Final

Name:

SID:

Name and SID of student to your left:

Name and SID of student to your right:

Exam Room:

Rules and Guidelines

- **The exam will last 170 minutes.**
- The exam has 183 points in total.
- 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!

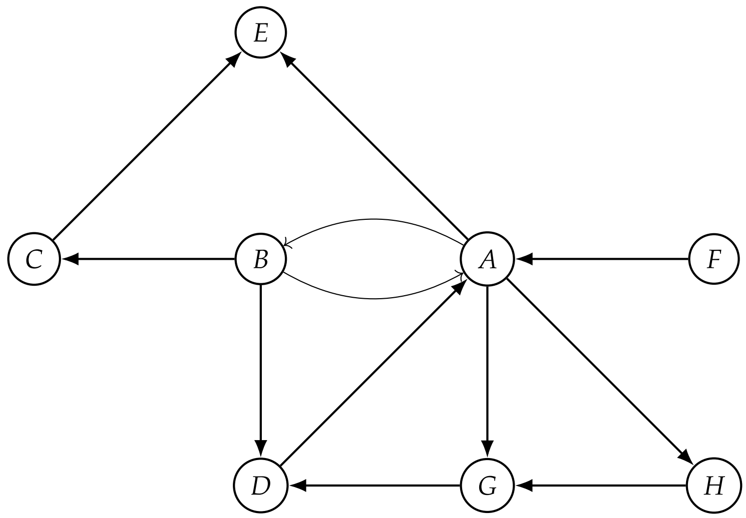
Discussion Section

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 Search tree (8 points)

Suppose we run depth-first search on the following graph, breaking ties alphabetically.



1. Draw the DFS search tree.

2. For each of these edges, indicate whether they are tree edges, forward edges, back edges or cross edges.

$A \to B$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$B \to A$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$B \to C$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$B \to D$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$D \to A$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$C \to E$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$A \to E$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$F \to A$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$G \to D$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$H \to G$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$A \to G$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge
$A \to H$	◯ Tree/Foward Edge ◯ Back Edge ◯ Cross Edge

2 Runtime Analysis (5 points)

Consider the following snippet of code.

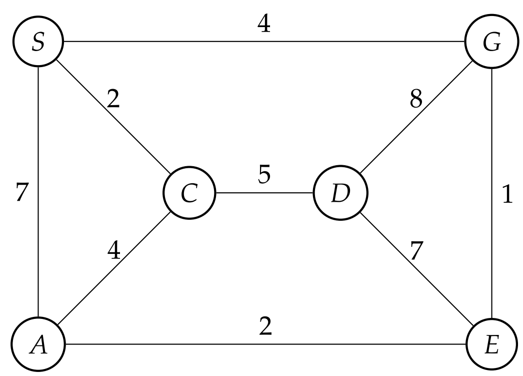
```
Function what(n) {

    if (n < 1) return

    for i = 1 to n {
        for j = 1 to n/2 {
            for k = 1 to n/4 {
                print BLAH
            }
        }
    }

    what(n/2)
    what(n/2)
    what(n/2)
    what(n/2)

}
```

Let $F(n)$ denote the number of "BLAH"s printed by a call to $what(n)$. Write a recurrence relation for $F(n)$.

Write the tightest upper bound for $F(n)$ (a bound that is asymptotically tight, up to constant factors).

3 Djikstra's algorithm (5 points)

Recall that Djikstra's algorithm for shortest paths maintains a priority queue. Let H denote the priority queue that is maintained by Djikstra's algorithm starting at vertex S in the graph shown below.



Initially, every vertex other than S is inserted in the queue with a key value of ∞. Then, Djikstra's algorithm performs a sequence of $deleteMin(H)$ and $decreaseKey(H, v)$ operations on the queue H, where v denotes some vertex.

