



CSIE 2136: Algorithm Design and Analysis (Fall 2022)

Final

Time: 14:30-17:30 (180 minutes), December 22 2022

Instructions

- This is a 3-hour closed-book exam. There are 7 problems worth a total of 120 points. If your raw score exceeds 100, it will be capped to 100.
- Please write clearly and concisely; avoid giving irrelevant information.
- You are allowed to use basic data structures (limited to array, stack, queue, deque, heap, balanced search tree, doubly linked list) without writing their implementation details. In addition, you can assume that sorting N numbers runs in $O(N \log N)$.
- Please use the assigned answer sheets for each problem. For each page, please write down **your name** and **student ID** on every paper. You will get **2 points** for complying with this policy.
- You **do not** need to prove the correctness and time complexity of your algorithms unless a problem requires you to do so.
- *TL; DR*: For each problem, if the description starts with italic text *TL; DR*, the following lines contains only interesting story with no critical information. Thus, you may rush into the subproblems if you would like to save some time.
- Answer sheet notations
 - : Your answers in the blanks labeled with scanning machines may be graded in a fancy way like auto judging by AI. So please write down your answer clearly and properly large especially in these cases.
 - : The words after this symbol are the keywords of the corresponding task. This is simply a friendly design to prevent you from misplacing your answer.

Problem Outline

- Problem 1 - Decision Problem (15 points)
- Problem 2 - Minimizing a Tree's Height (15 points)
- Problem 3 - Amortize Analysis (20 points)
- Problem 4 - Longest Simple Path (25 points)
- Problem 5 - Prove the Correctness (20 points)
- Problem 6 - Picky Roommate (20 points)
- Problem 7 - Goodbye 2022 Hello 2023 (3 points)
- Name, ID, and problem number on each page (2 points)

Problem 1 - Decision Problem (15 points)

TL; DR: Explain each term within 3 sentences.

ADA 2022 Lecturing Team has a decision problem to solve — are 80% of the students familiar with the terms mentioned in the class? To prove that you do know them, for each term, please explain what it is within 3 sentences.

- (a) (3 points) What is a decision problem?
- (b) (3 points) What is a strongly-connected component?
- (c) (3 points) What is a topological order in a directed acyclic graph?
- (d) (3 points) What is a polynomial-time reduction?
- (e) (3 points) What is a spanning tree in an undirected graph?

Problem 2 - Minimizing a Tree's Height (15 points)

Given a free tree (a tree without any designated root), our goal is to choose a vertex as the root of the tree such that the height of the tree can be minimized.

Considering all simple paths on this tree, we define the length (i.e. the number of edges) of the longest simple paths on the tree as x , and a set S contains all vertices on all of the longest simple paths.

Please simply explain the correctness of the following statement.

- (a) (5 points) As we define the height of a rooted tree by the length of the longest simple path from the root to a leaf, the height of the tree with an arbitrary root $\geq \left\lceil \frac{x}{2} \right\rceil$.
- (b) (5 points) The vertex v we choose to minimize the height satisfies $v \in S$.
- (c) (5 points) The median vertex of any of the longest simple paths is an answer to the problem.

Note: We say w is a median vertex on a path from u to v means that the distance between w and u and the distance between w and v differs by at most 1.

Problem 3 - Amortized Analysis (20 points)

Consider a heapified array storing $O(n)$ keys, and this data structure supports the following operations:

- **INSERT(i)**: insert a new key i into the current heapified array cost $c \log n$ worst-case time.
- **QUERY_MAX**: obtain the maximum key in the current heapified array cost c worst-case time.
- **DELETE_MAX(k)**: delete the k largest keys from the heapified array cost $k \cdot c \log n$ worst-case time.

Here, c is a positive real constant.

Suppose that we apply a sequence of n operations on an initially empty heapified array. We define a potential function Φ to analyze the amortized cost of each operation. Please fill in the blanks on the answer sheet to complete the analysis such that the total amortized cost of a sequence of n operations reaches the tightest bound.

Note: Do not write any asymptotic notations in the blanks as long as you are not required to.

Problem 4 - Longest Simple Path (25 points)

In this problem, you can assume $P \neq NP$, and the HAMILTONIAN-PATH-PROBLEM is NP-Complete (Check the appendix out if you have forgotten what HAMILTONIAN-PATH-PROBLEM is). You can use any single-source shortest path (SSSP) algorithm mentioned in the appendix as a blackbox, as long as you specify its name, input, and output clearly. You can reuse the answer to one subproblem in another if appropriate.

LONGEST-SIMPLE-PATH-PROBLEM (LSP). Given a weighted directed graph $G = (V, E)$, a source vertex $s \in V$, a destination vertex $t \in V$, and a weight function $w : E \rightarrow \mathbb{R}$ that maps an edge $e \in E$ to a real number, the LSP problem is to find a longest simple path from s to t .

