CS 3330: Algorithms

Midterm Retake, Fall 2024

Instructions

- You must neither give nor receive aid on this exam. To do otherwise is a **violation of the Honor Code.**
- There are five questions. The exam is out of 120 points. Each question specifies the points it is worth.
- The exam will be graded from 0 to 100; i.e., we will cut off the scores at 100.
- You will have 120 minutes to complete the exam.
- Ask the instructor for extra blank papers if you need any.
- Write your answers carefully. Use complete sentences. Illegible texts may not get points.

Good Luck!

First Name	
Last Name	
HawdkID	

The work in this paper is my own. Signature: ______________________

This page is intentionally left blank.

1. (25 points) Multiple Choice Questions. In each of the following, please select **ALL** options that are true.

(a) (5 points) Let $f(n) = \frac{n^5+n^{4.5}+\sin(n)}{n^2+2}$. Select all that applies:

- ☐ $f(n)$ is $O(n^9)$
- ☐ $f(n)$ is $O(n^{8\sin(n)})$
- ☐ $f(n)$ is $O(n^{-1})$
- ☐ $f(n)$ is $O(n^2)$
- ☐ $f(n)$ is $O(n^8 + \cos(n))$

(b) (5 points) Select all true statements:

- ☐ 2^{n+1} is $O(3^n)$
- ☐ $n^{0.5\sin(n)}$ is $O(\sqrt{n})$
- ☐ $\log(n^{12} + n^5 + n + 1)$ is $O(\log(n^{13} + n^{-1} + 2))$
- ☐ e^{n^2} is $\Theta(e^{n^5})$
- ☐ $\sqrt{n}$ is $O(\log_5(n))$

(c) (5 points) Consider the following statements, where a, b, k, and c are some constants greater than 1. Please select all the true statements.

- ☐ $f(n)$ is $\Omega(g(n))$ implies $g(n)$ is $O(f(n))$
- ☐ If $f(n) = \binom{n}{k}$ and $g(n) = k^n$, then $g(n)$ is $\Omega(f(n))$
- ☐ $n \log n$ is $O(n^2 \log n)$
- ☐ $f(n)/g(n) + g(n)/f(n)$ is $\Theta(\min[f(n), g(n)])$
- ☐ Assuming $0 < a < b$, if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \in [a, b]$, then $f(n)$ is $O(g(n))$.

(d) (5 points) Consider a set of preferences between hospitals and medical students. Assume there are n students and n hospitals. We will use the Gale-Shapley algorithm to find a stable matching between them. We will assume hospitals make proposals unless mentioned otherwise. Select all true statements:

- ☐ Students will always benefit by lying about their preferences.
- ☐ The running time of the gale-shapely algorithm for certain instances could potentially be $\Theta(n^2 \log n)$.
- ☐ Gale-Shapley algorithm avoids unstable pairs by maintaining a list of them.
- ☐ A hospital will always be paired with its best valid partner.
- ☐ A hospital and a student who rank each other the last in their preference list will never be matched.

(e) (5 points) Consider an undirected (unless mentioned otherwise) graph $G = (V, E)$. Suppose that $|V| = n \geq 3$, and that $|E| \geq 1$. Select all true statements:

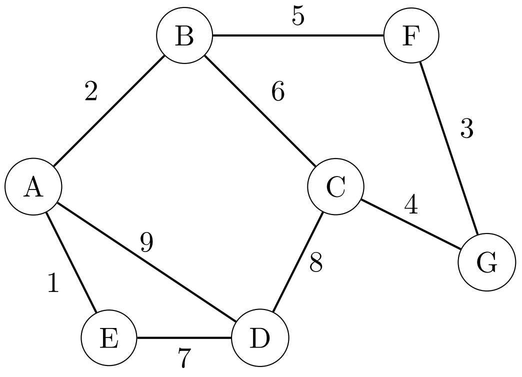
- ☐ If G has at least two leaves (nodes with degree 1), it is a tree.
- ☐ If G has a cycle with at least four nodes, then a cutset with at least four edges exists.
- ☐ Intersections between a spanning tree of V and a cutset of V will always consist of an odd number of edges.
- ☐ Kruskal's algorithm only applies blue rule.
- ☐ Let G be a directed graph. If G is a DAG, it is guaranteed to have a topological sorting even when disconnected.

This page is intentionally left blank.

