CSIE 2136: Algorithm Design and Analysis (Fall 2021)

Final Exam

Time: 14:20-17:20 (180 minutes), January 6 2022

Instructions

- This is a 3-hour closed-book exam, and there are seven questions worth a total of 115 points. The maximum of your score is 100, so you can allocate your time and select problems based on certain *optimal strategies* of your own.
- Please write clearly and concisely; avoid giving irrelevant information.
- You have access to the appendix on the last page, and you can use basic data structures (i.e., array, stack, queue, deque, heap, balanced search tree, doubly linked list) without writing their implementation details. In addition, you can assume that the time complexity of **sorting** n numbers is $O(n \log n)$.
- You can assume $P \neq NP$ unless specified otherwise.
- Please use a **separate answer sheet** for each problem. Please write down **your name, student ID, and problem number** in the **upper-right** corner of **every** answer page (see the example below). You will get **2 points** for complying with this policy.

Name: ADA Student ID: B09902xxx Problem K

- **You need to prove the correctness and time complexity of the algorithm you provide in all problems unless the problem does not require**.
- Grading policy: If a problem asks you to design an $O(f(n))$ algorithm, you can get partial points if you design an algorithm worse than $O(f(n))$. However, we will not accept any algorithm running in exponential time or $O(n!)$ unless the problem asks you to do so.

Problem Outline

- Problem 1 - Short Answer Questions (30 points)
- Problem 2 - Graph Theory (20 points)
- Problem 3 - Amortized Analysis (15 points)
- Problem 4 - Hardness (20 points)
- Problem 5 - Shopping Discount (25 points)
- Problem 6 - Feedback (3 points)
- Name, ID, and problem number on each page (2 points)

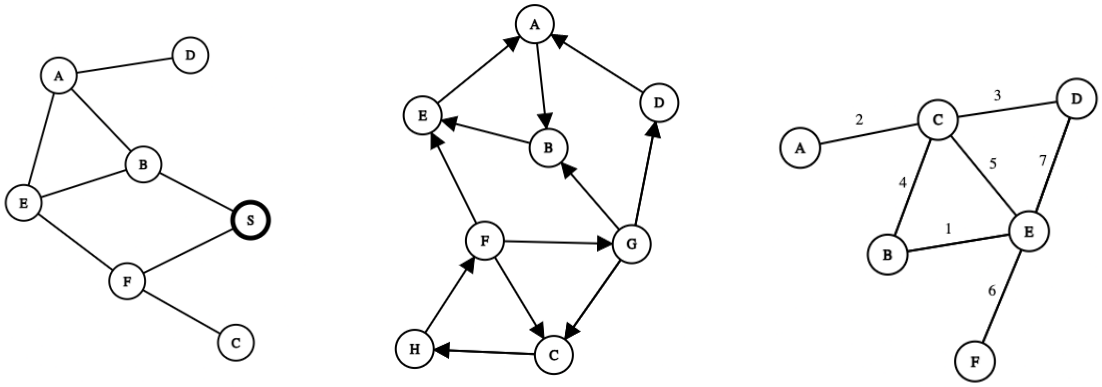
Problem 1 - Short Answer Questions (30 points)

(a) (20 points) Answer the following true or false questions. **No explanation is needed**.

1. T/F: A connected undirected acyclic graph has $|E| = |V| - 1$.
2. T/F: A connected undirected graph G has an Eulerian cycle if and only if every vertex of G has even degrees.
3. T/F: If we perform DFS on a directed acyclic graph, there will be no back edge.
4. T/F: Dijkstra algorithm may not terminate if negative cycles exist.
5. T/F: Given a directed acyclic graph G, we create G' by removing one edge from G. If T is a topological order on G, then T must also be a valid topological order on G'.
6. T/F: If problem A is NP-complete, given a problem instance I of A, no algorithm can solve I in polynomial time.
7. T/F: NP-Complete problems can be reduced to each other in polynomial time.
8. T/F: If A can be reduced to B in $O(n^2)$ time and there is an $O(n^3)$ algorithm for B, there is an $O(n^3)$ algorithm for A.
9. T/F: If A $\leq_p$ B, and there is a 2-approximation algorithm for B, there is a 2-approximation algorithm for A.
10. T/F: The randomized 2-approximation MAX-CUT algorithm we learned in class can always output an approximate answer whose cost is at least half of the optimal answer.

