CSE 100: Algorithm Design and Analysis
Final Exam

Spring 2023

This is a **closed** book examination. You have 180 minutes to answer as many questions as possible. The number in the square bracket at the beginning of each question indicates the number of points given to the question. Write all of your answers directly on this paper. Please make your answers concise as well as precise. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer. Or you can ask the TAs proctoring the exam for clarification. There is no penalty for attempting to answer a question with a wrong answer, so please answer as many questions as you can. Partial credit will be given even if the answer is not fully correct. If you run out of space you can ask for extra papers. Please write down your name on *every* page as some papers may fall off although it rarely happens. 14 pages in total including this cover.

Problem	Points earned	Out of
1		16
2		2
3		3
4		3
5		3
6		2
7		2
8		2
9		3
10		6
11		4
12		4
13		6
14		6
15		5
16		5
17		5
18		4
19		4
20		2
21		4
22		4
23		5
Sum		Max 100

Name ________________________

1. (16 points). If the statement is correct, choose 'true,' otherwise 'false.'

   (a) (1 point) Given two sorted arrays $A[1 \ldots n]$ and $B[1 \ldots n]$, we can merge them into one sorted array in $O(n)$ time. **Sol.** True
   True **False**

   (b) (1 point) The counting sort is a stable sorting algorithm. **Sol.** True
   True **False**

   (c) (1 point) Consider the decision tree of the Merge sort algorithm on n elements. Then, every path from the root to a leaf node in the tree has at most $O(n \log n)$ edges. **Sol.** True
   True **False**

   (d) (1 point) There exists a linear-time deterministic algorithm for the Selection problem. **Sol.** True
   True **False**

   (e) (1 point) We can build a max heap in $O(\log n)$ time. **Sol.** False
   True **False**

   (f) (1 point) Any comparison based sorting algorithm must make $\Omega(n \log n)$ comparisons for some input of n elements. **Sol.** True
   True **False**

   (g) (1 point) The expected running time of randomized selection is $\Theta(n \log n)$. **Sol.** False
   True **False**

   (h) (1 point) If we solve $T(n) = 2T(n/2) + n^2$, then we obtain $T(n) = \Theta(n^2)$. **Sol.** True
   True **False**

   (i) (1 point) There is a unique path on a tree $T = (V, E)$ between any pair of vertices, $u \neq v \in V$. **Sol.** True
   True **False**

   (j) (1 point) Given a DAG $G = (V, E)$ with weighted edges, we can compute the shortest distance from $s \in V$ to *each* vertex in time $O(V + E)$. **Sol.** True
   True **False**

   (k) (1 point) If there is a path from s to t that includes a cycle, there exists no shortest path from s to t. **Sol.** False
   True **False**

   (l) (1 point) An undirected graph $G = (V, E)$ with $|V| = |E|$ must have a cycle. **Sol.** True
   True **False**

   (m) (1 point) The run time of Dijkstra's algorithm is $O(V + E)$. **Sol.** False
   True **False**

   (n) (1 point) If all edges have distinct weights, there is a unique MST. **Sol.** True
   True **False**

   (o) (1 point) If $f = O(g)$, then we have $g = O(f)$. **Sol.** False
   True **False**

   (p) (1 point) If $f = O(g)$ and $f = \Omega(g)$, then we have$f = \Theta(g)$. **Sol.** True
   True **False**

2. (2 points) Briefly explain the Random Access Model (RAM). **Sol.** Single processor (no parallel computing). Assumes each basic operation takes $O(1)$ time. Simple memory structure (and random memory access). Full points if a student explains two out of these three correctly. If they get only one, they will earn 3 points.

3. (3 points) Rank the following functions in asymptotically increasing order. Formally, find an ordering $g_1, g_2, \cdots, g_k$ (here k is the number of functions given) such that $g_1 = O(g_2)$, $g_2 = O(g_3), \cdots, g_{k-1} = O(g_k)$. (For example, if you are given functions, $n^2, n, 2n$, your solution should be either $n, 2n, n^2$ or $2n, n, n^2$.)

   $$\log^{10} n \qquad n \log n \qquad 4^n \qquad \log\log n \qquad n^2$$

   Sol.

   $$\log\log n \qquad \log^{10} n \qquad n \log n \qquad n^2 \qquad 4^n$$

   -1 for giving a wrong ordering for *each* pair.

