CECS 528 – Midterm 1 - Fall 2022
75 Minutes

Name: Uday Teja Vunnam
Student ID: 029406505

Rules for completing the problems:
No electronic devices or interpersonal communication allowed when solving these problems.
FAILURE TO ABIDE BY THESE RULES MAY RESULT IN A FINAL COURSE GRADE OF F.

10/10

Directions:
Choose up to 5 problems to solve. Clearly mark each problem you want to be graded by placing a 'Y' to indicate "Yes, grade this problem".
If you don't mark any problems for me to grade or mark 6 or more problems, then I will grade for the 5 fewest points.

Problem	1	2	3	4	5	6
Grade?	Y	Y	Y	Y	Y	
Points Earned	2	2	2	2	2	

1- Does $(\log n)^{\log n}$ have quasi-polynomial growth? If yes, determine its $c - value$ (required in the definition of quasi-polynomial). If not, which grows faster? Show work and defend your answer.

$f(n) = (\log n)^{\log n}$, quasi-polynomial function $g(n) = 2^{\log^c n}$

→ doing the logarithmic test ⇒ $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(\log n)^{\log n}}{2^{\log^c n}}$; $\lim_{n\to\infty} (\log n)^{\log n} - \log 2^{\log^c n}$)

$= \lim_{n\to\infty} \log(\log n)^{\log n} - \log^c n$ ⇒ $\lim_{n\to\infty}$

Since $c > 1$, the above limit is equal to

$\lim_{n\to\infty} (\log(\log n) - \log^{c-1} n) = -\infty$.

Since e.g. letting $y = \log n$, $\log y = O(y^{c-1})$

Therefore $(\log n)^{\log n}$ grows more slowly than any quasi-polynomial function

2- Let $f(n)$ be an asymptotically positive function. Prove or disprove the following conjecture.

$$f(n) + o(f(n)) = \theta(f(n))$$

Ans: we need to prove that $f(n) + o(f(n)) = \Theta(f(n))$. But a function in $o(f(n))$ is definitely smaller than $f(n)$. So for sufficiently large n, we have

$f(n) + o(f(n)) \leq 2f(n) = \Theta f(n)$

$\therefore f(n) + o(f(n)) = \Theta(f(n))$ is positive if assuming functions are positive.

In the same way we can easily prove that this is valid

2/2

(continued)

* Continuation:-

$f(n) + O(f(n)) = O(f(n))$

if $h(n) = O(f(n))$, then.

$\lim_{n \to \infty} \frac{h(n)}{f(n)} = 0$, there exists n_0 such that for all $n \geq n_0$

we have $h(n) \leq f(n)$.

Also $f(n) = O(g(n))$ means

$\lim_{n \to \infty} \frac{|f(n)|}{g(n)} \to +\infty$

while $f(n) = \Omega g(n)$. means

$\lim_{n \to \infty} \frac{|f(n)|}{g(n)} > 0$.

∴ We can prove that

$f(n) + o(f(n)) = \Theta(f(n))$ if the function.

$f(n)$ is asymptotically positive

3- Answer the following with regards to a correctness-proof outline for the Fractional Knapsack algorithm.

a. Assume $x_1, x_2, \ldots, x_n$ is an ordering of the items in decreasing order of profit density (i.e. profit per unit weight). Let f_i denote the fraction of item i that the FK-algorithm adds to the knapsack, $i = 1, 2, \ldots, n$. Explain why $f_1 \geq f_2 \geq \cdots \geq f_n$ is a non-increasing sequence of fraction.

Answer: $f_i \geq f_{i-1}$, since the algorithm always adds up as much of an item as possible. Thus, the fraction sequence is in the form $1, \ldots, 1, f, 0, \ldots, 0$, where $f \in [0,1]$. In other words all of item will be added so long as there is enough remaining capacity. This is followed by almost one item for which only a fraction of the item can be added meaning of the knapsack will result in being filled. Therefore all subsequent fractions must equal to 0. 2/2

b. Let $f_1', f_2', \ldots, f_n'$ be a sequence of fractions that optimizes total profit, and assume that $f_i = f_i'$, for all $i < k$, but $f_k \neq f_k'$. Explain why, in this case, it must be true that $f_k' < f_k$. Hint: what is the contradiction in case the opposite was true?

From the above answer, $f_k' > f_k$, means that the algorithm did not add as much of the item x_k as it could have, a contradiction since the algorithm always adds as much of an item as is physically possible and available.

4- Use the substitution method to show that if

$T(n) = 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n^2$ then $T = O(n^2 \lg n)$

Inductive step: let $T(k) \leq ck^2 \log k$, $k < n$, $c > 0$.

$T(n) \leq cn^2 \log n$.

