CSE 100: Algorithm Design and Analysis: Midterm 3

Fall 2022

- This is a take-home exam with no proctoring. There are **6** questions in total covering Lecture slides 14-18. You can earn some partial points if you show progress even if you can't solve problems completely. You can access this exam from **12:00am on Nov. 21st to 11:59pm on Nov. 22nd**. Please submit your solution by **11:59pm, Nov. 22nd**, through CatCourses (Midterm 3 under Assignments). You can resubmit any number of times until the deadline. If you failed uploading by 11:59pm, email them to your TA, but it will incur 1 pt penalty for you for every minute delay(e.g. If we receive your solutions at 0:03am, you will lose 4pts).
- This is an open-book exam. The **only** resources you can use during the exam are all **course materials** uploaded to CatCourses plus the **textbook**. In particular, you are NOT allowed to search for solutions on the Internet. You must take the exam by yourself. You may be asked to explain your answers to the instructor or TA later. If you cannot provide a clear explanation of your answers, it violates the UC Merced Academic Honesty Policy. This means you'll get 0 grades, and further be reported to the Office of Student Rights and Responsibilities.
- Title your work as CSE 100 Midterm 3, Fall 2022 along with your full name. **Please copy and sign the following honor pledge at the beginning** of your solution sheet(s) right below the title: *By completing this exam, I acknowledge and confirm that I will not give or receive any unauthorized assistance on this examination. I will conduct myself within the guidelines of the university academic integrity.*
- **Write your solution by hand on your solution sheet(s).** Clearly and orderly write down the question numbers. When submitting, scan or take a picture of each sheet and upload them to Catcourse. You are responsible for the clarity and completeness of your uploads to get scores.
- If you have questions, you can email the instructor and the TAs. However, we may not answer your questions on time as we can't be available all the time.

Question	Points earned	Out of
1		14
2		19
3		18
4		17
5		12
6		20
Sum		100

1. (14 points) For each of the following claims, determine if it is true or false. *No* explanation is needed.

   (a) (2 points) In the tree representation corresponding to a Huffman code, there are at least two leaves of the highest depth.
   Sol. TRUE

   (b) (2 points) Let G be an undirected, weighted graph. Multiplying every edge weight by 2 does not change the shortest paths in G.
   Sol. TRUE

   (c) (2 points) A fixed-length code is prefix-free.
   Sol. TRUE

   (d) (2 points) Suppose in some graph G, the closest negative cycle to a node v has all its nodes at least 5 hops away from v. Consider any node x, whose shortest path from v has fewer than 5 hops. Bellman-Ford computes $d(v; x)$ correctly.
   Sol. TRUE

   (e) (2 points) If we are only interested in computing the length of LCS of two sequences of lengths m and n, we can compute it in $O(mn)$ time using $O(m + n)$ space.
   Sol. TRUE

   (f) (2 points) Suppose G has $\Theta(n)$ edges with integer weights on the range $O(n^3)$. The runtime of $Kruskal's$ and $Prim's$ algorithm will be asymptotically equivalent.
   Sol. FALSE

   (g) (2 points) The Huffman code algorithm can be implemented using priority queue. If we use binary heap to implement a min-priority queue, the Huffman code algorithm can be implemented in $O(n \log n)$ time.
   Sol. TRUE

2. (19 points) In the LCS problem, we are given as input two sequences, $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ and would like to find a longest subsequence common to both. Towards this end, we defined $c[i, j]$ to be the length of LCS of X_i and Y_j, where $X_i := \langle x_1, x_2, ...x_i \rangle$ and $Y_j := \langle y_1, y_2, ...y_j \rangle$.

   (a) (5 points) Give a recursion for computing $c[i, j]$. Do not forget the base cases.
   Sol.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1][j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_i \\ \max(c[i, j-1], c[i-1, j]) & \text{if } i, j > 0 \text{ and } x_i \neq y_i \end{cases}$$

   This is discussion CH 15 problem 7, and also in the slides, Notes and book!

   (b) (14 points) Fill out the the following empty LCS DP table of entries $c[i, j]$.

	j	0	1	2	3	4
i		y_j	A	B	B	A
0	x_i					
1	A					
2	C					
3	B					
4	A					
5	B					

   Sol.

	j	0	1	2	3	4
i		y_j	A	B	B	A
0	x_i	0	0	0	0	0
1	A	0	1	1	1	1
2	C	0	1	1	1	1
3	B	0	1	2	2	2
4	A	0	1	2	2	3
5	B	0	1	2	3	3

   This is a simpler problem than Discussion Ch 15 problem 6.

