York University
EECS 4101/5101, Winter 2023
Assignment 5

Due Date: April 11th, at 23:59

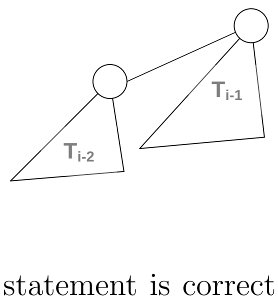
It's your road, and yours alone.
Others may walk it with you, but no one can walk it for you ...
Rumi

All problems are written problems; submit your solutions electronically **only via Crowdmark**. You are welcome to discuss the general idea of the problems with other students. However, you must write your answers individually and mention your peers (with whom you discussed the problems) in your solution. There is also a bonus question, which is harder than other questions. It is recommended to approach this question only if you have completed other questions. Please refer to the course webpage for guidelines on academic integrity.

Problem 1 Binomial Tree Variant [4 + 6 + 6 = 16 marks]

In the class, we learned that a binomial tree of order i can be formed by taking two copies of a binomial tree of order $i-1$ and letting the root of one tree have the other tree as its first child. Consider the following definition of **Fibomial** trees:

- A Fibomial tree of order 0 is a single node and a Fibomial tree of order 1 is also a single node.
- A Fibomial tree of order i is formed by taking a Fibomial tree of order $i-1$ and a Fibomial tree of order $i-2$ letting the root of the first tree (of order $i-1$) have the other tree as its first child (see the figure below).



a) Indicate whether the following statement is correct or not. Provide a justification of your answer.

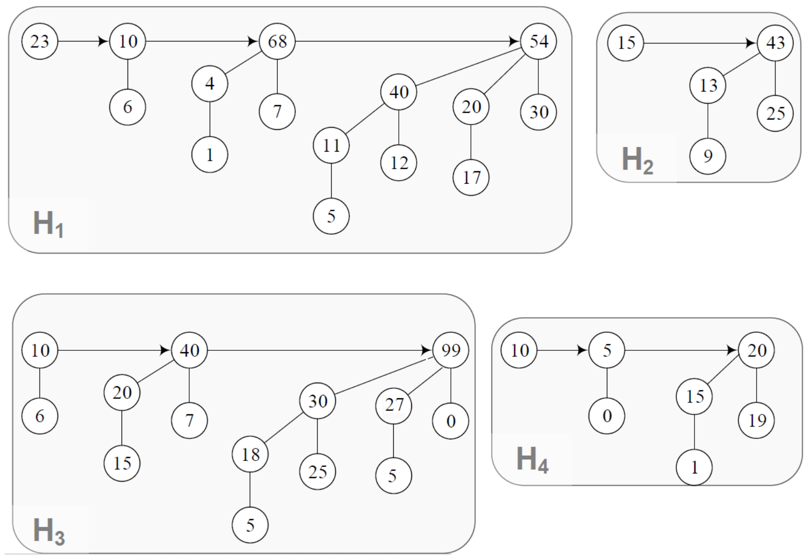
Let T_k denote the Fibomial tree of order k. The children of the root of T_k are the Fibomial trees $T_{k-2}, T_{k-3}, \ldots, T_2, T_1, T_0, T_0$ (more precisely, there is exactly one copy of each T_i for $i \in \{k-2, k-3, \ldots, 1\}$ and two copies of T_0 as a children of T_k).

b) Indicate how many nodes exist in a Fibomial tree of order k. You should provide a recursive formula and relate it to a known series.

c) Indicate what the height of a Fibomial tree of order k is. Again, you should provide a recursive formula and solve it.

Problem 2 Binomial Heap Operations [4 + 4 + 4 = 12 marks]

In this problem, we review binomial heap operations for heaps of the figure below. In case of merging trees in the following operations, in case there were three binomial trees of the same order, merge the two 'older' trees (keep the new tree which is the product of previous merge).



a) Apply the operation merge on heaps H_1 and H_2. Show intermediate steps.

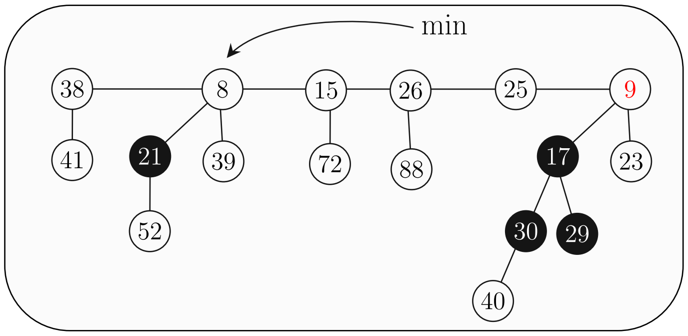
b) Apply operation *extract-max* on heap H_3. Show intermediate steps.

c) Apply operation *insert* on heap H_4. Assume you insert value $x = 30$ to the heap. Show intermediate steps.

