CSCI 570 Homework 1

Due Date: Jan. 26, 2023 at 11:59 P.M.

1. Arrange the following functions in increasing order of growth rate with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$

$$\sqrt{n}^{\sqrt{n}}, n^{\log n}, n\log(\log n), n^{\frac{1}{\log n}}, 2^{\log n}, \log^2 n, (\log n)^{\sqrt{\log n}}.$$

solution:

$$n^{\frac{1}{\log n}}, \log^2 n, (\log n)^{\sqrt{\log n}}, 2^{\log n}, n\log(\log n), n^{\log n}, \sqrt{n}^{\sqrt{n}}$$

Explanation: taking logarithms of all, we have

$$\begin{aligned}
\log \sqrt{n}^{\sqrt{n}} &= \sqrt{n}\log n^{1/2} = \frac{\sqrt{n}}{2}\log n \\
\log n^{\log n} &= \log^2 n \\
\log(n\log(\log n)) &= \log n + \log(\log(\log n)) \\
\log n^{\frac{1}{\log n}} &= \frac{1}{\log n}\cdot \log n = 1 \\
\log(2^{\log n}) &= \log n \\
\log(\log^2 n) &= 2\log(\log n) \\
\log(\log n^{\sqrt{\log n}}) &= \sqrt{\log n}\log(\log n)
\end{aligned}$$

Based on these, we can order all the functions by using

$$1 \le 2\log(\log n) \le \sqrt{\log n}\log(\log n) \le \log n \le \log n + \log(\log(\log n)) \le \log^2 n \le \frac{\sqrt{n}}{2}\log n$$

for large n. For example,

$$\begin{array}{llll}
n^{\frac{1}{\log n}} & = 2^1 & = O(2^{2\log(\log n)}) & = O(\log^2 n) \\
\log^2 n & = 2^{2\log(\log n)} & = O(2^{\sqrt{\log n}\log(\log n)}) & = O(\log n^{\sqrt{\log n}})
\end{array}$$

2. The Fibonacci sequence $F_1, F_2, F_3 \ldots$, is defined as follows. $F_1 = F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n \ge 3$. For example, $F_3 = F_1 + F_2 = 2$ and

$F_4 = F_3 + F_2 = 3$. Prove by induction that $F_n = \frac{a^n - b^n}{\sqrt{5}}$, where $a = \frac{1+\sqrt{5}}{2}$ and $b = \frac{1-\sqrt{5}}{2}$.

solution. BC: We first check the base cases. We have

$$F_1 = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}\right) = 1$$

and

$$F_2 = \frac{1}{\sqrt{5}}\left(\frac{(1+\sqrt{5})^2}{4} - \frac{(1-\sqrt{5})^2}{4}\right) = 1.$$

Therefore, F_1 and F_2 satisfy the formula.

IH: For a fixed $n \geq 3$ suppose the formula $F_k = \frac{a^k - b^k}{\sqrt{5}}$ holds for $k < n$.

IS: Then we have

$$\begin{aligned}
F_n &= F_{n-1} + F_{n-2} && \text{(by the definition of the Fibonacci sequence)} \\
&= \frac{a^{n-1} - b^{n-1}}{\sqrt{5}} + \frac{a^{n-2} - b^{n-2}}{\sqrt{5}} && \text{(by the inductive hypothesis)} \\
&= \frac{-a^n + a^{n-1} + a^{n-2}}{\sqrt{5}} + \frac{b^n - b^{n-1} - b^{n-2}}{\sqrt{5}} + \frac{a^n - b^n}{\sqrt{5}} \\
&= \frac{-a^{n-2}(a^2 - a - 1)}{\sqrt{5}} + \frac{b^{n-2}(b^2 - b - 1)}{\sqrt{5}} + \frac{a^n - b^n}{\sqrt{5}} \\
&= 0 + 0 + \frac{a^n - b^n}{\sqrt{5}} = \frac{a^n - b^n}{\sqrt{5}} && (a^2 - a - 1 = b^2 - b - 1 = 0)
\end{aligned}$$

Therefore, we conclude $F_n = \frac{a^n - b^n}{\sqrt{5}}$ by induction.