List all the $decreaseKey(H, \_)$ operations performed during the execution of Djikstra on the following graph starting at node S. (Note that there may be more spaces provided below than the number of *decreaseKey* operations executed, leave the additional spaces blank)

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

$decreaseKey\Big(H, \boxed{\quad\quad}\Big)$

4 Strongly Connected Components (8 points)

Suppose we execute DFS on a directed graph G and compute *pre* and *post* values for each node. Let

$$u \leftarrow \text{ vertex with smallest pre value}$$
$$v \leftarrow \text{ vertex with largest post value}$$

1. The vertex with the smallest *pre* value is
   (A) necessarily part of a source SCC.
   (B) necessarily part of a sink SCC.
   (C) none of the above.

   ○A ○B ○C

2. The vertex with the largest *post* value is
   (A) necessarily part of a source SCC.
   (B) necessarily part of a sink SCC.
   (C) none of the above.

   ○A ○B ○C

3. If the graph G is strongly connected then
   (A) there is necessarily an edge from $u \to v$.
   (B) there is necessarily an edge from $v \to u$.
   (C) vertex u is the same as vertex v.
   (D) none of the above.

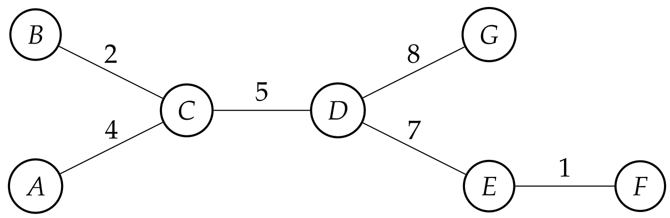
   ○A ○B ○C ○D

4. The vertex with the largest *pre* value
   (A) is necessarily in a sink SCC.
   (B) is necessarily not in a source SCC.
   (C) none of the above.

   ○A ○B ○C

5 Minimum Spanning Tree (8 points)

The tree shown below is a minimum spanning tree in a graph **H**. The remaining edges of the graph **H** are NOT shown in the picture.



1. There is an edge AG in the graph (not shown in the picture). The smallest possible value for the weight of edge AG is
2. The cost of the MST shown above is $2 + 5 + 8 + 4 + 7 + 1 = 27$. A new edge AE with weight 5 was added to the graph. The cost of the minimum spanning tree in the new graph is
3. In this class, we studied two algorithms for MST: Kruskal's and Prim's algorithm.

   We executed one of these two algorithms (Kruskal's/Prim's) on the graph **H** and it produced the tree shown above. In first three iterations, the algorithm added edges CD, BC, AC to the tree. The next edge added by the algorithm is
4. In this class, we studied two algorithms for MST: Kruskal's and Prim's algorithm.

   We executed one of these two algorithms (Kruskal's/Prim's) on the graph **H** and it produced the tree shown above. In first three iterations, the algorithm added edges EF, BC, AC to the tree. The next edge added by the algorithm is

6 TSP tour (4 points)

Suppose there are n vertices with distances d_{ij} between vertices i and j. Assume that the distances d_{ij} satisfy the triangle inequality.

1. If the cost of the minimum travelling salesman tour is C, then the cost of the minimum spanning tree is at most ______
2. If there is a spanning tree T (not necessarily the minimum spanning tree) of cost W, then the cost of the minimum travelling salesman tour is at most ______

7 NP-completness true/false (15 points)

For each of the following questions, there are four options:

(1) True (T); (2) False (F); (3) True if and only if $\mathbf{P} = \mathbf{NP}$; (4) True if and only if $\mathbf{P} \neq \mathbf{NP}$.

Circle one for each question.

Note: By "reduction" in this exam it is always meant "polynomial-time reduction with one call to the problem being reduced to."

