Midterm 1

Name: Rhaenyra Targaryen

SID: 0123456789

Name and SID of student to your left: Hightower

Name and SID of student to your right: Velaryon

Exam Room:

Rules and Guidelines

- **The exam is out of 104 points and will last 110 minutes.**
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Write your student ID number in the indicated area on each page.
- Be precise and concise. **Write in the solution box provided.** You may use the blank page on the back for scratch work, but it will not be graded. Box numerical final answers.
- The problems may **not** necessarily follow the order of increasing difficulty. ***Avoid getting stuck on a problem.***
- Any algorithm covered in lecture can be used as a blackbox. Algorithms from homework need to be accompanied by a proof or justification as specified in the problem.
- Throughout this exam (both in the questions and in your answers), we will use ω_n to denote the first n^{th} root of unity, i.e., $\omega_n = e^{2\pi i/n}$.
- You may assume that comparison of integers or real numbers, and addition, subtraction, multiplication and division of integers or real or complex numbers, require $O(1)$ time.
- Good luck!

Discussion Section

Which section(s) do you attend? Feel free to choose multiple, or to select the last option if you do not attend a section. **Please color the checkbox completely. Do not just tick or cross the boxes.**

☐ Kevin Zhu, Wednesday 12 - 1 pm, Wheeler 200

☐ Andrew Che, Wednesday 1-2 pm, Dwinelle 79

☐ Kevin Li and Param (Exam Prep), Wednesday 1-2 pm, Wheeler 224

☐ Adnaan Sachidanandan, Wednesday 2-3 pm, Wheeler 108

☐ Wilson Wu, Wednesday 2-3 pm, Hearst Memorial Gym 242

☐ Cindy Zhang, Wednesday 3-4 pm, Cory 289

☐ Tyler Hou (Leetcode), Wednesday 4-5 pm, Etcheverry 3109

☐ Elicia Ye, Thursday 11-12 pm, Wheeler 130

☐ Cindy Zhang, Thursday 12-1pm, Remote

☐ Reena Yuan, Thursday 1-2pm, Etcheverry 3113

☐ Tynan Sigg, Thursday 4-5 pm, Soda 310

☐ Adnaan Sachidanandan (LOST), Thursday 5-7, Cory 258

☐ Video walkthroughs

☐ Don’t attend Section

1 Asymptotic Analysis (4 pts)

For each pair of functions f and g, specify whether $f = O(g)$, $g = O(f)$, or both. Write "YES" or "NO" in the boxes.

f	g	$f = O(g)$	$g = O(f)$
$n^2 + 5n$	$1000(n+1)^2$		
n^3	$5n^3 + (\log n)^{10}$		
n^{100}	$(1.01)^n$		
$(\log n)^5 + 7 \log n$	$\sqrt{n}$		

2 Recurrences (6 pts)

1. For each recurrence, provide the tightest big O bound that you can.
   (a) $T(n) = 256 * T(n/2) + O(n^2)$
   (b) $T(n) = T(n-1) + O(n^2)$
2. Suppose $T(n)$ satisfies the recurrence: $T(n) = T(n-1) + T(n-3) + 1$, then mark each of the following statements as true or false.
   (a) $T(n) = O(n^{1000})$
   ○ True ○ False
   (b) $T(n) = O(n^{\log n})$
   ○ True ○ False

3 Huffman Encoding (5 pts)

Let g be a string that is made of $6n$ "A"s, $2n$ "B"s, n "C"s and n "D"s.

Suppose we use Huffman encoding to express g in bits.

1. What would the length of the encoding for each letter be?

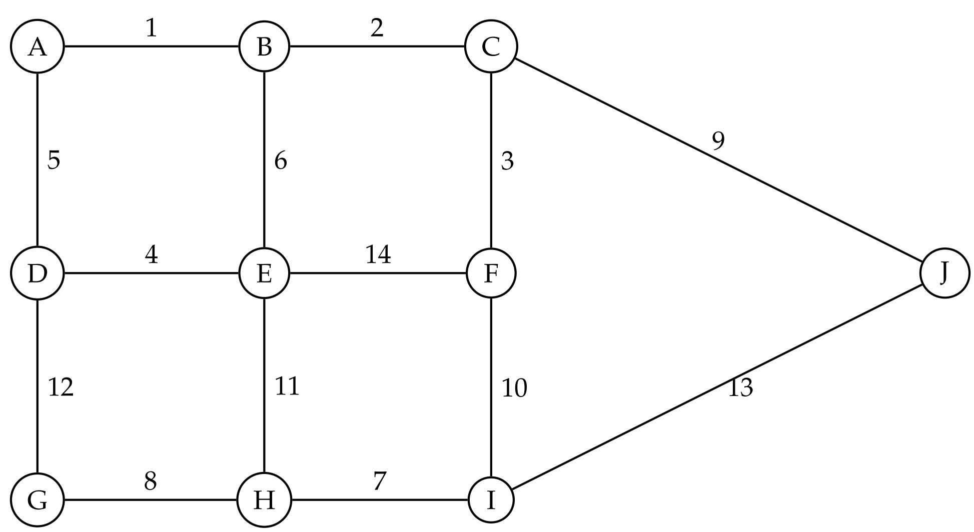
   A: $\boxed{}$ B: $\boxed{}$ C: $\boxed{}$ D: $\boxed{}$

