | 1 | 2 | 2 | 2 | 2 |

Figure 1: Question 1(a).

Answer: The length of the LCS is 2.

(b) (10 points) Traceback on the table to find out the exact longest common subsequence(s) of the two words.

**[You are required to draw the traceback path(s) on the table and find out all the LCSs.]**

Answer: There are two paths. The longest common subsequence of blue path is *in*. The longest common subsequence of pink path is *en*.

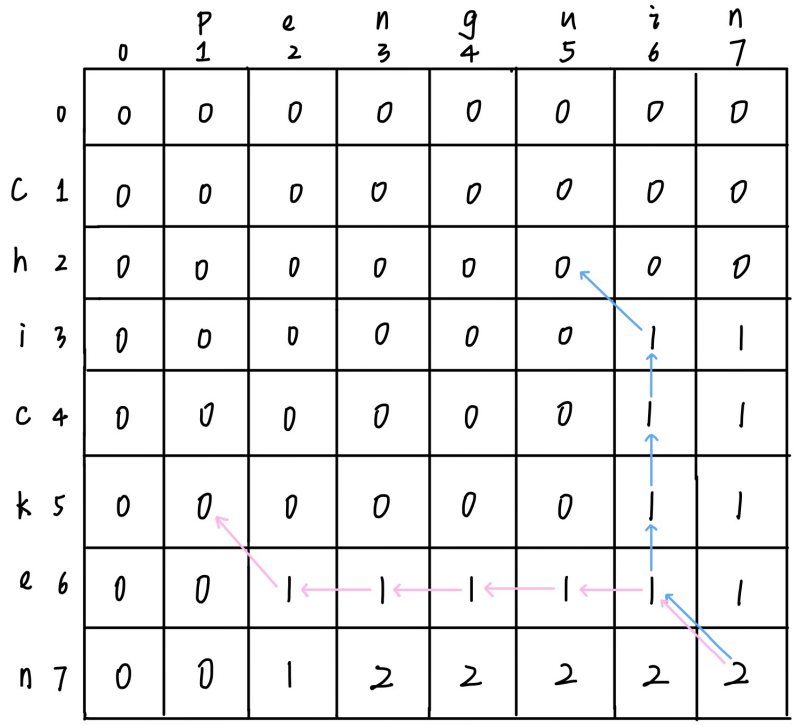



Figure 2: Question 1(b).

2. **Longest Increasing Subsequence** (20 points + 5 bonus points)

   Let $A$ be an array of length $n$ containing real numbers. A *longest increasing subsequence* (LIS) of $A$ is a sequence $0 \leq i_1 < i_2 < \ldots i_\ell < n$ so that $A[i_1] < A[i_2] < \cdots < A[i_\ell]$, so that $\ell$ is as long as possible. For example, if $A = [6, 3, 2, 5, 6, 4, 8]$, then an LIS is $i_0 = 2, i_1 = 3, i_2 = 4, i_3 = 6$ corresponding to the subsequence $2, 5, 6, 8$. (Notice that a longest increasing subsequence does not need to be unique). In the following parts, we will walk through the recipe that we saw in class for coming up with DP algorithms to develop an $O(n^2)$-time algorithm for finding an LIS.

   (a) (10 points) **(Identify optimal sub-structure and a recursive relationship).** We will come up with the sub-problems and recursive relationship for you, although you will have to justify it. Let $D[i]$ be the length of the longest increasing subsequence of $[A[0], \ldots, A[i]]$ that ends on $A[i]$. Expain why

$$D[i] = \max_k \{D[k] + 1 \ : \ 0 \leq k < i, A[k] < A[i]\}.$$

   [**Provide a short informal explanation. It is good practice to write a formal proof, but this is not required for credit.**]
   Answer:Given$D[i]$ is the length of the longest increasing subsequence of $[A[0], \ldots, A[i]]$ that ends on $A[i]$.We can express that the length of the longest increasing subsequence ending at any index "i" will be 1 more than max, that is, all the lengths of

the longest increasing subsequence ending before the ith index, and where, for all k ¡ i, A [k] ¡A [i]. We divide this array into many sub-array.first, we can ignore A[i], and see the first i-1elenments, then we find LIS of first (i-1) elenments. final, we add 1 into the D[i].