1. There is a polynomial-time reduction from Integer Programming to Circuit-SAT.

   ○ T ○ F ○ $\mathbf{P} = \mathbf{NP}$ ○ $\mathbf{P} \neq \mathbf{NP}$

2. There is a polynomial-time reduction from Circuit-SAT to Integer Programming.

   ○ T ○ F ○ $\mathbf{P} = \mathbf{NP}$ ○ $\mathbf{P} \neq \mathbf{NP}$

3. There is a polynomial-time reduction from Minimum-Spanning Tree to Integer Programming.

   ○ T ○ F ○ $\mathbf{P} = \mathbf{NP}$ ○ $\mathbf{P} \neq \mathbf{NP}$

4. If there is a polynomial time algorithm for one problem in NP, there is one for all of them.

   ○ T ○ F ○ $\mathbf{P} = \mathbf{NP}$ ○ $\mathbf{P} \neq \mathbf{NP}$

5. The Longest Increasing Subsequence problem is NP-complete.

   ○ T ○ F ○ $\mathbf{P} = \mathbf{NP}$ ○ $\mathbf{P} \neq \mathbf{NP}$

8 Edit Distance (10 points)

Given two strings $x[1,\ldots,n]$ and $y[1,\ldots,m]$, recall that the edit distance is the smallest number of keystrokes needed to edit the string x into string y. Here each insertion and deletion of a character takes one key stroke.

We devised a dynamic programming algorithm for the problem wherein the subproblems are

$$ED[i,j] = \text{ minimum number of keystrokes needed to edit the prefix } x[1,\ldots,i] \text{ into the prefix } y[1,\ldots,j].$$

Suppose we had a special key-board in which

- each insertion takes 2 keystrokes,
- each deletion takes 3 keystrokes, and
- each substitution takes 4 keystrokes.

1. Write down the modified recurrence relation for $ED[i.j]$.

2. Write down the modified base cases for $ED[i,j]$.

9 Fill in the Blanks (24 points)

1. A directed acyclic graph (DAG) on n vertices can have at most $\boxed{}$ number of edges

2. A strongly connected directed graph on n vertices must have at least $\boxed{}$ number of edges.

3. Let G be a graph on n nodes. The dynamic programming based algorithm for All-Pairs-Shortest-Paths (also known as the *Floyd-Warshall algorithm*) on G has $\boxed{}$ many subproblems.

   Moreover, the recurrence relation expresses the value of each subproblem in terms of $\boxed{}$ many other subproblems.

4. Suppose a hash function $h : \{1, \ldots, n\} \to \{1, \ldots, 2n\}$ is drawn from a universal hash family. Then

   $$\Pr[h(2) = h(1)] = \boxed{}.$$

   and

   $$\Pr[h(2) > h(1)] = \boxed{}.$$

5. If ω is a primitive 16^{th} root of unity then ω^8 is a k^{th} root of unity for $k = \boxed{}$. (Write the smallest possible value of k.)

6. Suppose we execute the reservoir sampling algorithm on a stream $s_1, \ldots, s_m$. What is the probability that the reservoir contains s_2 at the end of 10th iteration (i.e., after seeing $s_1, \ldots, s_{10}$)?

   $\boxed{}$ What is the probability that every element of the stream was in the reservoir at some point during the execution of the algorithm?

   $\boxed{}$

7. For each of the following quantum states, write down the probabilities that we observe 0 and 1 when we perform a measurement on them.

State	Pr[outcome is 0]	Pr[outcome is 1]
$\|0\rangle$		
$\sqrt{\frac{1}{3}} \cdot \|0\rangle - \sqrt{\frac{2}{3}} \cdot \|1\rangle$		

8. Recall the quantum operation Rotate$(\cdot, \cdot)$ that we saw in class. Let $|-\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle$.
   Supposing that we perform the operation Rotate$(\pi/4, |-\rangle)$ and then measure the resulting state, the probability of outcome 0 is

10 True/False (6 points)

1. Let $G = (V, E)$ be a graph with distinct positive edge weights $\{w_e \mid e \in E\}$. Construct a set of edges E_{light} as follows: for each vertex $v \in V$, include the lightest edge incident to v in E_{light}. Then the edges in E_{light} form a Minimum Spanning Tree of G.

   ○ True ○ False

2. A linear program cannot have exactly 2 distinct optimal solutions.

   ○ True ○ False

3. There are connected, undirected graphs G such that decreasing the capacity of any single edge in G does not change the value of the maximum flow.