$T(n) = 4T\left\lfloor \frac{n}{2} \right\rfloor + n^2$.

let $\frac{n}{2} = k$.

$\Rightarrow T(n) = 4 \cdot c\left(\frac{n}{2}\right)^2 \log \frac{n}{2} + n^2 \leq cn^2 \log n$.

$= cn^2(\log n - 1) + n^2 \leq cn^2 \log n$

$n^2(c(\log n - 1) + 1) \leq cn^2 \log n$

$c \log n - c + 1 \leq c \log n$

$c \geq 1$

$\therefore c \geq 1$

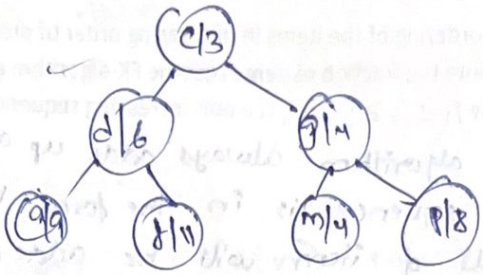
$\therefore T(n) = O(n^2 \log n)$ for any $c \geq 1$

$c \log n \cdot c \leq c \log n \Rightarrow c \geq 0$

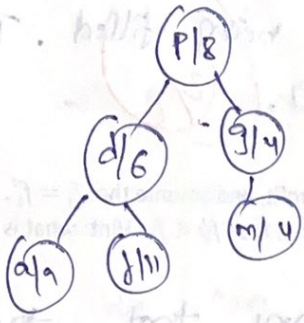
2/2

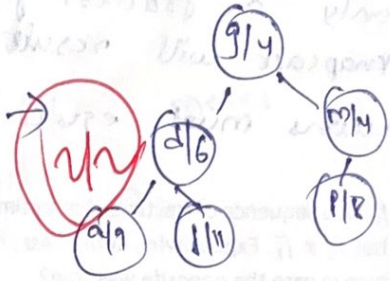
(5) Given edges of G are:

(b, c, 3), (c, e, 7), (c, f, 3), (c, P, 2)

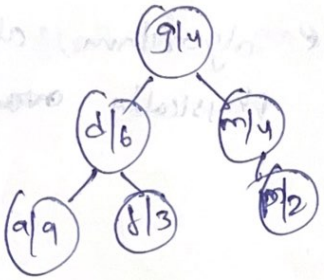


we pop c, then the heap becomes as follows.

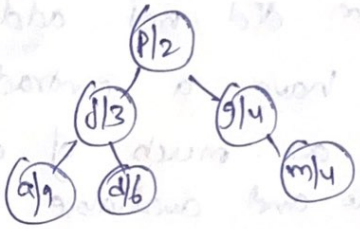




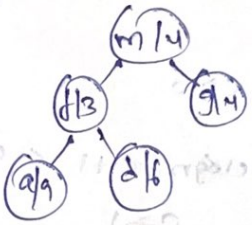
Then we update edge weights.



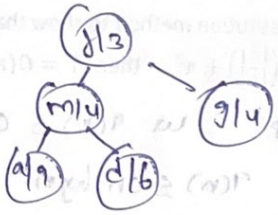
we rearrange →



pop P, then heap looks like



reassaye →



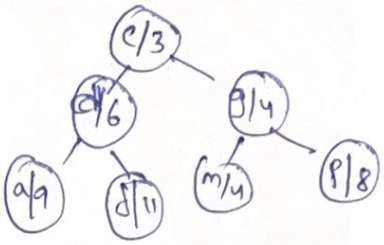
~~we update the edges reassaye~~

∴ The plausible state of the heap after the end of the round is as above

5- The tree below shows the state of the binary-min-heap at the beginning of some round of Prim's algorithm, applied to some weighted graph G. If G has edges:

$$(b, c, 3), (c, e, 7), (c, f, 3), (c, p, 2)$$

Then draw a plausible state of the heap at the end of the round.



6- You are given a graph G each of whose edges is either red or blue and an integer parameter $k \geq 0$. Your task is to develop an algorithm that decides whether G has a spanning tree with exactly k red edges. (The algorithm does not have to find such a spanning tree. This can be done and is not particularly hard but requires more time to figure out than you have in this exam.) The running time of your algorithm should be $O(n + m)$, where n is the number of vertices of G and m is the number of edges of G. To make your task easier, here are the three parts of the answer you have to figure out:

a. Argue that G has a spanning tree with exactly k red edges if and only if it has a spanning tree with at most k red edges and it has a spanning tree with at least k red edges.

b. Argue that a minimum spanning tree of a graph whose edges have weight 0 or 1 can be found in $O(n + m)$ time.