4. (3 points) The following is a pseudocode for the naive divide-and-conquer algorithm for matrix multiplication. Here, partitioning a matrix means doing so into four $n/2$ by $n/2$ (sub-)matrices.

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c_11 = a_11 · b_11
else partition A, B, and C as in equations (4.9)
    C_11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_11)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_21)
    C_12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_12)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_22)
    C_21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_11)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_21)
    C_22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_12)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_22)
return C
```

   Give the recurrence for the running time of Square-Matrix-Multiply-Recursive and solve it. We let $T(n)$ denote the running time when input matrices are n by n. No need to show how you solved it. Just state the final result along with the recurrence.

   Sol. $T(n) = 8T(n/2) + \Theta(n^2)$. (1.5 pts)
   $T(n) = \Theta(n^3)$. (1.5 pts)

5. (3 points) Describe the heapsort algorithm.

6. (2 points) Suppose we want to sort vectors $v_i = \langle v_i(5), v_i(4), v_i(3), v_i(2), v_i(1)\rangle$, where each entry has an integer value between 0 and 1000. Explain how. What is the run time? **Sol.**

   You can simply say Radix sort. Or say run counting sort starting from the least significant bit to the most significant bit. RT is $O(n)$.

7. (2 points) In the randomized quick sort algorithm, what is the probability that the max and min elements are compared to each other? Briefly explain why. Assume there are n elements to be sorted out and they all have distinct values. **Sol.** $2/n$. The two are compared to each other if and only if one of the two is chosen as the pivot in the first call of quick sort.

8. (2 points) Explain the advantage of hashing over direct address. **Sol.** Memory efficient.

9. (3 points) Huffman Code. Suppose we have a text consisting only of a, b, c, d, e where each character appears with the following frequency:

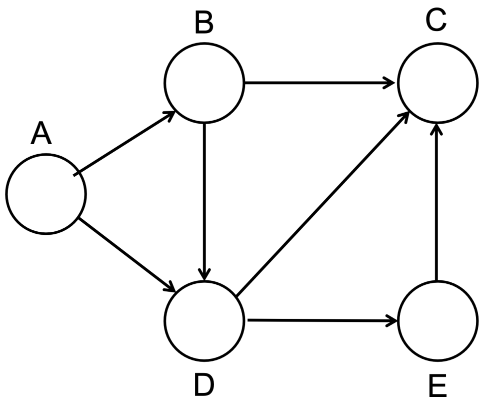
a	b	c	d	e	total
2	1	6	4	5	18

Show the code built by the Huffman algorithm as a ***tree***. When combining two trees, make sure that *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: 000
a: 001
d: 01
e:10
c:11

I believe you know how you can draw a tree corresponding to this code.

10. (6 points) Represent the following graph using *adjacency lists.* By running DFS, *topologically sort* the vertices; show the vertex ordering. You must show every node's *discover time and finish time.* When running DFS, consider vertices in *increasing alphabetical* order when considering a vertex's neighbors and starting a new DFT. Your topological sort result must be consistent with vertices' discover and finish time.



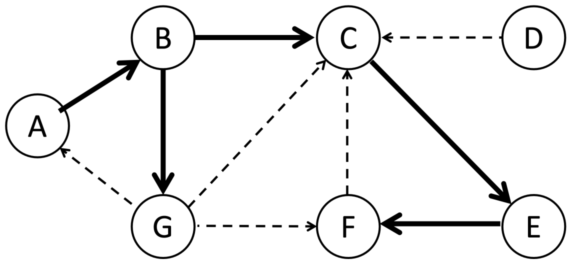
Sol.

1. 2 points
A: B → D
B: C → D
C:
D: C → E
E: C
(no penalties for not following alphabetical order)

2. 2 points
A = (1, 10)
B = (2, 9)
C = (3, 4)
D = (5, 8)
E = (6, 7)