   ○ True ○ False

4. There are connected, undirected graphs G such that increasing the capacity of any single edge in G does not change the value of the maximum flow.

   ○ True ○ False

5. There are directed graphs G such that decreasing the capacity of any single edge in G does not change the value of the maximum flow.

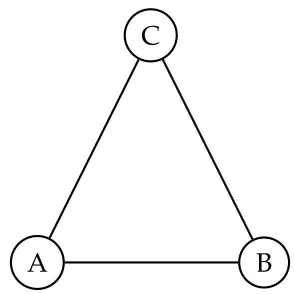
   ○ True ○ False

6. Quantum computers are believed to be able to efficiently solve **NP**-complete problems.

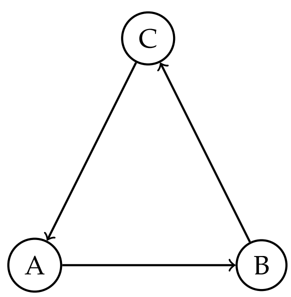
   ○ True ○ False

11 One-Way Streets (15 points)

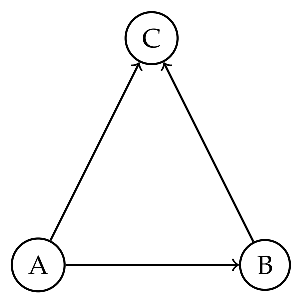
Suppose we have an undirected connected graph, and we would like to find a strongly connected orientiation of its edges: that is, an assignment of directions to its edges such that the entire graph becomes strongly connected (i.e. every node is reachable from every other node via a directed path). For the undirected connected graph below, the next figure is a strongly connected orientation, and the last is an orientation that is not strongly connected.



(a) An undirected graph



(b) A strongly connected orientation



(c) An orientation which is not strongly connected

1. Give an example of a connected undirected graph where this cannot be done.

2. Call an edge *lonely* if it is NOT part of any cycle. If an undirected graph G has a strongly connected orientation, then it can have at most ☐ many lonely edges.

3. Suppose e is a lonely edge in a graph G. In a DFS search tree of G, what kind of edge will e be? ☐

4. Let G be a connected undirected graph with no *lonely* edges. We executed a DFS on G and compute the $pre[v]$ and $post[v]$ values for every node v. How would you find a strongly connected orientation of the graph G from its pre and post values?

   Specifically, fill in the blanks below.

The following procedure will produce a strongly connected orientation if there exists one:

For each edge (u, v) in the graph G,

(a) Direct the edge $u \to v$ if $pre[u], pre[v], post[u], post[v]$ satisfy the condition

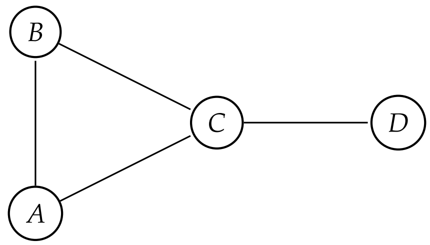
(b) Else direct the edge $v \to u$.

---

5. Argue that the above algorithm produces a strongly connected orientation if there exists one.

12 Power Stations (15 points)

The Frugal Inc. company runs four electric car charging stations A, B, C, D connected by the network of roads as shown below.



Frugal Inc. would like to compute the optimal schedule for keeping these charging stations open. Here are the details.

1. No charging station can be open for more than 18 hours in a day.
2. Each charging station costs a certain amount per hour to keep open. The costs are given in the following table.

Charging Station	Cost per hour
A	1
B	2
C	3
D	4

3. At any time in the day, for every road, at least one of the charging stations at its end points must be open.

The charging stations are allowed to be open for fractional (non-integer) number of hours.

12.1 A failed attempt

Frugal Inc. tried to write a linear program to determine the minimum total cost per day of running these charging stations while satisfying all the constraints.