3. Let T be a breadth-first search tree of a connected graph G. Let (x, y) be an edge of G, where x and y are nodes in G. Prove by contradiction that in T the level of x and the level of y differ by at most 1.

solution. We prove this by contradiction. Suppose their levels differ by more than 1 and without loss of generality assume x has a lower level, say, i. Then consider the the moment when the edges incident to x are being examined. Because (x, y) is an edge of G, y is one of the nodes discovered from x at this moment and thus belongs to level $i + 1$ or a

smaller level. This contradicts the assumption that their levels differ by more than one and y has a higher level, which completes the proof.

4. We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T. Prove by contradiction that G has the same structure as T, that is, G cannot contain any edges that do not belong to T.

   solution. We prove this by contradiction. Suppose G doesn't have the same structure as T. Then there exists an edge $(x, y) \in E$ but not in T. By the fact that T is a DFS tree, x is ancestor of y or y is ancestor of x. However by the fact that T is a BFS tree and from Problem 3., we know that the levels of x and y differ at most 1. Combining these two shows that (x, y) is an edge in T, which contradicts the assumption.

5. You have been given an unweighted, undirected, and connected graph G = (V, E). Device an algorithm to determine maximum of the shortest paths' lengths between all pairs of nodes in graph G (Also called diameter of the graph). Also, determine the time complexity of your algorithm and justify your answer.

   solution. Run BFS from each node as source node in G and determine the maximum layer reached while running BFS from that node. In every iteration of BFS, keep updating maximum layer reached (keeping the maximum value). Return the highest maximum layer reached (would be the diameter of the graph). Worst Case Time complexity: $O(V*(V+E))$

6. You have been given an unweighted and undirected graph G = (V, E), and an edge $e \in E$.

   a. Your task is to devise an algorithm to determine whether the graph G has a cycle containing that specific given edge e. Also, determine the time complexity of your algorithm and justify your answer. Note: To become eligible for full credits on this problem, the running time of your algorithm should be bounded by $O(V + E)$.

   b. Will your algorithm still work if the graph is unweighted but directed?

If not, how would you modify your algorithm to make it work on an unweighted-directed graph.

solution.

Case-1: Undirected graph

Let the two nodes forming the edge 'e' be s and t. Delete e from graph G and run BFS from s. If t can be reached on running BFS from s => there is a cycle in the graph containing edge e. Time complexity => time complexity of BFS = O(V+E)

Case-2 Directed graph

Let the directed edge 'e' in graph G be from node s− >t. Delete e and run BFS from t. If s can be reached on running BFS from t => there is a cycle in the graph containing edge e.

7. Use prove by induction to show that in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

solution.

Let L(T) be the number of leaf nodes in binary tree T, and N(T) be the number of nodes in the binary tree with two children.

To prove: $N(T) = L(T) - 1$

Base case: T has only one node. Here, $N(T) = 0, L(T) = 1$. The condition checks out.

Induction Hypothesis: let T' be a tree with n vertices. Here, $N(T') = L(T') - 1$

Induction Step: Let T be a tree with n+1 vertices. Let v be a leaf node and u be it's parent node. We obtain T' when we delete v from T.

Case-1: u had only one child v (prior to deletion):

After deletion, u becomes a leaf node, and there would be no change in the number of nodes having 2 children.

