

# CSE548/AMS542 Fall 2007 Analysis of Algorithms

## Final

- This is an open book exam.
- There are 6 problems and 30 points total.
- You can refer to the algorithms we have covered in class without referring to the details. If you give a greedy algorithm but do not give the proof of its correctness, you may lose all (if your algorithm is wrong) or partial points (if your algorithm is correct).
- 4pm-6pm on December 14th.

By signing below I declare that I follow the rule of academic integrity and finish the exam on my own, without the help of others.

Name \_\_\_\_\_

ID \_\_\_\_\_

Signature \_\_\_\_\_

Score	Grade
1	
2	
3	
4	
5	
6	
Total	

1. **Short questions** (2pts) Answer the following questions. You **do not** need to give proofs or counter-examples.

(a) Given a directed graph with a source  $s$  and destination  $t$ , each edge has an interger capacity. Take the max flow  $f$ , we call an edge to be saturated if the flow on that edge is the same as the capacity. Whether the following 2 claims are correct or not? (1) All edges on the min cut are saturated; (2) All saturated edges are on the min cut. Answer True or False.

(b) Given an undirected graph, the Hamiltonian cycle is a cycle that visits each *vertex* once and exactly once. The Eulerian cycle is a cycle that visits each *edge* once and exactly once. We know that checking whether a graph has a Hamiltonian cycle is NP-hard. Now, is it NP-hard to check whether a graph has an Eulerian cycle? Answer Yes or No.

2. **Selfish load balancing** (6pts). There are  $n$  selfish users and  $m$  machines. Each user  $i$  has a task to finish and it will take time  $t_i$  for any machine to finish the job. Each user will submit its task to one machine and would like to use the machine with the smallest total load (the total time to finish all the jobs scheduled on this machine).

Thus we define a game with the selfish users. The *cost* of user  $i$  is the total load of the machine that it selects. Each user would like to minimize its cost. A *Nash equilibrium* of a game is a collection of strategies for the users such that no user would like to switch to another machine, to have a smaller cost, given that the other users do not change their strategies. Thus a Nash equilibrium represents a stable state.

(a) Show that there exists at least one Nash equilibrium. (3pts)

- (b) Show that at *any* Nash equilibrium, the load of any machine is at most twice the highest load of the optimal load balanced assignment (the assignment that minimizes the maximum load of any machine). (3pts)

3. **Red-blue separation.** (4pts). Given a set of  $m$  red points and  $n$  blue points in the plane, find a minimum length cycle that separates the red points from the blue points. That is, the red points are inside the cycle and the blue points are outside the cycle, or vice versa. This problem is called the red blue separation problem.

Show that the red blue separation problem is NP-complete.

4. **Maximum matching and maximal matching.** (4pts) Given a bipartite graph with  $n$  left nodes and  $n$  right nodes, a maximal matching is a matching such that no edges can be included. A maximum matching is a matching with the maximum size.

(a) Give an example of a bipartite graph with  $2k$  left nodes and  $2k$  right nodes with a maximal matching of size  $k$  and a maximum matching of size  $2k$ . (2pts)

(b) Prove that the size of a maximal matching is at least  $1/2$  of the size of the maximum matching. That is, the example in (a) is the worst case scenario. (2pts)

5. **Flow algorithms.** (6pts) We will look at the max flow algorithm in a dynamic setting, i.e., with edge insertion, deletion or change of edge capacities. Given a directed graph  $G(V, E)$ . Each edge  $e = (u, v)$  has a capacity  $c(u, v)$ . We have already computed the maximum flow  $f$  from a source  $s$  to a sink  $t$ . Now there is some change to the network and we examine how to update the max flow accordingly. We do not want to run the flow algorithm from scratch.

- (a) Suppose that one edge in  $G$  has its capacity increased by 1. Denote the new graph as  $G'$ . Show how to find the max flow in  $G'$  in time  $O(n + m)$ . (3pts)

- (b) Suppose that one edge in  $G$  has its capacity *decreased* by 1. Denote the new graph as  $G'$ . Show how to find the max flow in  $G'$  in time  $O(n + m)$ . (3pts)

6. **Skip lists.** (8pts) Skip list is a data structure for fast search in a linked list data structure. Given a list of numbers organized in a linked list in an increasing order, we ask the following queries: is element  $x$  in the list or not? If we store the set of numbers in an array, then we can use binary search to answer the query in  $O(\log n)$  time. Now we propose a randomized structure for binary search in a linked list.

The idea is the following. Take the first list  $S_0$ . Now each element has a probability  $1/2$  to be promoted to the list  $S_1$ . The elements promoted to  $S_1$  will be organized in a linked list, also in an increasing order. Similarly, we take the list  $S_1$  and promote with probability  $1/2$  each element to the list  $S_2$ , and so on, until we have a list  $S_k$  with only 1 element.  $S_i$  is called the list at level  $i$ .

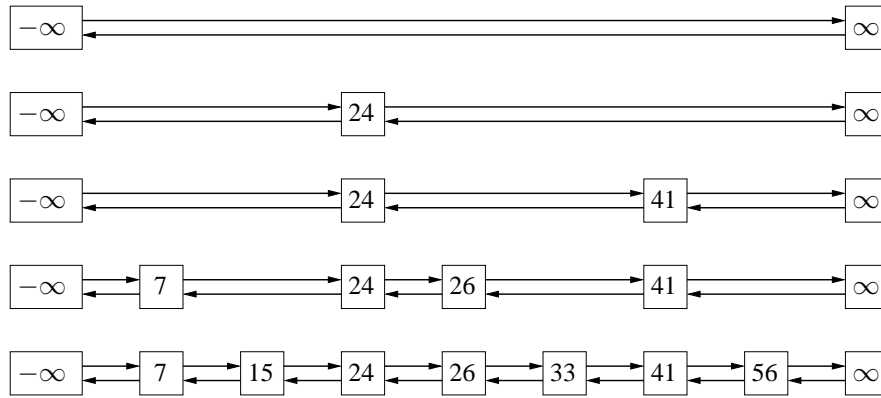


Figure 1: Skip list.

This problem is to analyze the performance of this data structure.

- (a) What is the probability that an element appears in  $S_i$ ? (1pt)

- (b) What is the expected size of the skip list? (1pt)

(c) Show that  $k$ , the number of levels, is  $O(\log n)$  with high probability, i.e., probability  $1 - 1/n$ ? (2pts)

(d) Consider an element  $x$  on  $S_i$  and its right neighbor  $y$  on  $S_i$ . Look at the list in  $S_{i-1}$  at one level lower. Now between  $x$  and  $y$  there are  $m$  elements in  $S_{i-1}$ . What is the expected value of  $m$ ? (2pts)

(e) Now we will use the skip list to answer a query for an element  $w$ . We take every adjacent elements  $x, y$  in a skip list  $S_i$  and call it an interval. We start from the highest level, with an interval  $[-\infty, \infty]$  containing the element  $x$ . And move down the lists to search for the interval containing  $x$  at each  $S_i$ , until we reach the interval at  $S_0$  containing  $x$ . In that case, we can answer whether  $w$  is inside  $S_0$  or not.

What is the expected query cost? (2pts)