Frugal Inc. started with the following choice of variables:

$$x_A = \text{number of hours station A is open}$$
$$x_B = \text{number of hours station B is open}$$
$$x_C = \text{number of hours station C is open}$$
$$x_D = \text{number of hours station D is open}$$

The constraints were:

- *"No charging station can be open for more than* 18 *hours a day"*

$$x_A, x_B, x_C, x_D \leq 18$$

- *At any time in the day, for every road, at least one of the charging stations at its end points must be open.*

- Road AB: $x_A + x_B \geq 24$
- Road BC: $x_B + x_C \geq 24$
- Road CA: $x_C + x_A \geq 24$
- Road CD: $x_C + x_D \geq 24$

Unfortunately, this linear program failed. Frugal Inc. solved the linear program to get a feasible solution $x_A = x_B = x_C = x_D = 12$. Argue that it is impossible to create a valid schedule with $x_A = x_B = x_C = x_D = 12$.

12.2 Correct Algorithm

Write a linear program that correctly computes the minimum total cost of running the 4 charging stations per day. (**Hint:** An entirely different choice of variables is needed. Observe that there are 7 permitted open/close states for the stations.)

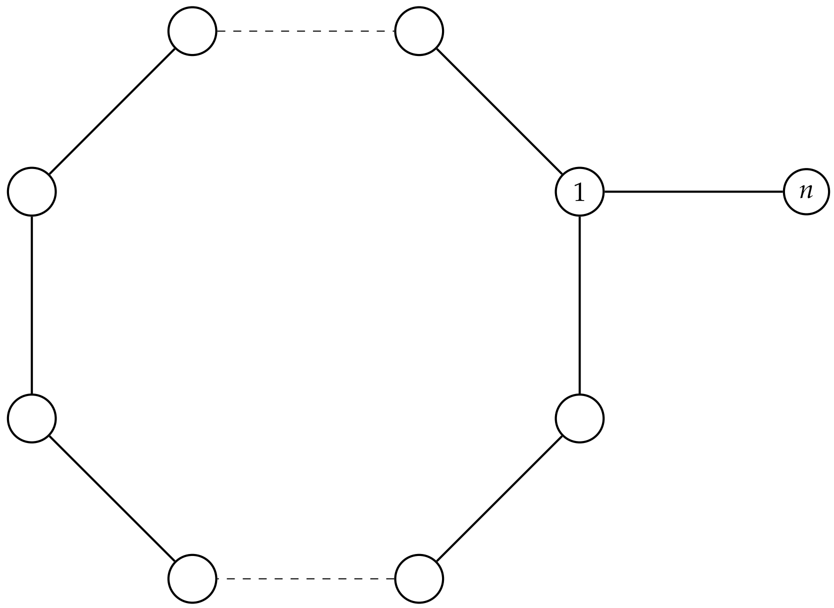
1. List all the variables of your linear program, and describe what each of them is intended to mean.

2. What are the constraints of the LP?

3. What is the objective function being minimized?

13 Karger's algorithm (5 points)

Consider the following undirected graph $G = (V, E)$ on n vertices.



This graph consists of an $(n-1)$-cycle involving the vertices 1 through $n-1$, which contains the edges $(1,2)$, $(2,3)$, $\ldots$, $(n-2, n-1)$, and $(n-1, 1)$. (The dashed lines in the picture represent vertices and edges from this cycle which are not drawn.) In addition, the graph has another vertex, vertex n, and an edge between vertices 1 and n.

1. What is minimum cut in this graph? What is the size of the minimum cut?

2. Suppose we run Karger's algorithm on this graph. What is the probability that it outputs the minimum cut? (Note: this probability may or may not be equal to the lower bound on the success probability of Karger's algorithm that we proved in class.) Explain your reasoning.

14 Longest k-modal subsequence (15 points)

A sequence of integers $a_1, \ldots, a_\ell$ is *k-modal* if it starts off increasing and switches between increasing and decreasing at most k times. For example, a 0-modal sequence is just an increasing sequence; a 1-modal sequence is one that increases until it reaches a maximum value, and then it decreases.