$N(T) = N(T')$

$L(T) = L(T')$

Now, we have, $N(T') = L(T') - 1$, which means $N(T) = L(T) - 1$

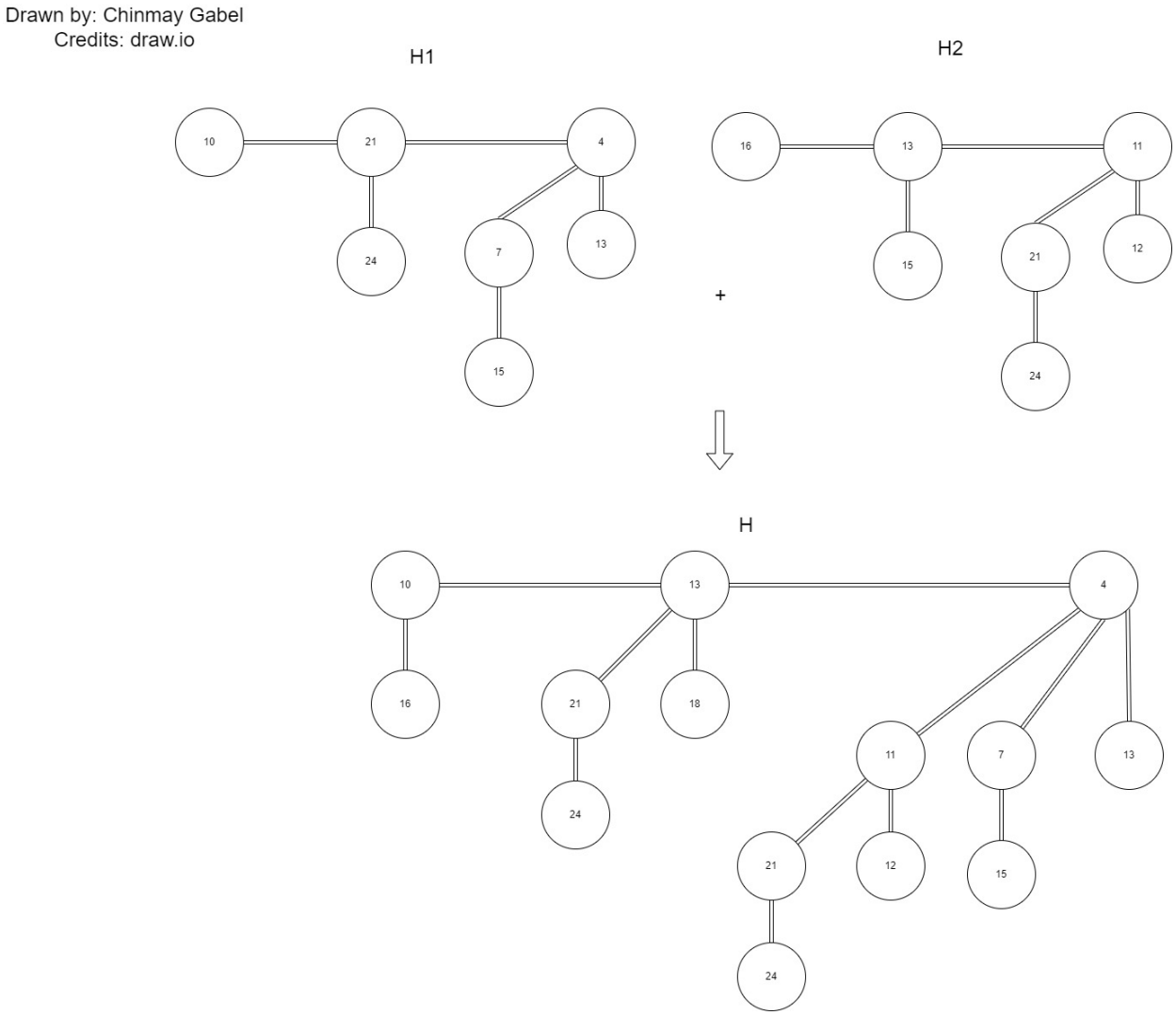
Case-2: u had only two children (prior to deletion):

