This is a closed-book, closed-notes, closed-door, open-brain exam. If you brought anything with you besides writing instruments and your $8.5'' \times 11''$ cheat sheet, please leave it at the front of the classroom.

If your cheat sheet is not handwritten by yourself, or it is photocopied, please do not use it and leave it in front of the classroom.

Submit your cheat sheet together with your exam. An exam without your cheat sheet attached to it will not be checked.

Print your name, netid, and alias in the boxes above. Circle U if you are an undergrad, or G if you are a grad student. Print your name at the top of every page (in case the staple falls out!).

Undergraduate and graduate students should answer all the questions in this exam.

Unless we specifically say otherwise, proofs are not required.

Write in a clear and readable handwriting. Your answers should be as short as possible.

The last few pages of this booklet are blank. If you run out of room for an answer, you can put it there, but please tell us where to look! If you prefer, you can just tear the pages off and use them as scratch paper. Please let us know if you need more paper.

Time limit: 180 minutes.

Relax. Breathe. This is just a stupid exam in an unimportant course, taken in an insignificant state, in an unimportant country, located on a minor planet, in a tiny galaxy in a vast and infinite universe.
1. [20 Points] Multiple Choice:
   For each question, write the letter that corresponds to your answer. You do not need to justify your answers. Each correct answer earns you 1 point.
   In the following, the questions always refer to the worst-case scenario. Furthermore, if you are asked about the running time of an algorithm, you have to provide the answer for the fastest algorithm possible.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Θ(1)</td>
<td>G</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>B</td>
<td>Θ(log n)</td>
<td>H</td>
<td>Θ(n + m)</td>
</tr>
<tr>
<td>C</td>
<td>Θ((log n) log log n)</td>
<td>I</td>
<td>Θ(n log* n)</td>
</tr>
<tr>
<td>D</td>
<td>Θ(log^2 n)</td>
<td>J</td>
<td>Θ(n log log n)</td>
</tr>
<tr>
<td>E</td>
<td>Θ(√n)</td>
<td>K</td>
<td>Θ(n log n)</td>
</tr>
<tr>
<td>F</td>
<td>Θ((√n + m) log* n)</td>
<td>L</td>
<td>Θ(n log n + m)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M</td>
<td>Θ((n + m) log n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>Θ(n log^2 n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O</td>
<td>Θ(n^2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P</td>
<td>Θ(m log n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q</td>
<td>Θ(nm)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>Θ(nm^2)</td>
</tr>
</tbody>
</table>

1. What is the solution to the recurrence \( T(n) = \log n + T(\sqrt{n}) \)?

2. What is the running time of \texttt{WeirdEuclid}(a, b) for \( a, b \leq n \)?

   \texttt{WeirdEuclid}(a, b):
   
   if \( b = 0 \)
   
   return \( a \)
   
   if \( a \) is even and \( b \) is even
   
   return \( 2 \times \texttt{WeirdEuclid}(a/2, b/2) \)
   
   if \( a \) is even and \( b \) is odd
   
   return \( \texttt{WeirdEuclid}(a/2, b) \)
   
   if \( a \) is odd and \( b \) is even
   
   return \( \texttt{WeirdEuclid}(a, b/2) \)
   
   if \( b > a \)
   
   return \( \texttt{WeirdEuclid}(b - a, a) \)
   
   else
   
   return \( \texttt{WeirdEuclid}(a - b, b) \)

3. Let \( a_0, \ldots, a_n \) be \( n \) given numbers inside an array. How fast can one compute \( f(n) \), where \( f(n) = a_n + \max_{i=1}^{\log n} f\left(\left\lceil \frac{n}{2^i} \right\rceil\right) \)?

4. Given two treaps with \( n \) elements overall, how fast can one merge them into a single treap, if all the elements in the first treap are large than all the elements in the second treap?
5. Let assume that we start with a union-find data-structure which contains \( n \) singletons, and we perform a sequence of \( n \) union operations, and then a sequence of \( n \) find operations. What is the overall time this sequence of operations required?

6. Let \( G \) be a graph (with \( n \) vertices and \( m \) edges) with weights on the edges. Assume that all the weights are distinct. You are given two different spanning trees \( T_1, T_2 \) of \( G \). How fast can one determine which one of them is not a minimum spanning tree?

7. Let \( G \) be a graph (with \( n \) vertices and \( m \) edges) with positive weights on the edges. Assume that all the weights are distinct. You are given a tree \( T \) of \( G \) which claims to be the shortest-path tree from a vertex \( s \in V(G) \). How fast can one determine whether \( T \) is indeed the shortest-path tree it claims to be?

8. Given two convex polygons \( C_1, C_2 \) that have both \( n \) vertices. Let \( P_1, P_2 \) denote their corresponding set of points. How fast can one compute the convex hull of \( P_1 \cup P_2 \)?

9. Given a simple not self-intersecting polygon. How fast can one decide if it is convex or not?

10. Given two polynomials \( p(x), q(x) \) of degree \( O(n) \) such that \( r(x) = p(x)/q(x) \) is a polynomial of degree \( n \). How fast can one compute \( r(x) \)?

11. Given \( n \) numbers \( a_1, \ldots, a_n \), how fast can one decide if there are at least \( n/2 \) distinct values among them in the bounded degree decision tree model? For example, the numbers might be \( 3, 4, 9, 3, 5, 3, 4 \). In this case, the distinct values are \( 3, 4, 5, 9 \) and there at least \( 4 \geq n/2 = 3.5 \) distinct values in this case.