Formally, a sequence is k-modal if there exists "switch points" $i_1, \ldots, i_k$ such that $a_1 < a_2 < \ldots < a_{i_1}$ and $a_{i_1} > a_{i_1+1} > \ldots > a_{i_2}$, and so on. For example, 1, 2, 3, 5, 8, 13, 11, 7, 5, 3, 2 is 1-modal.

In class, we saw an $O(n^2)$-time dynamic programming algorithm for computing the longest increasing subsequence of a sequence of numbers $a = (a_1, \ldots, a_n)$. In this problem, devise a dynamic programming based algorithm for computing the length of the longest *k-modal* subsequence given k and an input sequence $a_1, \ldots, a_n$.

1. What are the subproblems? (precise and succinct definition needed)

2. What are the recurrence relations?

15 Reductions (10 points)

Consider the following two problems:

1. **Matching**
   Input: Graph $G = (V, E)$ and a positive integer k
   Solution: A set of k edges $e_1, \ldots, e_k$, no pair of which share a vertex.
2. **Independent Set**
   Input: Graph $G' = (V', E')$ and a positive integer k
   Solution: A set of k vertices $S \subset V'$, no pair of which are connected by an edge.

Among the above two problems, one of them reduces to the other but not vice-versa. Fill in the blanks below (with answers in the context of above two problems).

1. The problem ____ is believed to not reduce in polynomial time to ____ because

2. On the other hand, ____ reduces in polynomial time to ____
   Here is the reduction.

   Reduction: *Given an instance Φ of the problem* ____ *we will construct an instance Ψ of the problem* ____ *as follows.*

The proof that this is a valid reduction is omitted here

16 NP-Completeness Reductions (15 points)

Show that the following problems are NP-complete by providing a polynomial-time reduction. You may assume that the following problems are NP-complete: Rudrata (Hamiltonian) Path, Rudrata (Hamiltonian) Cycle, Vertex Cover, Independent Set, 3-SAT and Integer Programming.

1. **Hitting Set**

   Input: Positive integers n, m, k and a family of subsets $S_1, \ldots, S_n$ of $\{1, \ldots, m\}$.

   Solution: A subset $H \subset \{1, \ldots .m\}$ of size k that overlaps every one of the sets $S_1, \ldots, S_n$, i.e., for each S_i, we have $S_i \cap H \neq \emptyset$.

   Proof. It is clear that the Hitting Set problem is in NP. Now we will use a polynomial time reduction to show that the problem is indeed NP-complete.

   Given an instance Φ of the problem ______ *we will construct an instance Ψ of the problem* ______ *as follows.*

   The proof that this is a valid reduction is as follows:

17 Good Pivot (10 points)

Given an array of numbers $a_1, \ldots, a_n$ (NOT sorted), a *good pivot* is a number b such that between 25% to 75% of the numbers in the array are less than b.

Formally if we define $S_b = \{i | a_i < b\}$ then a good pivot satisfies $\frac{n}{4} \leq |S_b| \leq \frac{3n}{4}$.

A good pivot b does not necessarily have to be an element of the array.

1. Devise a randomized algorithm that will output a number b such that b is a good pivot with probability at least $1 - 10^{-6}$. With probability 10^{-6}, the algorithm may output a number that is not a good pivot.

   Give a succinct and precise description of your algorithm. (For full credit, the runtime of your algorithm must be smaller than $\log^2 n$)

2. What is the asymptotic run-time of your algorithm?

18 Minimum Median Tree (5 points)

Design an efficient algorithm for the following problem:

Input: A graph $G = (V, E)$ and weights on the edges $\{w_e \mid e \in E\}$.

Solution: A spanning tree T that minimizes the $medianCost(T)$, where $medianCost(T)$ is defined as

$$medianCost(T) = \text{ median of the weights of the edges in the tree } T.$$

The runtime of your algorithm must be polynomial in the number of vertices $n = |V|$.

1. Give a succinct and precise description of your algorithm.

2. Prove the correctness of your algorithm.

Blank scratch page.

Blank scratch page.