Problem 3 Fibonacci Heap Operations [4 + 4 = 8 marks]

a) In the Fibonacci heap below, apply the operation extractMin. Show your work.

b) In the original heap, decrease the key of 40 to 10. Show your work.



Problem 4 Fibonacci Heap Analysis [5 + 5 = 10 marks]

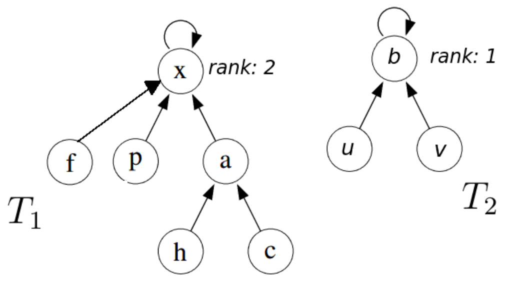
Consider a different potential for Fibonacci heaps, defined as $\Phi'(H) = 2t(H) + 3m(H)$. Indicate the amortized time for (a) extractMin and (b) decreaseKey operations. Show your work.

Problem 5 Union by Weight Analysis [8 marks]

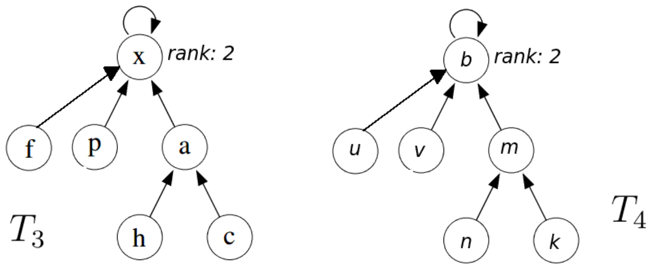
In this problem, we would like to show the amortized time of a union operation when union-by-weight on linked-lists is used is $\Omega(\log n)$. For that, we need to come up with a sequence of $\Theta(n)$ operations for which the amortized cost per operation is $\Omega(\log n)$. We start with $make - set(x_i)$ for $i \in \{1, 2, \dots n\}$ where n is a power of 2. Provide a consequent sequence of $\Theta(n)$ union operations so that the total number of updated pointers for all operations is $\Omega(n \log n)$.

Problem 6 Union-Find Operations [4 + 4 = 8 marks]

a) Consider a union-find structure based on union-by-rank and path-compression which is formed by T_1 and T_2 in the following figure. Draw the result after the following operations: union(p, v), find(h).



b) Consider a similar structure formed by T_3 and T_4 in the following figure. Draw the result after the following operations: union(p, u), find(k). For the union operation, as both trees have the same rank, assume x becomes the parent of the united tree.



Problem 7 Skip Lists [4 + 6 = 10 marks]

a) Starting with an empty skip list, insert the seven keys $30, 10, 25, 20, 15, 70, 12$. Draw your **final answer** as we saw in the slides. Use the following coin tosses to determine the heights of towers (note, not every toss is necessarily used):

$$H, T, T, H, H, H, T, H, H, T, T, H, H, T, H, T, T, H, T, H, H, T, T, H, H, H, T, H, T, \ldots$$

b) Consider a skip list in which we build new towers with probability 3/4.

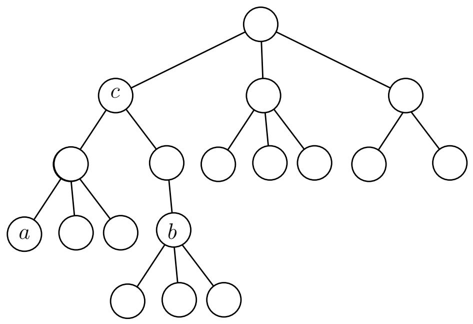
> When adding an element to the skip list, we flip two coins at the same time, and repeat this until both coins come up tails. The number of times we toss both coins defines the height of the tower (i.e., if both coins come up tails in the first flips, there will be one node in the skip list, if both come coin on the second try, the number of nodes will be 2, etc.).

Using the probability for tower heights described in the above quote, derive the expected height of a tower.

Problem 8 Bonus [7 marks]

Given a rooted tree T, the *deepest common forefather (DCF)* of two nodes u and v is the deepest node x in the tree such that u and v belong to the subtree rooted at x. For example, DCF(a, b) in the figure below is node c.

Assume T is a tree of size n (for some large n) and suppose we are given n pairs of nodes from T. Describe an algorithm that reports the DCF of all pairs in time $o(n \log n)$. Briefly justify your answer.


c
a
b