2. (20 points) Execute the following pseudocode for Kruskal's algorithm and fill in both the tables on the next page after each iteration of the for loop (line 14). Recall that Make-Set, Find-Set, and Union functions are functions of the Union-Find data structure. For the first table, assume you call the Find-Set function over all the vertices at the end of each iteration. Assume that the Find-Set function returns the lexicographically highest vertex in the corresponding set. For example, Find-Set(A), assuming the A is in the set {A, D, E}, will return E.

Algorithm 1 Kruskal's Algorithm

```
1:  **Input:** Graph $G = (V, E)$ with edge costs $c$
2:  **Output:** A minimum spanning tree $T$
3:  Sort edges $E$ by cost in non-decreasing order: $c(e_1) \le c(e_2) \le \cdots \le c(e_m)$
4:  $T \leftarrow \emptyset$
5:  for all vertices $v \in V$ do
6:      Make-Set($v$)                              ▷ Create a new set with element $v$
7:  end for
8:  for $i = 1$ to $m$ do
9:      Let $(u, v) \leftarrow e_i$
10:     if Find-Set($u$) $\neq$ Find-Set($v$) then   ▷ Check if $u$ and $v$ are in the same set.
11:         $T \leftarrow T \cup \{e_i\}$
12:         Union($u, v$)                          ▷ Merge the sets that include $u$ and $v$.
13:     end if
14: end for
15: return $T$
```



Iter.	Find-Set(A)	Find-Set(B)	Find-Set(C)	Find-Set(D)	Find-Set(E)	Find-Set(F)	Find-Set(G)
0	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							
9	G	G	G	G	G	G	G

Iteration	T
0	$\emptyset$
1	
2	
3	
4	
5	
6	
7	
8	

This page is intentionally left blank.

3. (15 points) You are given a set of n items, where each item i has a weight w_i and a value v_i. You are also given a knapsack with a maximum weight capacity W. Your task is to maximize the total value that can be obtained by selecting items to place in the knapsack, such that the total weight does not exceed W.

   In this version of the problem, you are allowed to take fractions of items, meaning you can take any fraction x_i of item i, where $0 \leq x_i \leq 1$. Design a **greedy** algorithm to solve the problem. You can assume the input and output are in the following format.

   Input:

   - An integer n representing the number of items.
   - Two arrays of length n:
     - $v = [v_1, v_2, \ldots, v_n]$ where v_i is the value of the i-th item.
     - $w = [w_1, w_2, \ldots, w_n]$ where w_i is the weight of the i-th item.
   - A positive integer W representing the maximum weight capacity of the knapsack.

   Output:

   - A list of tuples representing the items selected. Each tuple is of the form (i, x_i), where i is the original index of the item and x_i is the fraction of item i selected.

   (a) (7 points) Please write down the **pseudocode**. [Hint: Think of the two extreme cases: i) when the values of all the items are the same and the weights are different, and ii) when the weights of all the items are the same and the values are different.]

(b) (8 points) Use the **greedy-stays-ahead** argument to prove that your algorithm is optimal.

This page is intentionally left blank.

4. (30 points) During a relatively peaceful time among the crime families in Gotham, Penguin is planning to host a gathering for all gangsters. However, certain members of the Falcone family still have lingering resentments against some members of the Maroni family. Other well-known animosities exist between pairs of gangsters. To avoid trouble, Penguin decides to throw two separate parties, inviting some gangsters to the first party and the rest to the second.

   Ideally, every gangster should be invited to exactly one party, and no two enemies should attend the same party. However, it may not always be possible to split the gangsters this way, for instance, if three gangsters are all enemies with each other. Given the list of enemy pairs among the gangsters, your task is to determine whether the Penguin can divide the gangsters into two peaceful parties such that no enemies attend the same party. You have the following information.

   - An integer n representing the number of gangsters.
   - A list of m enemy pairs, where each pair (a, b) indicates that gangster a and gangster b cannot attend the same party.

   Your algorithm should output `YES` if it is possible to divide all the gangsters into two parties without any enemies attending the same party and it should output `NO` otherwise.

   (a) (5 points) Design a naive exponential algorithm that solves the problem. You may write your solution in plain English.

(b) (5 points) Now, let's formulate the problem using graph theory. Describe how you will construct the graph (i.e., define the nodes, edges, etc.).

