Anna Karlin, 426C Sieg

# Final Exam

Due Monday, December 11, 10am

## Directions

- You may NOT discuss or otherwise communicate in any way about any aspect of this exam with anyone. You are on a strict honor code.
- You may use any non-human materials you want (including books, notes, videotaped lectures, etc.)
- You can turn the exam in by either emailing it to **both** me (`karlin@cs.washington.edu`) and Ashish (`ashish@cs.washington.edu`), bringing it to my office (Sieg 426C), or faxing it to me at (206) 543-2969. In the latter case, please allow for a possible queue at the fax machine. I will not take any exams that arrive after 10am on Monday.
- I will be available from 4pm to 5pm each day to answer questions. On Friday during this period, you can reach me in my office at (206) 543-9344, and on Saturday and Sunday, you can reach me during this period either at (206) 543-9344 or at (206) 223-9344. You can email me questions at any time, but I don't promise to answer them until this time period, so please plan appropriately. In addition, I prefer to answer questions by phone, so if at all possible, if you email me a question, please include a phone number where I can reach you.
- Please check your mail at least once a day over the weekend. It is entirely possible that I will send clarifications out by email to the entire class.

1. (4 points) Consider the stable marriage problem. Suppose that the preference list for boy $i$ and girl $i$ is the same and is $(i, (i+1) \bmod n, \ldots, (i+n-1) \bmod n)$.

   - How many days will it take until the stable marriage algorithm finds a stable pairing?
   - Is the resulting pairing female pessimal?

2. (2 points each) **True or False**:

   - $100n - 5\log n = O(n)$.
   - $100n - 5\log n = O(n \log n)$.
   - $100n - 5\log n = \Omega(n)$.
   - $100n - 5\log n = \Omega(n \log n)$.
   - $2^n = o(2.1^n)$
   - Let $G$ be an undirected graph with $n$ nodes and $m$ edges. There is an algorithm to determine whether $G$ contains a cycle that runs in $O(n)$ time.
   - Let $G$ be a directed graph with $n$ nodes and $m$ edges. Let $v$ and $w$ be nodes in $G$. Depth first search can be used to determine if there is a path in $G$ from $v$ to $w$ in $O(n)$ time.
   - The maximum weight edge in an undirected weighted graph is never in a minimum spanning tree of the graph.
   - In a depth-first search of an undirected graph, there can be edges connecting two unrelated nodes in the depth first search tree. (Two nodes are unrelated if neither is an ancestor of the other.)
   - The path between a pair of vertices in a minimum spanning tree of an undirected graph must be a shortest path between the two vertices in the full graph.
   - The value of any flow in a flow network is upper bounded by the value of any cut in that network.
   - For variables $x_i$ and constants $a_i$,

     $$minimize \sum_{1 \le i \le n} |x_i - a_i|$$

     is a valid objective function for a linear program trying to minimize the distance between the $n$-dimensional point $(x_1, x_2, \ldots, x_n)$ and the $n$-dimensional point $(a_1, a_2, \ldots, a_n)$.
   - The Simplex algorithm may sometimes take superpolynomial time to terminate.
   - If problem $A$ polynomially reduces to problem $B$ and $B$ is NP-hard, then $A$ is also NP-hard.
   - Clique$(n, k)$ is NP-complete because the only way we know how to solve it is to look at all possible subsets of vertices of size $k$ and see if they form a clique, which takes exponential time.
   - The existence of an $n^{\log n}$ time algorithm for the 3-SAT problem implies the existence of an $O(n^{\log n})$ time algorithm for all NP-complete problems.
   - Let $S$ be a set of $n$ points in the plane. Then the weight of a minimum spanning tree on these points is a lower bound on the length of the travelling salesman tour through these points.

- Let $S$ be a set of $2n$ points in the plane. Then the weight of the minimum weight matching on these 2n points is a lower bound on half the length of the optimal travelling salesman tour through these points.
- If you use a completely random hash function to map $N$ elements to a table of size $N$, then with high probability every location will have $O(1)$ elements that hash to it.
- In double hashing all elements are stored in the hash table itself.
- Consider a malicious adversary that chooses keys to be hashed in order to maximize the average search cost. If the adversary knows the hash function to be used, he can choose $n$ keys to obtain an average search cost that is $\Omega(n)$.
- P does not equal NP.