2. What would be the total length of the encoding for g? $\boxed{}$

4 Minimum Spanning Tree (4 pts)

1. List the first six edges added by Prim's algorithm in the order in which they are added. **Start Prim's algorithm from vertex H.**

2. List the first six edges added by Kruskal's algorithm in the order in which they are added.



5 Polynomial multiplication via FFT (6 pts)

Suppose we want to use FFT to multiply two polynomials $P(x)$, $Q(x)$ of degree 40 and 80 respectively.

What would be the size n of the FFT used? (Here n is a power of 2.)

Suppose ω denotes the corresponding root of unity. What is ω^{64}? (It is a very simple complex number.)

In the above polynomial multiplication algorithm via FFT, how many times do you run the FFT algorithm? (Make no assumptions about the algorithm's implementation.)

In the above polynomial multiplication algorithm via FFT, how many times do you run the inverse FFT algorithm? (Make no assumptions about the algorithm's implementation.)

6 Median (4 pts)

Suppose we used the randomized median finding algorithm (i.e Quickselect) to find the median of the following list:

$$\{1, 2, 3, 4, 5, \ldots, 99, 100, 101\}$$

(Note: the list is of length 101 and contains all integers from 1 to 101)

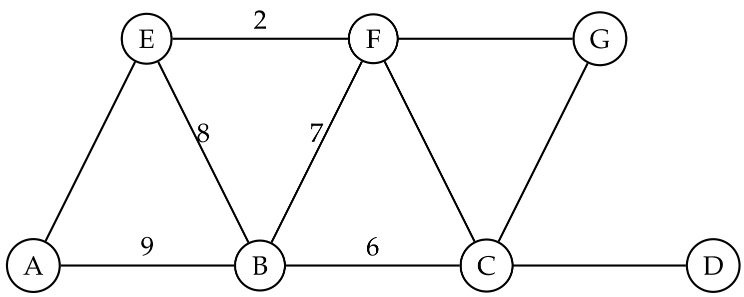
with successive pivot choices $70, 30, 47, 51$.

Write down the length of the relevant sublist and corresponding value of k in each recursive call after performing the partition with respect to the pivot.

Pivot	Length of list	k
70		
30		
47		
51		

7 Minimum Spanning Tree (Reprise) (10 pts)

Here is a weighted graph G. Some of the edge weights are revealed, while others are not.



1. Find all edges that are necessarily part of every MST of the graph and justify.

(a) Edge [ ] is necessarily part of every MST because:

(b) Edge [ ] is necessarily part of every MST because:

(c) Edge [ ] is necessarily part of every MST because:

(Note: there may be more spaces allotted than there are edges. In the case that you think there are less than 3 edges that satisfy the conditions, leave the other parts blank)

2. List all edges that are NOT part of any MST of the graph (no justification needed).

8 Interpreting Pre/Post Values (8 pts)

DFS was run on an undirected connected graph of 8 vertices and m edges and the pre/post values were recorded.

The order of pre/post values of the 8 vertices $\{A, B, C, D, E, F, G, H\}$ is as follows.

$$pre[A] < pre[D] < pre[c] < pre[G] < post[G] < post[C] < post[C] < post[D] < pre[F] < post[F] < \\ < pre[E] < pre[H] < post[H] < post[E] < post[A]$$

or equivalently, in brackets:

$$[_{A}\ [_{D}\ [_{C}\ [_{G}\ ]_{G}\ ]_{C}\ ]_{D}\quad [_{F}\ ]_{F}\quad [_{E}\ [_{H}\ ]_{H}\ ]_{E}\ ]_{A}$$

1. Deleting F from the graph will separate D and E into two different connected components.

   ◯ True ◯ False

2. Deleting E will separate G and H into two different connected components.

   ◯ True ◯ False

3. Deleting A will separate G and H into two different connected components.

   ◯ True ◯ False

4. An edge in the graph is ”critical”, if deleting it disconnects the graph. List all pairs of vertices that necessarily are critical edges of the graph:

9 Inequalities (20 pts)

Given a list of n variables $x_1, \ldots, x_n$ and m inequalities of the form $x_i < x_j$ or $x_i \leq x_j$ for some $i, j \in [n]$, you would like to find values for the variables such that all inequalities are satisfied, or determine that not all inqualities can be satisfied simultaneously.