- (a) Consider the LSP problem on various types of graphs. Specifically, in each subproblem, you will be given additional constraints on G and w . Please show whether its *decision problem* is NP-hard or P. If the problem is NP-hard, you're required to prove it through reduction. If the problem is P, you're required to describe an efficient algorithm to solve the *optimization problem*, and your grade will depend on the efficiency of your algorithm. Subproblem are independent.

(i) (4 points) G is a DAG.

(ii) (4 points) $\forall e \in E, w(e) \in \mathbb{R}^-$.

(iii) (7 points) $\forall e \in E, w(e) \in \mathbb{R}^+$.

- (b) (10 points) Now we consider another variant, whose differences from the LSP problem are highlighted in bold.

LONGEST-COMPLETE-PATH-PROBLEM (LCP). Given a weighted, directed, **complete** graph $G = (V, E)$, a source vertex $s \in V$, a destination vertex $t \in V$, and a weight function $w : E \rightarrow \mathbb{R}$ that maps an edge $e \in E$ to a real number, the LCP problem is to find a longest simple path from s to t , and **the path must visit all vertices exactly once**.

Does a polynomial-time 2-approximation algorithm for the LCP problem exist? If your answer is YES, describe such an algorithm. If your answer is NO, prove its inapproximability.

Note: The value of an optimal solution to the LCP problem may be negative. A 2-approximation algorithm for the LCP problem should satisfy

- If $C^* \geq 0$, then $C \geq C^*/2 \geq 0$.
- If $C^* < 0$, then $2C^* \leq C < 0$.

As you can see, this definition preserves the inequality $\max(\frac{C}{C^*}, \frac{C^*}{C}) \leq \rho(n)$, where C^* and C denote the value of an optimal solution and the approximation algorithm, respectively.

Problem 5 - Prove the Correctness (20 points)

There are N real-number variables a_1, a_2, \dots, a_N .

Let \mathbb{P} be the set collecting all the possible sequences which are permutations of $1, 2, \dots, N$. For example, if $N = 3$, $\mathbb{P} = \{\langle 1, 2, 3 \rangle, \langle 1, 3, 2 \rangle, \langle 2, 1, 3 \rangle, \langle 2, 3, 1 \rangle, \langle 3, 1, 2 \rangle, \langle 3, 2, 1 \rangle\}$.

Let statement $P(\sigma)$ be defined as $a_{\sigma_1} \leq a_{\sigma_2} \leq \dots \leq a_{\sigma_k}$, where σ is a sequence with length k .

There are M lemmas $P(\sigma_1), P(\sigma_2), \dots, P(\sigma_M)$, where $\sigma_i \in \mathbb{P}$, $\forall i = 1, 2, \dots, M$.

Now, for the problem itself.

REASONABLE-DECISION-PROBLEM (RDP). Can you, by reasoning from the M lemmas, prove the correctness of the statement $P(\mu)$ where each term in the sequence μ belongs to $\{1, 2, \dots, N\}$?

- (a) (5 points) Given $N = 5$ and $M = 2$ lemmas: $\begin{cases} a_1 \leq a_2 \leq a_3 \leq a_4 \leq a_5 \\ a_3 \leq a_1 \leq a_2 \leq a_5 \leq a_4 \end{cases}$.

Show if the statement $a_2 \leq a_1 \leq a_1 \leq a_3 \leq a_4$ is reasonable from the M lemmas.

In other words,

$$\left(a_1 \leq a_2 \leq a_3 \leq a_4 \leq a_5 \wedge a_3 \leq a_1 \leq a_2 \leq a_5 \leq a_4 \right) \stackrel{?}{\Rightarrow} a_2 \leq a_1 \leq a_1 \leq a_3 \leq a_4$$

- (b) (15 points) Given Q RDPs, solve them in online mode in $O\left(NM + \sum |\mu|\right)$ time. Solving in online mode means you should finish solving the current RDP first before fetching the information of the next RDP. Hint: Consider a graph with N vertices.

You may obtain up to 5 points even if you have only solved the case where $M = 1$.

You may obtain up to 10 points if you propose an algorithm which does solve the problem fails to run in the intended time complexity.

Problem 6 - Picky Roommate (20 points)

MAXIMUM-ROOMMATE-PROBLEM. N people are going to rent a room, where the i -th person has a **set** of bad habits H_i and a **set** of pet peeves P_i . Each element x in H_i and P_i stands for an issue — for example, if issue $x \in H_i$ means person i makes noise in the midnight, then issue $x \in P_j$ may mean person j needs silence in the midnight. Here, we may label the elements in the bad habit set and the pet peeve set with indices so $H_i, P_i \subseteq \{1, 2, \dots, M\}$, $\forall i = 1, 2, \dots, N$. Since each of these people has their own consistent standard, $H_i \cap P_i = \emptyset$, $\forall i = 1, 2, \dots, N$ should be guaranteed. Your goal is to find a maximum subset of $S \subseteq \{1, 2, \dots, N\}$ such that $H_i \cap P_j = \emptyset$, $\forall i, j \in S$.