(b) Simple Graphs (10 points)

1. (2 points) Use BFS to traverse the graph in Figure 1a from vertex s. Write down the order of visited vertices. If there are multiple valid orders, output the smallest lexicographical order.
2. (2 points) Use DFS to traverse the graph in Figure 1a from vertex s. Write down the order of visited vertices. If there are multiple valid orders, output the smallest lexicographical order.
3. (2 points) Identify all strongly connected components of the graph in Figure 1b, and draw the component graph.
4. (4 points) Recall that to find single-source shortest paths on a graph $G = (V, E)$, the Bellman-Ford algorithm first performs $|V| - 1$ rounds of edge relaxation. To detect negative-weight cycles, it then checks whether $v.d > u.d + w(u, v)$ for all $(u, v) \in E$. Suppose there is a negative-weight cycle C on G, is it correct that **every** edge (u, v) on the cycle C must satisfy $v.d > u.d + w(u, v)$ when the Bellman-Ford algorithm performs the check? Please explain why.



(a) (b) (c)

Figure 1: Graphs

Problem 2 - Graph Theory (20 points)

For simplicity, you can assume all edge weights are distinct in this problem.

2-(a) (5 points) Recall the cycle property we learned in class: "Let C be any cycle in a weighted undirected graph G, and let e be an edge with the maximum weight on C. The minimum spanning tree of G does not contain e." Please prove the correctness of the cycle property.

2-(b) (10 points) A *bottleneck spanning tree* T of an undirected graph G is a spanning tree of G whose largest edge weight is minimum over all spanning trees of G. We say that the value of the bottleneck spanning tree is the weight of the maximum-weight edge in T. Please prove that a minimum spanning tree must be a bottleneck spanning tree. (Hint: Show a contradiction if there is a minimum spanning tree T that is not a bottleneck spanning tree. It may be helpful to consider the largest-weight edge of T and the cycle property. If you get stuck, you can still use this as a fact to answer the following subproblems.)

2-(c) (2 points) On the graph in Figure 1c, identify a bottleneck spanning tree that is not a minimum spanning tree.

2-(d) (3 points) A *bottleneck shortest path* between two vertices s and t of a graph G is a path whose largest edge weight is minimized over all possible paths. One of the following statements is correct. Please use your best judgment to find out. **No explanation is needed.**

(i) The path between s and t on the minimum spanning tree T must be a bottleneck shortest path between s and t.

(ii) The path between s and t on a bottleneck spanning tree T must be a bottleneck shortest path between s and t.

(iii) A shortest path between s and t on G must be a bottleneck shortest path between s and t.

Problem 3 - Amortized Analysis (15 points)

Consider the following two fundamental stack operations:

1. PUSH(S, x): push an integer value x onto stack S.
2. POP(S): pop the top of stack S and return the popped object. Calling pop on an empty stack generates an error.

Now we want to augment it with the following new operations:

1. MULTIPOP(S, k): pop the top k elements of stack S
2. REDUCE(S, k): pop the top k elements of stack S then push back the elements whose values are less than the median of the k popped elements in the original order.

Please answer the following questions:

3-(a) (5 points) Suppose we would like to support MULTIPOP but not REDUCE. We implement MULTIPOP using the following algorithm:

Algorithm 1 MULTIPOP(S, k)

```
while not STACK-EMPTY(S) and k > 0 do
    POP(S)
    k = k − 1
end while
```

Please show that the amortized time complexity of the **three** operations (PUSH, POP, and MULTIPOP) are all $O(1)$ by **one of the amortized analysis methods taught in class**.

3-(b) (10 points) Now we would like to support both MULTIPOP and REDUCE. Assuming the median of n numbers can be found in $O(n)$ time complexity, please design an algorithm that implements REDUCE such that the amortized time complexity of all the **four** operations are still $O(1)$. Remember to show the time complexity analysis by **one of the amortized analysis methods taught in class** as well.

Problem 4 - Hardness (20 points)

The following is the definition of the K-Center-Triangle problem:

> Given a complete graph (V, E) with a distance function $d(\cdot, \cdot)$ satisfying the triangle inequality (i.e., $d(u, v) \leq d(u, x) + d(x, v)\ \forall u, v, x \in V$) and an integer k such that $1 \leq k \leq |V|$, find a set $F \subseteq V$ with $|F| = k$ which minimizes $\max_{j \in V} d(j, F)$. Here, $d(j, F)$ is defined as the minimum distance from j to any vertex in F, i.e., $\min_{f \in F} d(j, f)$.

Simply put, the problem aims to find k *center vertices* in V such that the maximum distance from each vertex to its nearest *center vertex* is minimized.