After deletion, u is no longer a node with two children, and there would

be a decrement in the number of nodes having 2 children. Also, we lost a leaf node.
$N(T') = N(T) - 1$
$L(T') = L(T) - 1$
Now, we have, $N(T') = L(T') - 1$, which means $N(T) - 1 = L(T) - 1 - 1$ or, $N(T) = L(T) - 1$

8. This problem is divided into following subsections:
   a. Consider a sequence of numbers: 4, 13, 7, 15, 21, 24,10 Construct a binomial min-heap H1 by reading the above numbers from left to right. Draw all the intermediate binomial heaps as well as the final binomial heap H1. Illustrate your work clearly and concisely.
   b. Repeat a. for another sequence of numbers: 11,12,21,24,18,13,16 to construct another binomial min-heap H2.
   c. Merge H1 and H2 to construct the final binomial heap H. Draw all the intermediate binomial heaps (while merging H1 and H2) as well as the final binomial heap H.

   solution. Refer 1.*jpg* under images folder (or the image below)



9. Consider an array on which the following operations can be performed:
addLast(e): An element 'e' is added at the end of the array.
deleteEveryThird(): Removes every third element in the list i.e. removes the first, fourth, eighth, etc., elements of the array.

   You may assume that addLast(e) has a cost 1, and deleteEveryThird() has a cost equals to the number of elements in the list. What is the amortized cost of addLast and deleteEveryThird operations? Consider the worst sequence of operations. Justify your answer with proper explanation using accounting method.

   solution. The amortised cost for each operation is constant. Can be

shown using the accounting method by giving a cost 4 to addLast operation that the credit never becomes negative.

Try#1

Charge for addLast = 2

Charge for deleteEveryThird = 0

Operation	Charge	Actual Cost	Total Credit
addLast	2	1	1
addLast	2	1	2
addLast	2	1	3
addLast	2	1	4
deleteEveryThird	0	4	0
deleteEveryThird	0	2	-2

Try#2

Charge for addLast = 3

Charge for deleteEveryThird = 0

Operation	Charge	Actual Cost	Total Credit
addLast	3	1	2
addLast	3	1	4
addLast	3	1	6
addLast	3	1	8
addLast	3	1	10
addLast	3	1	12
deleteEveryThird	0	6	6
deleteEveryThird	0	4	2
deleteEveryThird	0	2	0
deleteEveryThird	0	1	-1

Try#3 (works!)
Charge for addLast = 4
Charge for deleteEveryThird = 0 or 1 (will not matter)
When n is large and we perform a series of n addLast operation, the total credit stored would be 3n. Once we perform a deleteEveryThird operation, roughly n/3 of the list items are removed, and 2n/3. list items are retained.
For removing each element from the list with the help of multiple calls to deleteEveryThird function, we would need $n+2n/3+4n/9+8n/27+\ldots =$

$3n$ charges, which is exactly equal to the total number of credits stored. Therefore, the total credit will never become negative.

10. Given a sequence of n operations, suppose the i-th operation cost $i + j$ if $i = 2^j$ for some integer j; otherwise, the cost is 1. Prove that the amortized cost per operation is $O(1)$.

    solution. By the definition of the amortized cost per operation, we have

$$
\begin{aligned}
0 &\le \frac{1}{n}\left(\sum_{i=1}^{n} \mathbf{1}[\exists j,\ 2^j = i] \cdot (i+j) + \mathbf{1}[\nexists j,\ 2^j = i]\right) \\
&\le 1 + \frac{1}{n}\left(\sum_{i=1}^{n} \mathbf{1}[\exists j,\ 2^j = i] \cdot (2 \cdot 2^j)\right) \qquad (i + j \le 2i = 2 \cdot 2^j) \\
&\le 1 + \frac{1}{n}\left(\sum_{j=1}^{\lfloor \log n \rfloor} (2 \cdot 2^j)\right) \le 5 = O(1)
\end{aligned}
$$

11. **Online Questions.** Please go to DEN (https://courses.uscden.net/) and take the online portion of your assignment.