(a) (3 points) Given $N = 4$ and $M = 4$. The bad habit sets and pet peeve sets are:

- $H_1 = \{3\}, P_1 = \{2, 4\}$
- $H_2 = \{3\}, P_2 = \{4\}$
- $H_3 = \{2, 3\}, P_3 = \{1, 4\}$
- $H_4 = \{1, 3\}, P_4 = \{4\}$

What is the maximum subset S such that $H_i \cap P_j = \emptyset$, $\forall i, j \in S$? If there are multiple possible S 's, you may write any. No partial credits is given for an incorrect answer.

“Finding roommates are extremely hard...”, said -10^{11} . “It is as hard as solving an NP-complete problem.”

“You might be making up a metaphor but it is literally that hard.” Baluteshih replied.

(b) (10 points) Given the fact that CLIQUE-PROBLEM is NP-complete, show that the decision version of MAXIMUM-ROOMMATE-PROBLEM is NP-complete as well. Check the appendix out if you have forgotten what CLIQUE-PROBLEM is.

“In that case, maybe we can come up with a randomized approximation algorithm instead.” proposed -10^{11} .

-10^{11} 's ALGORITHM. First, uniformly randomly pick a permutation p_1, p_2, \dots, p_N among all $N!$ permutations of N people. Let S be an empty set in the very beginning. In the i -th iteration, let's see if $S \cup \{p_i\}$ is valid as an output — if so, add p_i to S (i.e. $S \leftarrow S \cup \{p_i\}$); otherwise, do nothing in this iteration. After N iterations, output S .

-10^{11} 's CLAIM. -10^{11} 's ALGORITHM is a randomized 10-approximation algorithm. That is, if an optimal solution to the MAXIMUM-ROOMMATE-PROBLEM is a set with size C^* and the expected size of the set S output from -10^{11} 's ALGORITHM is C , $\frac{C^*}{C} \leq 10$.

(c) (7 points) Is -10^{11} 's CLAIM correct? If so, prove the correctness; otherwise, construct a counterexample and show that it breaks -10^{11} 's CLAIM.

Problem 7 - Goodbye 2022 Hello 2023 (3 points)

Congratulations! You have finished one of the most difficult required courses in the NTU CSIE. We would like to see some feedback from you in a more specific way. For each subproblem, you will get full credits with any valid response.

- (a) (1 point) What are the two most difficult topics for you in the lecture (listed in order)?
 - (a) Time complexity and asymptotic notations
 - (b) Divide and conquer
 - (c) Dynamic programming
 - (d) Graph
 - (e) Amortized analysis
 - (f) NP-completeness and problem reduction
 - (g) Approximation algorithm
- (b) (1 point) Sort the regular homework according to the difficulty you think, from difficult to easy. Please sort the hand-written part and the programming part separately.
 - (i) Regular Hand-Written Homework 1 \sim 4.
 - (ii) Regular Programming Homework 1 \sim 4.
- (c) (1 point) How frequently do you utilize the TA hours on average?
 - (a) Never
 - (b) Participate in $(0, 1]$ sessions per week
 - (c) Participate in $(1, 2]$ sessions per week
 - (d) Participate in $(2, 3]$ sessions per week
 - (e) Participate in $(3, 4]$ sessions per week
 - (f) Participate in $(4, 5]$ sessions per week
 - (g) Participate in $(5, 6]$ sessions per week
 - (h) Participate in $(6, 7]$ sessions per week
 - (i) Participate in $(7, 8]$ sessions per week
 - (j) Participate in $(8, 9]$ sessions per week
 - (k) Participate in $(9, 10]$ sessions per week
 - (l) Participate in $(10, \infty)$ sessions per week

Thanks again for taking this course — we hope that you have enjoyed a lot.

And most importantly... happy holidays!

Appendix

Hamiltonian-Path-Problem Given an undirected or directed graph $G = (V, E)$, determine whether there is a simple path that visits all the vertices exactly once.

Single-source shortest-path algorithms

- **Bellman-Ford algorithm**

Input a weighted directed graph $G = (V, E)$ and a source vertex s in V . Bellman-Ford algorithm can correctly output the shortest path from s to every vertex in V when there is no negative-weight cycle; or detect the existence of negative cycles. Both in $O(|V||E|)$ time.

- **Dijkstra algorithm**

Input a weighted directed graph $G = (V, E)$ and a source vertex s in V . Dijkstra algorithm can correctly output the shortest path from s to every vertex in V when there is no negative-weight edge. In $O(|E| + |V| \log |V|)$ time.

Clique-Problem A clique in an undirected graph $G = (V, E)$ is a subset of $V' \subseteq V$ in which each pair of the vertices are connected by an edge in E . Given an undirected graph $G = (V, E)$, determine whether the graph contains a clique of k vertices.