1. Design an efficient algorithm for the case that all inequalities are strict (i.e. of the form $x_i < x_j$). Give a succinct and precise description of the algorithm (proof of correctness or runtime analysis are **not** needed).

2. Design an efficient algorithm that solves the above problem in general (when some inequalities are of the form $x_i < x_j$, and some are of the form $x_i \leq x_j$).

   Give a succinct and precise description of the algorithm (proof of correctness or runtime analysis are **not** needed).

10 Most Probable Value (12 pts)

There's a device that generates a random positive integer between 1 and n.

For $1 \leq i \leq n$, let p_i denote the probability that the device generates the number i.

Devise an algorithm to find the most probable value of the sum of 4 independent samples from the device (Your algorithm should run in time less than or equal to $O(n^2)$.)

Give a succinct and precise description of your algorithm (proof of correctness and runtime analysis are **not** needed).

Hint:

$$\Pr[X_1 + X_2 + X_3 + X_4 = n] = \sum_{x_1+x_2+x_3+x_4=n} \Pr[X_1 = x_1, \ldots, X_4 = x_4]$$

11 Red Blue Edges (20 pts)

The Story: (*Feel free to skip the story if you prefer a formal problem description.*)

There is both a rail network and a bus network on the same set of n cities. Given a start city s and a destination city t, can we find the shortest path from s to t that uses exactly k bus rides and k train rides (in any order)?

Formal Problem Description:

Input:

1. Two undirected graphs $G_{red} = (V, E_{red})$ and $G_{blue} = (V, E_{blue})$ on the same set of vertices V. We refer to edges E_{red} as *red* edges and edges E_{blue} as *blue* edges.
2. Length ℓ_e for each edge $e \in E_{red} \cup E_{blue}$. (All edge lengths are positive)
3. Two vertices s and t.
4. Positive integer k.

Goal: Find the shortest path from s to t that contains exactly k red edges and k blue edges, in any order.

In other words, among all paths from s to t that has k red edges and k blue edges, find the shortest. If there is no path from s to t that has k red edges and k blue edges, then your algorithm should return FAIL.

(A path is permitted to go through the same edge multiple times.)

Devise an algorithm for the problem that runs in time less than or equal to $O(k^4 \cdot |V|^2)$. Give a succinct and precise description of your algorithm. (Proof of correctness and runtime analysis are not needed).

12 Bellman-Ford (5 pts)

(Please refer to the Bellman-Ford pseudocode from the textbook for reference on the following page.)

Suppose we run the Bellman-Ford algorithm on the following graph to compute shortest paths from the vertex A.

Let T denote the number of individual "update" calls after which the value of $dist[J]$ reaches its correct value (which may occur prior to the algorithm's termination). T depends on the actual weights on the edges and the order in which the edges are updated.

Assume there's a fixed ordering of the edges. That is, we always go through the edges in the same order every time we run `for all e in E`.

(**Both the weights and order of edge updates are NOT given.**)

1. What is the smallest possible value of T?

2. What is the largest possible value of T?

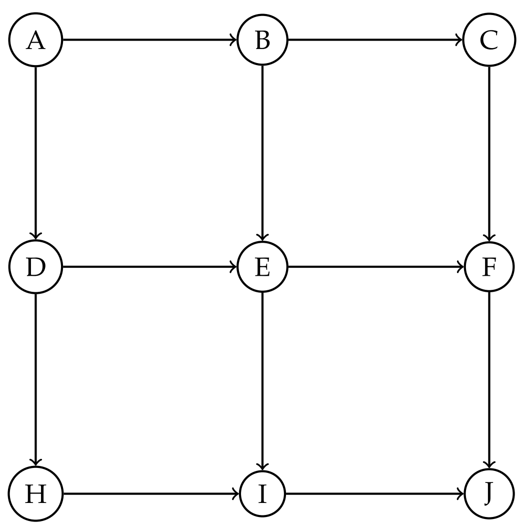


Figure 4.13 The Bellman-Ford algorithm for single-source shortest paths in general graphs.

```
procedure shortest-paths(G, l, s)
Input:    Directed graph G = (V, E);
          edge lengths {l_e : e ∈ E} with no negative cycles;
          vertex s ∈ V
Output:   For all vertices u reachable from s, dist(u) is set
          to the distance from s to u.

for all u ∈ V:
   dist(u) = ∞
   prev(u) = nil

dist(s) = 0
repeat |V| − 1 times:
   for all e ∈ E:
      update(e)
```

Blank scratch page.