Quiz 1

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have 75 minutes to earn 180 points. **However, 150 points is viewed as full mark. Anything above that are bonus points.** Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- **This is a closed-book exam**. No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions **using non-erasable pens** in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is also okay but not required.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.
- Turn in your exam paper to your designated AI after the exam, and sign your name on the sign-in sheet.

Problem	Parts	Points	Grade	Grader
Name	0	4		
1	9	81		
2	4	40		
3	1	25		
4	1	30		
Total	–	180		

Name: ______________________________

Problem 1. Single Choice Problems [81 points] (9 parts)

For each of the following questions, choose the unique best answer and write the corresponding letter in the bracket before the problem. There is no need to justify the answers. Each problem is worth 9 points.

(a)( ) In class, we talked about the Stable Matching Problem and presented the Gale-Shapley algorithm. Which of following statements is wrong?

A. The goal of this problem is to produce a one-to-one matching between men and women so that the matching is stable.

B. A stable matching is one such that with no unstable pairs.

C. An unstable pair is a pair of men (say X and Y) so that X prefers Y's partner to his current partner and Y prefers X's partner to his current partner.

D. The Gale-Shapley algorithm always generates a stable matching.

Solution: C. An unstable pair is a pair of unmatched man and woman (say X and Y) so that X prefers Y to his current partner and Y prefers X to her current partner.

(b)( ) In the Union-Find data structure with path compression, after we do a $find(x)$ operation, the height of the tree that x is in

A. always decreases.

B. may decrease or remain unchanged.

C. may decrease, remain unchanged, or increase.

Solution: B.

(c)( ) In a weighted graph with n nodes and $m = \frac{n \log n}{\log \log n}$ undirected edges. If we run Boruvka's algorithm for r rounds and then apply Prim's algorithm to the remaining graph, the best choice of r and the total runtime of the procedure is (the bases of all logarithms are 2)

A. when $r = \log n$, runtime becomes $O(m \log n)$.

B. when $r = \log \log n$, runtime becomes $O(m \log \log n)$.

C. when $r = \log \log \log n$, runtime becomes $O(m \log \log \log n)$.

D. when $r = \log \log \log \log n$, runtime becomes $O(m \log \log \log \log n)$.

Solution: C.

In Parts d), e), f), and g), let $G = (V, E)$ be an undirected graph with edge weights $w : E \to \mathbb{R}$. In addition, let n denote the number of vertices (i.e. $|V|$) and m be the number of edges (i.e. $|E|$).

(d)( ) Suppose all the edges have distinct positive weights and the minimum spanning tree is T. If we square all the edge weights and compute the Minimum Spanning tree (MST) again, do we still get the same tree structure?

A. Yes, we still get the same tree structure.
B. No, since the edge weights have changed.

Solution: A.

(e)( ) How many of the following statements are true?

i. Prim's and Kruskal's algorithms will always return the same MST.
ii. Prim's algorithm only works if the weights are positive.
iii. An MST for a connected graph has exactly $n - 1$ edges.
iv. A graph where every edge weight is unique (there are no two edges with the same weight) has a unique MST.
v. The MST can be used to find the shortest path between two vertices.

A. 0. B. 1.
C. 2. D. 3.
E. 4. F. 5.

Solution: C. Only statements iii and iv are true.

(f)( ) Assume all edge weights are positive. Consider a cycle $\langle v_1, v_2, \ldots, v_k, v_{k+1}\rangle$ in G, where $v_{k+1} = v_1$, and let (v_i, v_{i+1}) be the edge in the cycle with the largest edge weight. It is possible that (v_i, v_{i+1}) might belong to an MST.

A. This statement is True when the edge weights are not distinct.
B. This statement is True when the edge weights are distinct.
C. Neither of A and B is correct; the statement is False in general.

Solution: A.

(g)( ) Assume all edge weights are positive. Let s and v be two distinct vertices in G and suppose $p = \langle s, v_1, v_2, \ldots, v_k, v\rangle$ is one possible shortest path from s to v. Then p is still a shorted path from s to v if

A. for every edge $e \in E$, let $w(e) = 2w(e)$.
B. for every edge $e \in E$, let $w(e) = w(e) + 2$.
C. for every edge $e \in E$, let $w(e) = w(e) - 2$.
D. for every edge $e \in E$, let $w(e) = w(e)^2$.

Solution: A.

(h)( ) We have seen in class that Dijkstra's algorithm with a Fibonacci heap solves the Single Source Shortest Path Problem (SSSP) with non-negative edge weights in time $O(n \log n + m)$, where n is the number of vertices and m is the number of edges. However, under certain special assumptions, one can modify the algorithm and solve the SSSP in time $O(n + m)$. Such assumptions can be

A. all edge weights are odd numbers in range $[-50, 50]$.
B. all edge weights are real numbers in range $[0, 10]$.
C. all edge weights are integer numbers in range $[0, 100]$.
D. both A and B.
E. both A and C.

Solution: C.