4-(a) (7 points) Please complete the 2 parts marked with "**?**" in the following reduction from Vertex-Cover to K-Center-Triangle and **prove the "only if" direction**. That is, the answer to the instance of Vertex-Cover is "yes" implies the answer to the reduced instance of K-Center-Triangle is also "yes":

Given an instance of Vertex-Cover (V, E, k)

1. Construct a new graph $(V \cup V_e, E')$ where $V_e = \{v_e \mid e \in E\}$ and $E' = \{(u, v) \mid u, v \in V \cup V_e\}$. The distance function of the new graph is assigned as follows:
   - for $e = (u, v) \in E$, $d(v, v_e) = d(u, v_e) = 1$
   - for $(u, v) \in E$, $d(u, v) =$ **?**
   - for all the other edges (x, y) that have not been assigned distance, $d(x, y) =$ **?**
2. Output $(V \cup V_e, E', d, k)$

4-(b) (10 points) Please prove the "if" direction of the above reduction. That is, the answer to the reduced instance of K-Center-Triangle is "yes" implies the answer to the instance of Vertex-Cover is "yes".

4-(c) (3 points) Please conclude that there is no $(2 - \epsilon)$-approximation for K-Center-Triangle for any $\epsilon > 0$ when $P \neq NP$.

Problem 5 - Shopping Discount (25 points)

Today is the ADA shop's anniversary sale. The ADA shop offers customers with M discount combinations among the N items. Generally speaking:

- The i^{th} item's original cost is p_i dollars ($p_i \geq 0$ for all $i = 1, 2, \ldots, N$).
- Each discount combination contains a non-empty subset of items (possibly one item only) and a cost d_j, indicating that a customer can buy this set of items together by spending d_j dollars ($d_j \geq 0$ for all $j = 1, 2, \ldots, M$).

However, to prevent evil customers from buying and reselling, a customer **cannot buy the same item twice**. Formally, we say that two item sets are in *conflict* if and only if there is an item belonging to both sets. Thus, customers cannot buy two combinations (including buying an item at its original price) simultaneously if they are in *conflict*.

Vivian, who loves shopping, will not miss this opportunity, of course. She decides to buy **as many items as she can** with the limit that she has only C dollars.

For example, if:

- $N = 5, M = 2, C = 7$
- $p = [1, 4, 5, 4, 5]$
- Combination 1: $\{1, 4, 5\}$ cost 4 dollars
- Combination 2: $\{1, 2, 3\}$ cost 3 dollars

The best deal for Vivian is to buy items $\{1, 2, 3, 4\}$. Since she can spend 3 dollars to buy combination 2, and then buy the item 4 at its original price, 4 dollars. Noting that she cannot buy combinations 1 and 2 simultaneously since they are in conflict, even though she has enough money.

We define the problem above as the *shopping discount* problem. Please answer the following questions:

5-(a) (2 points) Describe a $O(N \log N)$-time **greedy algorithm** to solve the shopping discount problem when $M = 0$. No proof is needed.

Vivian finds it challenging to calculate the maximum items she can buy. Please help her prove that the decision version of the *shopping discount* problem is in NP-Complete by the following steps.

5-(b) (2 points) Describe the decision version of the shopping discount problem. You do not need to explain the definition of items and combinations again.

※ In the following problems, you can regard any graph as a simple graph (i.e., no multi-edges, no self-loops).

5-(c) (10 points) Recalling the famous NP-Complete problem, *independent set*:

> Given a graph with N vertices and M edges, decide if there is a set of vertices with size $\geq K$ such that any two vertices of the set do not share an edge.

Let us define a different problem $\mathcal{A}$ which is similar to the independent set problem:

> Given a graph with N vertices and M edges, decide if there is a set of vertices with **the sum of the vertices' degrees** $\geq K$ such that any two vertices of the set do not share an edge.

Construct a polynomial reduction that reduces problem $\mathcal{A}$ to the shopping discount problem and then prove its correctness. Please clearly describe your construction. Avoiding descriptions like "big enough."

Hint: what if we regard an edge as an item?

5-(d) (7 points) Construct a polynomial reduction that reduces the independent set problem to the shopping discount problem and then prove its correctness.

The reduction of **5-(c)** can serve as a hint for this problem. However, notice that we **do not expect** you to prove that "independent set $\leq_p$ problem $\mathcal{A} \leq_p$ shopping discount problem".

5-(e) (4 points) Prove that the *shopping discount* problem is NP-Complete. You can use the conclusion of **5-(d)** directly (i.e., assuming that the polynomial reduction is possible) without answering it again in this problem.

Problem 6 - Feedback (3 points)

We have the tradition of letting students write feedbacks about the course in the exam. Please write down anything that you would like to tell us.

Appendix

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Figure 2: Dijkstra's Algorithm