(no penalties for not following alphabetical order)
-1 for one mistake (-2 for incorrect discover and finish time of a vertex)

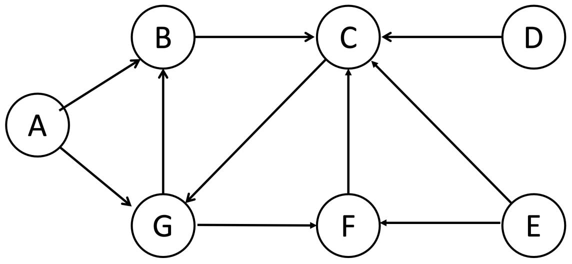
3. 2 points
A, B, D, E, C
If correct topological sort and consistent with discover, finish times, 2
If correct topological sort and inconsistent with discover, finish times, 1
If incorrect topological sort 0.

11. (4 points) The following shows a depth first forest we obtained after running DFS, where the tree edges are solid. Label other (dashed) edges with F (forward), B (back), or C (cross).



Sol. GA: B
GC: C
GF: F
FC: B
DC: C
-1.5 points for each mistake.

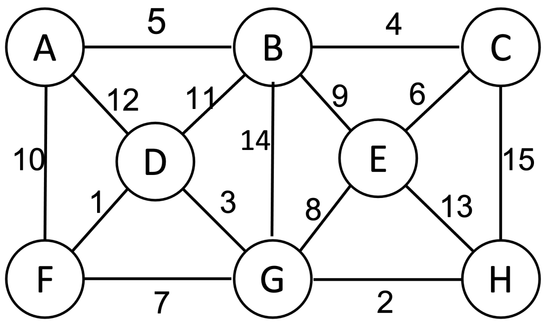
12. (4 points) Show the component graph of the following graph. You only need to show the graph.



Sol. The SCCs are $\{A\}$, $\{B, C, G, F\}$, $\{D\}$, $\{E\}$.
Edges: $(\{A\}, \{B, C, G, F\}), (\{D\}, \{B, C, G, F\}), (\{E\}, \{B, C, G, F\})$

13. (6 points) Describe an algorithm that determines if the given directed graph has a cycle or not. *Prove* its correctness. **Sol.** Run DFS. The graph has a cycle if and only if it has a back edge (Algorithm 2 points)
If there is a back edge (u, v), then, by definition u is v's descendant, which means there is a path from u to v (consisting of tree edges), which forms a cycle with the back edge. (1 point)
If there is a cycle $(v_1, v_2, ..., v_k, v_1)$, consider the first time when DFS discovers a vertex on the cycle. Suppose it is v_1 wlog. At the moment, there is a white path from v_1 to v_k. Therefore, due to the white path theorem, v_k becomes v_1's descedant. Thus, (v_k, v_1) is a back edge. (3 points)

14. (6 points) Consider the following weighted undirected graph.



(a) (2 points) Explain why edge (C, E) is safe. In other words, give a cut where the edge is the cheapest edge crossing the cut. **Sol.** e.g. $\{A, B, C\}$, and the others

(b) (2 points) Suppose we run the Kruskal's algorithm on this graph. List the edges appearing in the Minimum Spanning Tree (MST) in the order they are added to the MST. If you prefer, you are allowed to index edges by their weight. (e.g. Instead of DF, GH, ..., you can write 1, 2, ...,) **Sol.** 1, 2, 3, 4, 5, 6, 8

(c) (2 points) Suppose we run the Prim's algorithm on this graph using A as initial vertex. List the edges appearing in the Minimum Spanning Tree (MST) in the order they are added to the MST. **Sol.** 5, 4, 6, 8, 2, 3, 1

15. (5 points) Consider an undirected graph with distinct positive edge weights. *Prove* that the max weight edge on a cycle is NOT safe. **Sol.** Say (v_0, v_1) is the max weight edge on cycle $(v_0, v_1, v_2, ..., v_k, v_0)$. For the sake of contradiction, suppose it is the lightest edge crossing some cut (S, T) where $v_0 \in S, v_1 \in T$. Consider the path $(v_1, v_2, ..., v_k, v_0)$. The path must have an edge crossing the cut (S, T) and the edge must have a smaller weight than (v_0, v_1). A contradiction.

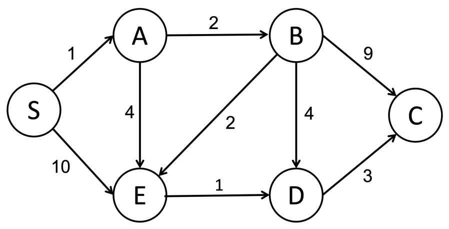
16. (5 points) Describe Kruskal's algorithm for finding the Minimum Spanning Tree of an undirected graph $G = (V, E)$, where each edge (u, v) has weight $w(u, v)$. The description must include how to use the disjoint-set data structure, which supports the following operations.