(c) (10 points) Design a linear time algorithm that uses the graph you constructed to solve the problem. You may write your solution in plain English. **Write down the running time of your algorithm**.

(d) (10 points) Prove the correctness of your algorithm from part (c).

This page is intentionally left blank.

5. (30 points) A new pirate adventure game called **Pirates of the Lost Isles** has been released! In the game, you and your crew sail through seas filled with islands, digging up buried treasures on each one. Your next mission is to explore the **Dead-Eye Islands**, a series of islands connected by treacherous routes, swarming with rival pirates. You are provided with a map of the islands and must hire your crew before embarking on the expedition. Depending on the difficulty level, different numbers of crew members are required for each journey, as you will inevitably lose some along the way. Your task is to devise a linear-time algorithm that determines the number of crew members needed before the expedition begins for each difficulty level. You can assume access to all the algorithms we have covered in class. The map contains information about the available routes between the islands and the number of crew members needed to travel between them. You cannot take any other routes. For this problem, you may describe your algorithm in plain English.

   An example of an older map used in an easy difficulty expedition for **One-Eye Islands** is shown below, where the skeletons on each route denote the number of crew members you will lose if you choose to take that path:

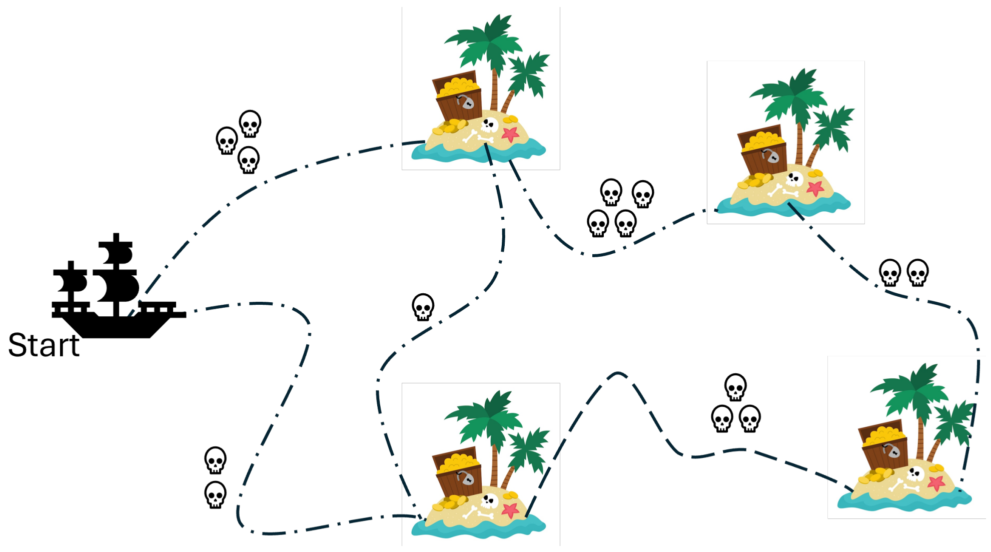


Figure 1: One-Eye Islands - Easy Difficulty

   (a) [5 points] Let's start by modeling the problem. Describe the problem in terms of a graph, specifying what the nodes, edges, and edge weights represent.

(b) [5 points] **Easy Difficulty:** In this difficulty level, each island has a treasure, and after clearing a path between two islands, no new pirates will spawn on that path again. Using the graph from part (a), design a polynomial-time algorithm that minimizes the number of crew members needed to explore all the islands and return to the starting point, given a map with n islands.

(c) [10 points] **Medium Difficulty:** In this difficulty level, the map shows two islands that do not have any treasures, so a trip to those islands may be unnecessary. All the other islands have treasures, and as in the previous case, after clearing a path between two islands, no new pirates will spawn on that path again. Design a polynomial-time algorithm that minimizes the number of crew members needed to gather all the treasures and return to the starting point, given a map with n islands.

(d) [10 points] **Hard Difficulty:** In this difficulty level, all routes have the same number of skeletons. However, the catch is that pirates will respawn after you pass through a route an even number of times, i.e., you will face them again when passing for the 3rd, 5th, and so on. Once again, all islands have treasure in this scenario. Design a polynomial-time algorithm that minimizes the number of crew members needed to gather all the treasures and return to the starting point, given a map with n islands.

This page is intentionally left blank.