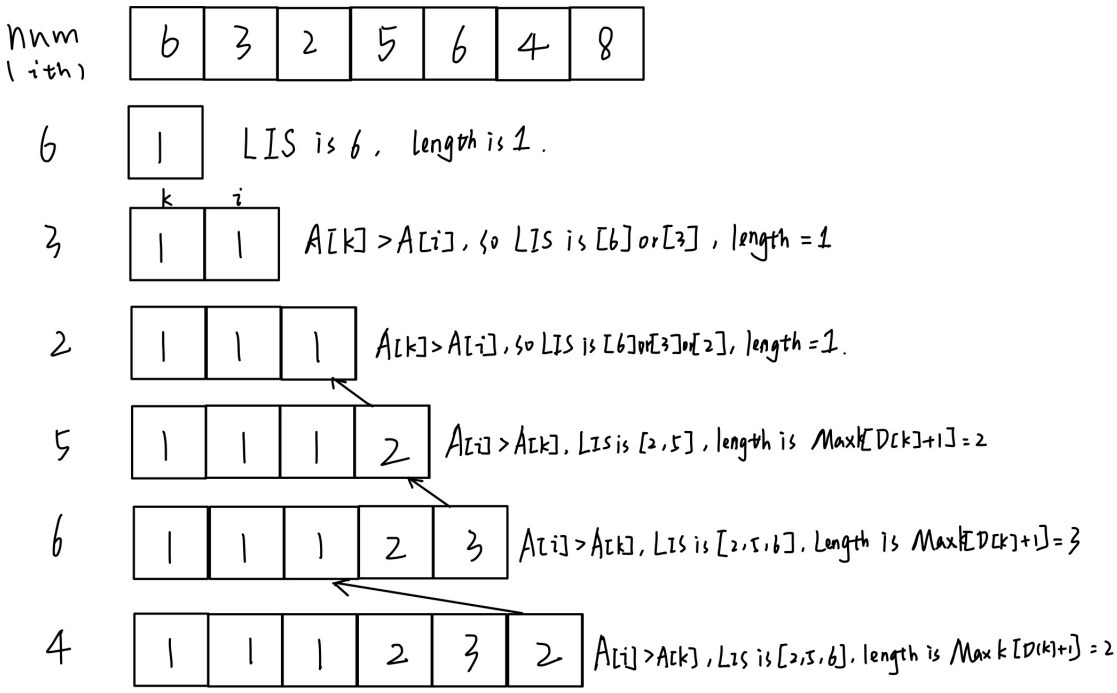



Figure 3: Question 2(a).

Therefore, the max of k is 3, then 3+1 = 4, the length of LIS of A[i] is 4.

(b) (10 points) **(Develop a DP algorithm to find the value of the optimal solution)** Use the relationship above to design a dynamic programming algorithm that returns the *length* of the longest increasing subsequence. Your algorithm should run in time $O(n^2)$ and should fill in the array $D$ defined above.

[**Provide the pseudo-code, and a brief justification of why the complexity is $O(n^2)$, no formal proof is required.**]

Answer:

function getLIS(int[] A, int n)
initial variable array > int D[size n]
set $D[0] < -1$
FOR $(i = 1, i <= n, i++)$
set $D[i] < -1$
FOR $(k = 0, k <= i, k++)$
IF $(A[i] >= A[k]$ and $D[i] <= D[k] + 1)$
set $D[i] < -D[k] + 1$

END IF
END FOR
END FOR
initial variable lengthLIS $< -1$
FOR$(i = 1, i <= n, i++)$
IF $(D[i] >= lengthLIS)$
set $lengthLIS < -D[i]$
END IF
END FOR
return lengthLIS

For this algorithm, we have to consider the outer loop and inner loop. The inner loop k moves from 0 to i, there are n times for each iteration of the outer loop, so for each iteration of the outer loop, we must move at most n previous elements and in the inner for loop. Therefore, the complexity is $O(n^2)$.

(c) (5 bonus points) **(Adapt the DP algorithm to return the optimal solution)** Adapt your algorithm above by tracking some useful information during the DP procedure, so that it returns the actual LIS, not just its length.

[**Pseudocode and a short explanation.**]

**Note:** Actually, there is an $O(n \log n)$-time algorithm to find an LIS, which is faster than the DP solution in this exercise.

Answer:
Input: An array A
Output: The Longest Increasing Sub-sequence of A
Function longestIncreasingSubsequence(A)
$a = 0$
FOR $(i = 1ton - 1)$:
IF $(A[i] > A[I[a]])$
$a = a + 1$
$I[a] = i$
$P[a - 1] = I[a - 1]$
ENDIF
ELSE
do binary search in $I[1...a]$ for j such that: $A[I[j]]$ is the smallest element in all $A[I[j]]$,
$j = 0, 1, ...length(I[\,])andA[I[j]] > A[i]$
$I[j] = i$
END ELSE
if $j == a$
$P[b] = I[a - 1]$
ENDIF
ENDELSE
ENDFOR
$LIS[a] = A[I[a]]$

FOR $(i = a - 1, i <= 1, i - -)$
$LIS[i] = A[P[b]]$
ENDFOR
return LIS

when the new element is not larger, we find the smallest element in the LIS sequence that is larger than our value and replace it with our current value. We could use binary search to find this element since the index of the value by I[ ] are in ascending order. This runs in O(log(n)) time.