Homework #3

(Due: Nov 28)

Task 1. [80 Points] Randomization in Binomial Heaps

Consider the following way of modifying a binomial heap with lazy union to support *Decrease-Key* operations. Whenever the key of a node is decreased it is cut from its parent (along with the subtree rooted at that node) and inserted as a separate tree in the root list. Then starting from the parent of that node we keep moving toward the root and flip a coin at each node we encounter until we either reach the root or obtain "tails", whichever comes first. We do not flip a coin at the root. If the coin flip at a node comes up "heads" we cut the node along with the subtree rooted at it from its parent and include it as a separate tree in the root list. If we obtain "tails", we stop. The implementations of all other operations remain unchanged. The *rank* of a node is still equal to the number of its children, and the rank of a tree is equal to the rank of its root. As in the original binomial heap we do not link two trees unless they have the same rank.

(*a*) [**20 Points**] Suppose the coin is fair, i.e., the probability of obtaining a head is $\frac{1}{2}$. Find the expected amortized costs of all operations.

(*b*) [**25 Points**] Suppose the coin is biased in the sense that a tail is three times more likely to occur than a head. Then show that the expected maximum degree of a node is $\log_\phi n$, where $\phi \approx 1.380277569$ and n is the number of nodes in the heap.

(*c*) [**35 Points**] Find the expected amortized costs of all operations assuming the coin is biased as in part (*b*).

Task 2. [100 Points] Overcrowded Office Hours

I am teaching a huge class this semester, and so a lot students show up during my office hours, particularly in weeks when homeworks are due. When I answer a question from one student I often want others to listen as well, but my office is too small to accommodate all students that show up on any given day. So students enter my office in small groups, and I end up answering the same question to each group. That seems like a waste of time. Also often I am unable to talk to all students during the limited 1.5 hours of time (MW) I have reserved as my office hours. I have decided to use a simple randomized algorithm based on a simple observation to improve the situation.

I have made the following observation. Most students have one or more friends. So a student does not need to listen to me directly provided at least one of his friends has already done so. He can simply consult that friend of his. So I want each student x who enters my office and listens to my answers to tell each of his/her friends about everything I said. But I do not want x's friends to update their other friends on whatever they hear from x beacuse the more times a message is

```
1.  let S_0 be the initial set of students waiting outside my office
2.  let S ⊆ S_0 be the set of students selected to enter my office
3.  S ← ∅
4.  i ← 0
5.  while |S_i| > 0 do
6.      for each student x ∈ S_i do
7.          mark[x] ← False
8.          if x has no friend in S_i then mark[x] ← True          {mark x for inclusion into S}
9.          else
10.             let f_{x,i} be the number of friends of x in S_i
11.             mark[x] ← True with probability 1/(2f_{x,i})       {mark x for inclusion into S}
12.         mark'[x] ← mark[x]
13.     for every pair of friends x, y ∈ S_i with x ≠ y and mark'[x] = mark'[y] = True do
14.         if x has fewer friends than y in S_i then mark[x] ← False
15.         else mark[y] ← False
16.     for each student x ∈ S_i with mark[x] = True do
17.         include x into S
18.         remove x along with all its friends from S_i
19.     S_{i+1} ← S_i
20.     i ← i + 1
21. return S
```

Figure 1: Given an initial set S_0 of students waiting outside my office this algorithm chooses a subset $S \subseteq S_0$ that will get to enter my office during my office hours.

relayed the more likely that it will get corrupted. So only the students who hear directly from me are allowed to relay my answers to their direct friends.

The algorithm in Figure 1 takes the initial set S_0 of students waiting outside my office as input, and outputs a set $S \subseteq S_0$ of students that will get to enter my office to talk to me directly. This task asks you to prove properties of that algorithm.

For any $i \geq 0$ and $x \in S_i$ let $F_{x,i}$ be the set of friends of x in S_i, and let $f_{x,i} = |F_{x,i}|$. Then we say that x is *important* in the i-th iteration of the ***while*** loop in lines 5–20 of Figure 1 provided at least $\frac{f_{x,i}}{3}$ of those friends has no more than $f_{x,i}$ friends each in S_i. Otherwise x is *unimportant*. Let $\bar{S}_i \subseteq S_i$ be the set of *important* students in S_i.

If x and y are friends then the the pair $\{x, y\}$ denotes a *friendship*. Friendship is commutative meaning that if x is y's friend then y is x's friend. However, the relationship is not transitive, that is, if x is y's friend and y is z's friend then x is not necessarily z's friend. We say that a *friendship* $\{x, y\}$ is *important* in a given iteration of the ***while*** loop provided either x or y or both are *important* in that iteration. Otherwise $\{x, y\}$ is *unimportant*. Let F_i be the set of all friendships in S_i, and let $\bar{F}_i \subseteq F_i$ be the set of *important* friendships.

In iteration i (≥ 0) of the ***while*** loop (in lines 5–20), let $S'_i \subseteq S_i$ be the set of students marked for inclusion in S in lines 6–12. Also let $S''_i \subseteq S_i$ be the set of students included in S in the ***for*** loop of lines 16–18, and let $F''_i = \bigcup_{x \in S''_i} F_{x,i}$.

Let L be the number of times the ***while*** loop iterates, i.e., L is the smallest value of i (≥ 0) such that $|S_i| = 0$.

(a) [**10 Points**] Consider the initial set S_0 of students in line 1 of the algorithm in Figure 1 and the selected set S of students returned by the algorithm in line 21. Argue that for each student $x \in S_0$ either x or at least one of his/her friends in S_0 must be in S. Also prove that if $x \in S$ then none of x's friends in S_0 are in S.

(b) [**15 Points**] Let $x \in S_i$ be a important student in the i-th iteration of the ***while*** loop (in lines 5–20) of Figure 1 with at least one friend in S_i. Prove that $\Pr\{\ F_{x,i} \cap S'_i \neq \emptyset\ \} \geq 1 - e^{-\frac{1}{6}}$.

(c) [**15 Points**] Prove that $\Pr\{\ s \in S''_i \mid s \in S'_i\ \} \geq \frac{1}{2}$.

(d) [**10 Points**] Use your solutions from parts 2(b) and 2(c) to show that $\Pr\{\ s \in S''_i \cup F''_i \mid s \in \bar{S}_i\ \} \geq \frac{1}{2}\left(1 - e^{-\frac{1}{6}}\right)$.

(e) [**20 Points**] Prove that $|\bar{F}_i| \geq \frac{|F_i|}{2}$.

(f) [**10 Points**] Use your results from parts 2(d) and 2(e) to prove that the expected number of friendships removed from S_i in the ***for*** loop of lines 16–18 is at least $\frac{1}{4}\left(1 - e^{-\frac{1}{6}}\right)|F_i|$.

(g) [**10 Points**] Use your result from part 2(f) to find the expected value of L.

(h) [**10 Points**] What is best upper bound on L you can obtain that holds w.h.p. in $|S_0|$?