(i)( ) Let us implement the Union-Find data structure with both union-by-size and path-compression techniques. Suppose there are n elements, then the worst-case time costs for each operation and a sequence of n operations are respectively

A. $O(\alpha(n))$ and $O(n \cdot \alpha(n))$.
B. $O(\log n)$ and $O(n \cdot \alpha(n))$.
C. $O(n)$ and $O(n^2)$.
D. $O(1)$ and $O(n)$.

Solution: B.

Problem 2. Single Source Shortest Paths [40 points] (4 parts) Suppose you're using Dijkstra's algorithm starting from some source vertex s. The table on the left shows all edges and their weights in the graph. The table on the right shows the two arrays used in the algorithm (dist[] and from[]) immediately after node 4 has been finalized and relaxed.

edge	weight
$0 \to 2$	3.0
$0 \to 3$	1.0
$1 \to 3$	1.0
$1 \to 6$	5.0
$2 \to 1$	2.0
$3 \to 1$	17.0
$3 \to 2$	14.0
$3 \to 5$	3.0
$3 \to 7$	8.0
$4 \to 7$	1.0
$5 \to 1$	4.0
$6 \to 0$	3.0
$6 \to 2$	1.0
$6 \to 4$	1.0
$7 \to 4$	2.0

v	dist[]	from[]
0	∞	null
1	7.0	5
2	14.0	3
3	0.0	null
4	10.0	7
5	3.0	3
6	12.0	1
7	8.0	3

(a) [10 points] Give the order in which the first 5 vertices were finalized and relaxed.

3	5	1	7	4

(b) [10 points] Fill in the table below to reflect the new shortest paths tree (by changing edgeTo[] and distTo[]) after the next vertex is relaxed. Only fill in those entries which change from the values shown in the table above.

v	dist[]	from[]
0	15	6
1		
2	13	6
3		
4		
5		
6		
7		

(c) [10 points] Suppose we would like to change the weight of the edge $6 \to 4$ to a negative number $a < 0$. Please specify the range of a so that Dijkstra's algorithm does not output the correct shortest path distances.

Solution: When we change the weight of the edge $6 \to 4$ to a number less than -2, the shortest path from 3 to 4 will become less than 10, which means Dijkstra's algorithm does not output the correct shortest path distances.

(d) [10 points] Suppose you are given a directed weighted graph $G = (V, E, w)$ where all but one edge (namely (u, v)) has negative weight. Describe how you can perform Dijkstra's algorithm twice (perhaps on some modified graphs) to compute the correct shortest path distances from a designated source $s \in V$, in time $O(|V| \log |V| + |E|)$. You may assume that there is no cycle whose weight is negative in the graph G. You do not have to prove the correctness or runtime of your algorithm.

Solution:

Remove the negative edge (u, v) from the original graph G, and then run Dijkstra's algorithm twice. First, we run it by setting s as the source node and get the distance D_1. Then, we run it again by setting v as the source node and get the distance D_2. The shortest path between s and t will be $\min(D_1[t], D_1[u] + D_2[t] + w(u, v))$.

Problem 3. **k-friendship** [25 points] Suppose you represent the friendships of students in the class as an undirected graph $G = (V, E)$ (i.e. there exists an edge between vertices u and v if these students are friends). For any integer k, we say that a group of students, $S \subseteq V$, is "k-friendly" if every student in S has at least k friends in S.

Describe an efficient algorithm (i.e. whose runtime is polynomial in $|V|$) which, given the graph G and an integer $k > 0$, finds the largest set $S \subseteq V$ that is k-friendly; if no such set exists, then the algorithm should say so. Also, justify the correctness of your algorithm and its runtime.

[We expect a description or pseudocode of your algorithm as well as a brief justification of its correctness and runtime.]

Solution:

Algorithm. We proceed by rounds/iterations; at the i-th round, the remaining graph is $G_i = (V_i, E_i)$. Initially $G_1 = G$. For each Round $i = 1, 2, 3, \ldots$, if there exists a vertex in G_i with degree less than k in G_i, remove it and all its incident edges; otherwise all vertices have degree at least k in G_i, and the algorithm simply returns V_j. If G_i eventually becomes empty, report "no solution".

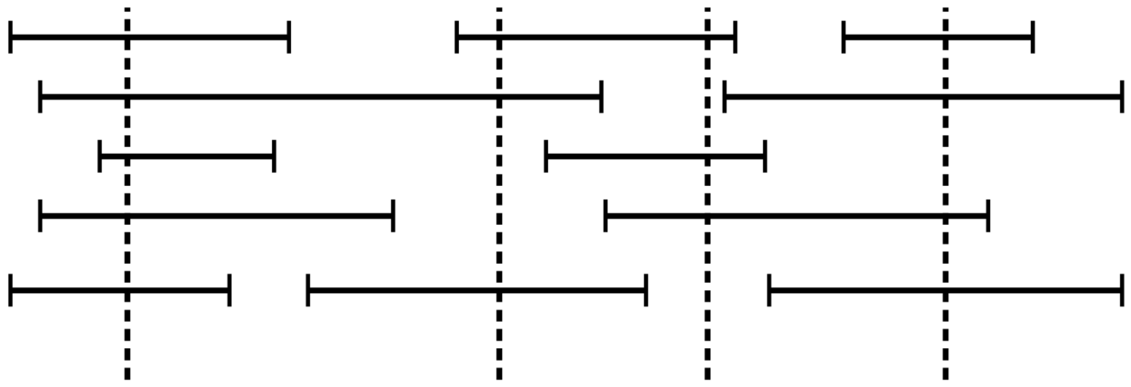
Correctness. Let F^* be the k-friendly set with maximum number of vertices. Two observations:

- For all i, $V_i \supseteq F^*$. We prove this by induction. For $i = 1$ we trivially have $V_1 = V \supseteq F^*$. For $i \geq 2$, by induction hypothesis we have $V_{i-1} \supseteq F^*$. In Round $(i - 1)$, if the algorithm deletes $v \in V_{i-1}$, we know that v has less than k neighbors in V_{i-1}, and therefore v has less than k neighbors in F^*. Therefore $v \notin F^*$ and $V_i = V_{i-1} \setminus \{v\} \supseteq F^*$.
- When we terminate in Round j, V_j equals to the solution F^*. This is because according to our algorithm, every vertex in G_j has degree at least k, therefore V_j is a k-friendly set, and $|V_j| \leq |F^*|$. By our first observation we have $V_j \supseteq F^*$. In all, we get $V_j = F^*$.

Runtime. Let $n = |V|$ and $m = |E|$. A naive runtime upper bound is $O(n^2 + m) = O(n^2)$, as follows. There are at most n rounds and in each round we can use $O(n)$ time to find a vertex with degree less than k or report such vertex does not exist (by keeping the degree for each vertex in an array). We also need to remove an vertex and all its incidental edges, and update the degree array. It takes $O(1)$ time to remove an edge and update the degree of the incidental vertex. There are at most m edges to be deleted, and therefore the deletion process takes $O(m)$ time.

Problem 4. [30 points] Suppose that you are in charge of the PA system at a school. When you make an announcement over the PA system, all classes currently in session hear it. There is a very important announcement you need to make, and to ensure that everyone hears it, you want to announce it enough times to guarantee that every single class hears the announcement at least once. Because some students might be in multiple classes during the day, you want to minimize the number of times that you have to make the announcement. It's fine if you make the announcement two or more times during a particular class; it might be a nuisance, but your priority is ensuring that each class hears it at least once.

For example, if you were given the set of classes (denoted by their timespans), one possible set of times you could make the announcements is given by the vertical dotted lines:



This particular set of times is not a minimum set of times; it is possible to cover each class using only three announcements.

You are given as input a list of intervals representing the start and end times of each class. Design a polynomial-time algorithm that finds a minimal set of announcement times that is guaranteed to reach every class. Then:

- Describe your algorithm.
- Prove that your algorithm finds a minimal set of announcement times.
- Prove that your algorithm runs in polynomial time.

For simplicity, you can assume that no class starts as soon as another one ends and that when you start making the announcement all classes in session will hear it, including those that are just starting or just ending.

(You may write your answer to Problem 4 on the next page.)

Solution:

Algorithm.

1. Sort the classes in non-decreasing order (break ties arbitrarily) by the finishing time and get the following sequence: $[s_1, f_1], [s_2, f_2], [s_3, f_3], \dots, [s_n, f_n]$, where each interval corresponds to the span of the class.

2. Identify first class in the sequence, and let it be (s, f). Add an announcement time $t = f$, and remove all the classes those are active at time t from the sequence. Repeat this process until the sequence is empty.

Correctness.

1. Assume the announcement times found by our algorithm $S = \{t_1 < t_2 < \cdots < t_m\}$ is not optimal. Let us select an optimal set $S' = \{t_1 < t_2 < \cdots < t_k < t'_{k+1} < \cdots < t'_{m'}\}$ so that k is maximized (i.e. k the the number so that the first k times in S agrees with those in S'). Note that we have $m' < m$. We also have $k < m'$ because otherwise (when $k = m'$) the algorithm would have stopped after choosing $t_{m'}$.

2. Observe that $t'_{k+1} \le t_{k+1}$ because otherwise the class identified for t_{k+1} at Step 2 of the algorithm would not be covered by S'.

3. Let us construct a new set of times $S'' = \{t_1 < t_2 < \cdots < t_k < t_{k+1} < t'_{k+2} < \cdots < t'_{m'}\}$ and S'' is also a valid solution. This is because all classes those are active at time t'_{k+1} but not active at time t_k will also be active at time t_{k+1} (as t_{k+1} is the earliest finish time among all such classes according to our algorithm).

4. Now S'' is another optimal solution that agrees with our output S at the first $k+1$ times, contradicting to our assumption that k is the maximum number of shared first few announcement times.

Runtime. It is straightforward to see that the runtime of the algorithm is $O(n^2)$. A careful implementation can improve the runtime to $O(n \log n)$.