3. (2 points each part, unless otherwise specified) **No explanations needed.**

- Draw a directed graph with a unique topological order.
- (2 points per property) An undirected graph $T = (V, E)$ is a tree if given any two vertices $u$ and $v$ of $T$, there is a unique simple path from $u$ to $v$. If $T$ is a tree, then $T$ has the following properties:

  (a) $T$ is connected.
  (b) $T$ is acyclic.
  (c) deleting any edge of $T$ yields a disconnected graph.
  (d) if $u$, $v$ in $V$ and $e = \{u, v\}$ is not an edge of $T$, then adding $e$ to $T$ yields a graph with exactly one simple cycle, and that cycle contains $e$.
  (e) $T$ has exactly $n - 1$ edges, where $n$ is the number of vertices of $T$.

  For each of these 5 properties, draw an undirected graph that has that property but is not a tree.
- *Fill in the blanks.* (2 points per blank): Consider a linear program with 32 variables. Then ............. equations are needed to define a corner (vertex) of the polytope. Two adjacent corners of the polytope differ in ............. equation(s). A solution to a linear program, say a minimization problem, is found when the simplex algorithm reaches a vertex of the polytope such that the value of the objective function at that vertex is ........................ the value of the objective function at all adjacent vertices.
- What is the worst case running time of `Lookup(K)` in a hash table of size $R$ storing a set of size $n$?
- What is the worst case running time of `Lookup(K)` in a splay tree storing a set of size $n$?
- What is the worst case running time of `Minimum` in a hash table of size $R$ storing a set of size $n$? (`Minimum` returns the value of the key of minimum value in the set being stored.)
- What is the amortized running time of `Minimum` in a splay tree storing a set of size $n$? (`Minimum` returns the value of the key of minimum value in the set being stored.)

- What is the rank of the element 7 in the following splay tree $T$?

```
          20
         /
        10
       /
      7
     /
    4
   /
  3
 /
1
 \
  2
```

- *Fill in the blank:* If the Money Invariant holds for $T$, the minimum number of dollars you would need to put into the tree above in order to (a) satisfy the Money Invariant after a Splay on key 7 and (b) pay for the Splay operation (assuming it costs \$1 to do this Case II, or zig-zig, rotation) is ............................
- Show the result of performing a splay around element 2 in tree $T$.

4. (5 points) Find a maximum flow in the following flow network using the Edmonds-Karp augmenting path algorithm. Show the flow after each augmentation. The label on each edge is its capacity, and all edges are unidirectional (running from the left side of the page to the right side).

```
                 e
              >    \
             /      \
           3/        \4
           /          \
          /            >
s ----> a ----> b ----> c ----> t
\   4      2   >   5       8
  \           /
   \         /
   3\       /2
     \     /
      >   /
        f
```

5. Consider the standard Knapsack problem: Given an integer $K$ and a set of $n$ different items, where the $i$th item has size $s[i]$ (an integer), the problem is to determine if there is a subset of the items whose sizes sum to exactly $K$. We will consider two different variants on this problem here:

   - Our goal in this part of the problem is to give a dynamic programming algorithm for the variant of the Knapsack problem where each item occurs in unlimited supply. This time the problem is to pack items of the given sizes $s[i]$ in the knapsack of size $K$, but each item may appear many times in the knapsack.

     We formulate this as follows. We have $n$ *types* of items, with an infinite supply of items of each type. Each item of the $i$th type has size $s[i]$. We wish to determine if there a subset of the items whose sizes sum to exactly $K$. (There can be an arbitrary number of items of any type in this subset.)

     – (7 points) For any nonnegative integer $J$, define $B(J)$ to be 1 if there is a subset of the items whose sizes sum to exactly $J$, and 0 otherwise (so $B(K)$ is the solution we are seeking). Give a recurrence for $B(J)$. (Don't forget the base case.)

     – (7 points)Give pseudocode for computing $B(K)$.

   - (7 points) We next consider the variant of the original Knapsack problem where each item has a value $v[i]$, and our goal is to find a way to pack the knapsack fully, such that the items in it have the maximal total value among all possible ways to pack the knapsack. Give an integer linear programming formulation of this problem.