University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2013

Name:	
Email:	

- This exam lasts 2 hours. It is closed book and closed notes wing no electronic devices. However, you are allowed a 1 page cheat sheet.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer.
- Write your solution in the space provided for the corresponding problem.
- If any question is unclear, ask for clarification.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Short Answer**

   Answer the following questions using *simplest possible* θ notation. Draw a box around your final answer. No need to justify answers for problems on this page.

   (a) $\binom{n}{3}\frac{1}{n^2}$ *Solution:* $\theta(n)$

   (b) *Worst case* runtime of randomized quicksort on a list of n elements? *Solution:* $\theta(n^2)$

   (c) Expected number of items at the $\log n$ level of a skip list? *Solution:* $\theta(1)$

   (d) Solution to the following recurrence $T(n) = 2T(n/4) + \sqrt{n}$ *Solution:* $\theta(\sqrt{n}\log n)$ *by Master Method.*

   (e) Solution to the following recurrence relation: $f(n) = 3f(n-1) - 2f(n-2)$. *Solution:* $\theta(2^n)$ *or* $\theta(c_1 2^n + c_3)$. *Annihilator is* $(L-2)(L-1)$.

(f) The time to determine if a weighted graph with n nodes and m edges has a negative cycle that is reachable from a given node. *Solution:* $\theta(mn)$. *Use Bellman-Ford.*

(g) Recall that in class we showed how to create a Dynamic table where the amortized costs for Insert and Delete were $\theta(1)$. If an algorithm makes $\theta(n)$ calls to Insert or Delete in a table, what is the worst case cost of all of these calls? *Solution:* $\theta(n)$

(h) What is the worst case cost of a single one of the n calls in the problem above? *Solution:* $\theta(n)$

(i) Recall that Kruskal's algorithm uses the Union-Find data structure as follows: there are n calls to Make-Set, at most $2m$ calls to Find-Set and at most n calls to Union. In class, we showed that the amortized cost of each of these three operations is $O(log^*n)$ when there are n elements in the sets. Based on these facts, what is the amount of time Kruskals spends on Union-Find operations in the worst case? *Solution:* $\theta((m+n)\log^* n)$

(j) You have computed a max flow f in a network G with n nodes and m edges, and now an edge of G has its capacity increase by exactly 1. What is the cost of the most efficient algorithm to find a new max flow for G? *Solution: Compute the residual graph of* G *and then determine whether or not there is an augmenting path in this residual graph. The augmenting path can be found in* $\Theta(m+n)$ *time by using BFS from* s. *At most* 1 *augmenting path is necessary to find the new max flow since the increased capacity of the new edge can increase the value of the flow by at most* 1

2. **Short Answer**

(a) (10 points) Before a party, n people check their hats. The hats are mixed up during the party so that at the end of the party, each person gets a random hat. In particular, each person gets their own hat with probability $1/n$. What is the expected number of people who receive their own hat?

Solution: Let X_i be 1 if person i gets their own hat and 0 otherwise. Let $X = \sum_{i=1}^{n} X_i$. Then $E(X) = \sum_{i=1}^{n} E(X_i) = 1$. So we expect 1 person to get their own hat back.

(b) (10 points) In 4-SAT problem, you are given a boolean formula, f, in conjunctive normal form where each clause has exactly 4 variables, and you are asked if this formula can be satisfied. For example, given $f = (a \vee b \vee c \vee d) \wedge (\neg a \vee \neg b \vee \neg c \vee \neg d) \wedge (a \vee \neg b \vee c \vee \neg d)$, you should return YES since f can be satisfied (for example when a and b are TRUE and c and d are FALSE). Show that 4-SAT is NP-HARD by a reduction from one of the following problems: SAT, 3-SAT, CLIQUE or INDEPENDENT-SET.

Solution: We show this by a reduction from 3-SAT. Let f be an arbitrary 3-CNF formula. Create a new 4-CNF formula f' from f by creating a new variable x and adding it to every clause of f. Then add one additional clause to f' that is $(\neg x \vee \neg x \vee \neg x \vee \neg x)$. Note that any assignment that satisfies f can be used to satisfy f': set all variables for f' the same as for f and set x to FALSE. Further, any assignment that satisfies f' will satisfy f: set all variables for f the same as for f' and ignore the value of x. Thus, f' is satisfiable iff f is satisfiable and so a polytime algorithm for 4-SAT gives a polytime algorithm for 3-SAT.

3. **Dynamic Programming**

   You are given an input string and a dictionary of words, and need to determine if the input string can be segmented into a space-separated sequence of dictionary words. For example, given the dictionary $\{algorithms, data, structure, i, love, snow\}$ and the input string "$iloveal gorithms''$", you should output TRUE since the input can be segmented as "i love algorithms".