3. (18 points) Huffman Code. Suppose we have a text consisting only of a, b, c, d, e, f where each character appears with the following frequency:

a	b	c	d	e	f
5	6	9	2	4	1

(a) (12 points) Show the code built by the Huffman algorithm, *both as a tree and as a list* (character, codeword). When combining two trees, *the tree with lowest root frequency becomes the left child* and the tree with the second-lowest root frequency becomes the right child. Left children are associated with the bit 0, right children with the bit 1.

Sol.

(b) (3 points) How many bits are required to represent the input using this Huffman code?
Sol.
5*2 + 6*2 + 9*2 + 2*4 + 4*3 + 1*4 = 64

(c) (3 points) How many bits are required to represent the input using a fixed-length code? Of course, you want to use as few bits as possible.
Sol.
3 * 27 = 81
This is very similar to Discussion Ch 16 problem 3.

4. (17 points) Dijkstra shortest path algorithm. Do the following:

   (a) (7 points) Write the pseudocode of the Dijkstra algorithm.

Sol.

Algorithm 1: DIJKSTRA(D, w, s)

```
for each vertex v ∈ G, V do
    v.d = ∞;
    v.π = NIL;
s.d = 0;
S = 0;
Q = G.V;
while Q ≠ 0 do
    u = EXTRACT-MIN(Q);
    S = S ∪ {u};
    for each vertex v ∈ G.Adj[u] do
        if v.d > u.d + w(u, v) then
            v.d = u.d + w(u, v);
            v.π = u;
```

Note to TAs:
The pseudocode has to respect the spirit of the above, the notation may slightly change depending on the variables used. Also, it is fine to do something similar (with more words) as I did in the Lecture slides. See Lecture 14 Weighted Graphs for more details!

This is covered in the Lecture 10 slides and notes and in the CLRS book!

   (b) (10 points) For the following graph, assuming you start from node s, calculate the shortest path distances between s and the rest of the vertices. Assume that the array $d[y]$ has the shortest distance values for every single vertice destination in alphabetical order, i.e., t, x, y, z. Show the intermediate values of the array at *each* iteration of the loop in the algorithm.

Sol. The vector $d(v)$ will have the following values as we run the algorithm:

	s	t	x	y	z
$d^{(0)}$	0	∞	∞	∞	∞
$d^{(1)}$	0	10	∞	5	∞
$d^{(2)}$	0	8	14	5	7
$d^{(3)}$	0	8	13	5	7
$d^{(4)}$	0	8	9	5	7

This is EXACTLY 'AS IS' Discussion Ch 24 problem 5.

5. (12 points). Consider the following weighted undirected graph.

For each of the following two algorithms, show the minimum spanning tree and list the edges appearing in the MST *in the order they are added*. For example, the first edge added by Kruskal's algorithm is (B, C). Also provide the weight of the MST found.

(a) (6 points) Kruskal's algorithm.
Sol. BC, BD, EF, DF, AB. Weight of MST is 17.

(b) (6 points) Prim's algorithm using A as initial vertex.
Sol. AB, BC, BD, DF, EF. Weight of MST is 17.

6. (20 points) In the rod cutting problem, we are given a rod of length n, along with p_i, which indicates the price of a rod/piece of length p_i, for i = 0, 1, 2, ..., n. Consider the following instance: $p_0 = 0, p_1 = 2, p_2 = 7, p_3 = 15, p_4 = 16$ and $n = 4$. Let r_i denote the maximum profit you can obtain by cutting a rod of length i into smaller pieces (or by making no cuts). What is the respective value of $r_0, r_1, ..., r_4$? You *must* show how you obtained the values by applying the recurrence.

   Sol. $r_0 = 0$
   $r_1 = p_1 = 2$
   r_2 :
   max of $p_1 + r_1 = 4$ and $p_2 + r_0 = 7$.
   so $r_2 = 7$.
   r_3 :
   max of $p_1 + r_2 = 9$, $p_2 + r_1 = 9$and $p_3 + r_0 = 15$
   so $r_3 = 15$.
   r_4 :
   max of $p_1 + r_3 = 17$, $p_2 + r_2 = 14$, $p_3 + r_1 = 17$ and $p_4 + r_0 = 16$
   so $r_4 = 17$.