12. Given \( n \) numbers \( b_1, \ldots, b_n \), each one of them either 1 or 2, how many numbers do one have to read, in the worst case, to decide if \( \prod_{i=1}^{n} b_i \geq \lfloor \log n \rfloor \)?

13. How fast is \texttt{RandomSelect} in the worst case?

14. What is the running time of KMP on a string of length \( n \) and a pattern of length \( m \)?

15. Given a graph \( G \) represented using adjacency lists, how fast can one check whether \( G \) is a tree?
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Θ(1)</td>
<td>G.</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>B.</td>
<td>Θ(log n)</td>
<td>H.</td>
<td>Θ(n + m)</td>
</tr>
<tr>
<td>C.</td>
<td>Θ((log n) log log n)</td>
<td>I.</td>
<td>Θ(n log* n)</td>
</tr>
<tr>
<td>D.</td>
<td>Θ(log^2 n)</td>
<td>J.</td>
<td>Θ(n log log n)</td>
</tr>
<tr>
<td>E.</td>
<td>Θ(√n)</td>
<td>K.</td>
<td>Θ(n log n)</td>
</tr>
<tr>
<td>F.</td>
<td>Θ((√n + m) log* n)</td>
<td>L.</td>
<td>Θ(n log n + m)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M.</td>
<td>Θ((n + m) log n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Q.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R.</td>
</tr>
</tbody>
</table>

16. For two strings of length \( n \) and \( m \), how fast can one compute their EDIT DISTANCE?

17. Given a set \( P \) of \( n \) points in the plane, so that the coordinates of each point are positive integers in the range \([1, \ldots, n]\). How fast can one compute the convex-hull of \( P \)?

18. You are given \( m \) numbers, in \( n \) sorted lists. How fast can one sort the \( m \) numbers?

19. Given \( n \) sorted numbers, how fast can one decide if there are two numbers that their sum is zero?

20. Given \( n \) strings of total length \( m \), how fast can one decide if there are two equal strings?
2. [10 Points] A graph $G$ is given to you by an adjacency matrix of $A$ of size $n \times n$. The graph $G$ has $n$ vertices and $m$ edges. Assume that you are also given a function $\text{IsZero}([i..j] \times [k..l])$, which tells you in constant time whether the submatrix $A[i..j][k..l]$ is all zeros (i.e., $a_{xy} = 0$ for $x = i, \ldots, j; y = k, \ldots, l$).

(a) [2 Points] Prove that, in the worst case, just determining the number of neighbors of a vertex $v$ in $G$ would require $\Omega(n)$ calls to $\text{IsZero}(...)$.

(b) [4 Points] Present an algorithm such that given a vertex $v$, it computes all its neighbors in $G$, as fast as possible (as a function of $k$ and $n$), where $k$ is the the number of neighbors of $v$. In particular, how fast is your algorithm, when $v$ has $k = \sqrt{n}$ neighbors? How fast is your algorithm, when $v$ has $k = n/2$ neighbors?

(c) [4 Points] Present an algorithm, as fast as possible, such that given a graph $G$ as above, it generates a linked list representation of $G$. How fast is your algorithm?
3. [10 Points] Prove that the following problem is NP-Complete or provide a polynomial time algorithm for it.

**Problem:** MIN GUARDS

**Instance:** A bipartite graph $G = (A \cup B, E)$, and a positive integer $K$.

**Question:** Is there a set $X \subseteq B$ and $|X| \leq K$, such that for any $a \in A$ there is a vertex $x \in X$ such that $ax \in E$. Namely, can we find a set of $K$ “guards” in $B$ that “sees” all the vertices of $A$. 
4. [10 Points]

The following problem was shown to be NP-Complete in class.

**Problem:** \( K \)-COLORABLE

**Instance:** A graph \( G = (V, E) \), and a positive integer \( K \).

**Question:** Is there a coloring of \( G \) by \( K \) colors. Namely, is there a function \( f : V \rightarrow \{1, \ldots, K\} \) such that \( f(u) \neq f(v) \) if \( uv \in E \)?

(a) [5 Points] You are given an oracle \( \text{ORAC} \) that answers \( K \)-COLORABLE instances (i.e., given a graph \( G \) and \( K \) it outputs whether the graph \( G \) is colorable by \( K \) colors or not) in polynomial time, present a polynomial time algorithm that computes the \( K \) coloring \( f : V \rightarrow \{1, \ldots, K\} \) of \( G \) if such a coloring exists.

(b) [5 Points] You are given an oracle \( \text{ORAC} \) that answers (in polynomial time) \( 25 \)-COLORABLE instances (i.e., given a graph \( G \) it outputs whether the graph \( G \) is colorable by 25 colors or not), present a polynomial time algorithm that computes a 3 coloring \( f : V \rightarrow \{1, 2, 3\} \) of \( G \) if such a coloring exists.
5. [10 Points]

Given two convex polygons $P_1, P_2$ with $n$ vertices (each polygon is defined by a counterclockwise list of its vertices), describe an algorithm, as fast as possible, that computes the convex polygon $P_1 \cap P_2$ which is the intersection of the two polygons. This is the region of all points that belong to both polygons. How fast is your algorithm?
6. [10 Points] 3SUM

Describe an algorithm that solves the following problem as quickly as possible: Given a set of \( n \) numbers, does it contain three elements whose sum is zero? For example, your algorithm should answer True for the set \(-5, -17, 7, -4, 3, -2, 4\), since \(-5 + 7 + (-2) = 0\), and False for the set \(-6, 7, -4, -13, -2, 5, 13\). In case it answer True, it should also output the three numbers that their sum is zero. How fast is your algorithm?