   Assume you have a function "InDictionary(x)" that returns TRUE iff a string x is in the dictionary, and this function runs in $O(1)$ time. As input, you are given a string s, which is represented as an array of length n, i.e. $s = s[1, \ldots, n]$. Define a function f such that $f(i)$ is TRUE iff the substring $s[1..i]$ can be segmented for $0 \leq i \leq n$. Define $s[0]$ to be the empty string.

   (a) (15 points) Write a recurrence relation for f. *Solution: Base case:* $f[0] = TRUE$. *For* $1 \leq i \leq n$, $f(i) = TRUE$ *iff there exists some* j, $0 \leq j < i$, *such that* $f(i-j)$ *is TRUE and InDictionary(s[j+1, ..., i]])*

   (b) (5 points) Describe in 1-3 sentences (no need for pseudo-code) how you would create a dynamic program based on your recurrence to find the value of $f(n)$. What are the time and space costs of your algorithm?

   Solution: Create an array of size n to store the values of f. *Compute values of* f *from left to right in this array. Return* $f(n)$. *Time is* $\theta(n^2)$. *Space is* $\theta(n)$.

4. **Max Flow**

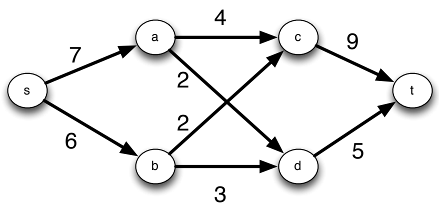

Figure 1

(a) (3 points) Consider the above network (the numbers are edge capacities). Find the max flow, f, and a min cut in this network. *Solution: The flow The cut is* $S = \{s, a, b\}$ *and* $T = \{c, d, t\}$

(b) (3 points) Draw the residual graph G_f (along with its edge capacities). In this residual network, mark the vertices reachable from s and the vertices from which t is reachable. *Solution:*

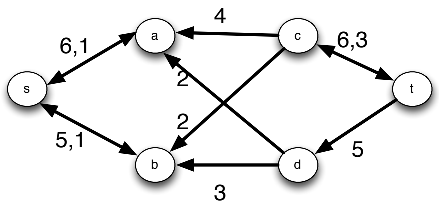

Figure 2

(c) (3 points) An edge of a network is called a *bottleneck* edge if increasing its capacity results in an increase in the maximum flow. List all bottleneck edges in the above network. *Solution: (a,c) and (b,c)*

(d) (3 points) Give a very simple example (containing at most four nodes) of a network which has no bottleneck edges. All capacities on your network should be finite. *Solution: Nodes are s,a,b,t. There are* 4 *edges, each with capacity* 1*: (s,a), (a,t), (s,b), (b,t)*

(e) (8 points) Give an efficient algorithm to identify all bottleneck edges in a network. (Hint: Start by running the usual network flow algorithm, and then examine the residual graph.)
Solution: Run the network flow algorithm and find the final residue network. For each edge in the original graph, check in the residual network whether the source node of the edge is reachable from s and the target node of the edge can reach t. You can use a depth first search algorithm to check the reachability. If the paths from s to u and v to t exist, then edge (u, v) is a bottleneck edge. In fact, you do not have to run the depth first search for each edge. Instead you can run one depth first search for s on the residue network and another for t on the reverse residue network (all edges reversed) and keep the reachable nodes.

5. **Square in Matrices**

   You are given a m by n, matrix, M, where each cell is either a "1" or "0". Your goal is to find a maximum size square sub-matrix with all 1's.

```
0 1 1 0 1
1 1 0 1 0
0 1 1 1 0
1 1 1 1 0
1 1 1 1 1
0 0 0 0 0
```

   For example, the above matrix has a maximum size square matrix that is 3 by 3, with bottom right corner at $M(5,4)$. Give an efficient algorithm to solve this problem. Compute the time and space costs of your algorithm.

   Solution: Let $S(i,j)$ be the size of the largest square matrix of all 1's whose bottom right corner is $M(i,j)$. We can write a recurrence relation for $S(i,j)$ as follows. If $i < 1$ or $j < 1$, then $S(i,j) = 0$. Else if $M(i,j) = 0$ then $S(i,j) = 0$. Otherwise if $M(i,j) = 1$, then $S(i,j) = 1 + \min(S(i,j-1), S(i-1,j-1), S(i-1,j))$. We create this S array moving from left to right and top to bottom. We also keep track of the maximum value, M, in the S array and the location L_M where this maximum value occurs. After filling in all entries of S, we return that the largest all 1 submatrix is of size M by M and has bottom right corner at location L_M. This algorithm take $O(mn)$ time.

5. **Square in Matrices, continued.**