   - Make-Set(x): creates a new set whose only member is x.
   - Union(x, y): Merge two (distinct) sets containing x, y into one.
   - Find-Set(x): return (a pointer to) the representative of the set containing x.

   Sol. For each vertex v, create a set including x (by Make-Set(v)). Consider edges in increasing order of their weight. When considering an edge (x, y), we know x and y belongs to different components iff Find-Set(x) $\neq$ Find-Set(y). If they do, we choose it and call Union(x, y).

17. (5 points) We would like to find a shortest path from s to t in a directed graph $G = (V, E)$ where edges have weights with possibly negative weights. Suppose that we know that there is a shortest path from s to t that consists of at most k edges. Describe an algorithm that finds a shortest path from s to t in $O(k(E + V))$ time. Please make your description concise.
Sol. We repeat the following k times: Relax every edge exactly once.

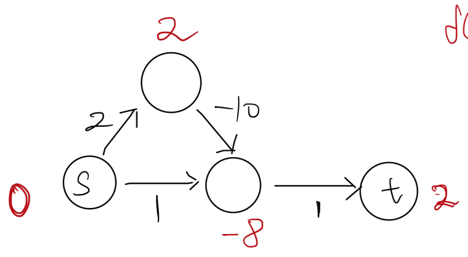
18. (4 points) Recall that the Dijkstra algorithm dequeues from the min-priority queue a node v with the smallest value of $v.d$ in each iteration. List the vertices in the order they are dequeued from the queue.



Sol. S, A, B, E, D, C

19. (4 points) Give an input graph that has negative weights on some edges for which Dijkstra fails to find a shortest path from s to t. You must i) specify s and t; ii) show $v.d$ for all vertices at the end of the Dijkstra algorithm; and iii) state the shortest distance from s to t, i.e. $\delta(s, t)$. Your graph must *not* include a negative-weight cycle. To get full points, your graph must have *at most four* vertices, including s and t. If we use more than five vertices, you will get 0 points.

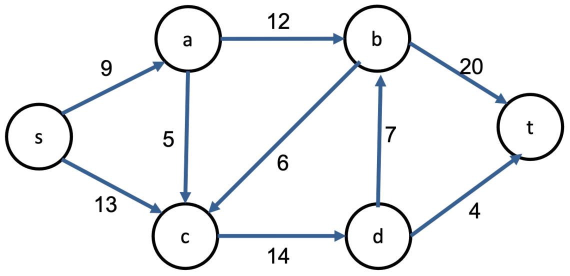
Sol.



$\delta(s,t) = -7$

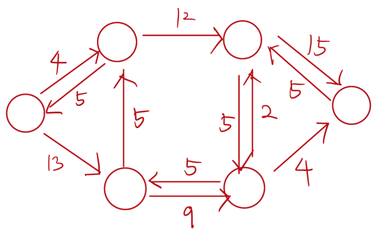
20. (2 points) Explain the "optimality of subpaths" lemma. **Sol.** Any subpath of a shortest path must be a shortest path between the two ending points of the subpath.

21. (4 points) Consider the following input graph to the max flow problem, where the number next to each edge indicates its capacity. Suppose you augment a flow of value 5 along the path $\langle s, a, c, d, b, t\rangle$. Show the resulting residual graph.



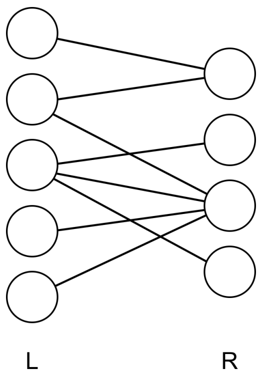
Sol.

-1 for each mistake



22. (4 points) Explain how you can find an s-t min-cut using an algorithm for computing an s-t max flow. **Sol.** After finding a max flow, let S be the vertices reachable from s, and T be the other vertices. Return (S, T).

23. (5 points) In the Maximum Bipartite Matching, we are given as in put a bipartite (undirected) graph $G = (V = L \cup R, E)$ (for example, see the graph below), and the goal is to choose as many edges sharing no end points as possible. Explain how to solve this problem using the max flow algorithm.



Sol. Create a source vertex s and connects it to all L vertices. Create a sink vertex t and connects all R vertices to t. Assign capacity 1 to every edge. Solve the max flow (say using FF). Choose the edges having non-zero flow on them.