AD - assignment 1

Determine the asymptotic behavior of the following three CPSBC pricing scheme using the master theorem.

Task 1

1. $p(n) = 8p(n/2) + n^2$

For this recurrence, we have $a = 8, b = 2, f(n) = n^2$, and thus we have that $n^{\log_2(8)} = n^3 = \Theta(n^3)$. Since $f(n) = O(n^{3-\varepsilon})$, where $\varepsilon = 1$, we can apply case 1 of the master method. Hence

$$p(n) = \Theta(n^3)$$

2. $p(n) = 8p(n/4) + n^3$

For this recurrence, we have $a = 8, b = 4, f(n) = n^3$, and thus we have that $n^{\log_4(8)} = n^{3/2} = \Theta(n^{3/2})$. Since $f(n) = \Omega(n^{3/2+\varepsilon})$, where $\varepsilon = 3/2$ case 3 applies if we furthermore can show that $af(n/b) \leq cf(n), \forall n >= n_0, c < 1$. So

$$\begin{aligned} af(n/b) = 8f(n/4) = 8\left(\frac{n}{4}\right)^3 &= \frac{8}{4^3}n^3 \\ \leq cn^3 = cf(n) \end{aligned}$$

where the inequality hold for all $n > 1$ and for $c > 8/4^3$. Hence we can easily pick a $c < 1$ and the regularity condition holds so

$$p(n) = \Theta(n^3)$$

3. $p(n) = 10p(n/9) + n\log_2 n$

For this recurrence, we have $a = 10, b = 9, f(n) = n\log_2 n$. Since $\log(n) = o(n^k)$ for $k > 0$ and $\log_9(10) > 1$ it follows that $f(n) = O(t)$ we are in case 1 of the master theorem, hence

$$p(n) = \Theta(n^{\log_9(10)})$$

Task 2

For the following price schemes use the substitution method.

1. $p(n) = p(n/2) + p(n/3) + n$

The recursion tree can be seen in Figure 1 for the first few levels. The longest path from the root to a leaf is to follow the left branch. $n \to n/2 \to n/2^2 \to \cdots \to 1$. Hence the height of the tree is a most $\lg(n)$. The work done at level i is $(1/2 + 1/3)^i n = (5/6)^i n$ which is decreasing. If the tree was completely binary there would be at most $2^{\log_2(n)} = n$ leaves each with a cost of 1, hence all the leaves would at most be done in $O(n)$ time. We expect the solution to the recurrence to be at most the height of the tree times the work at the worst level in the tree. Hence we guess that $p(n) = O(n\log_2(n))$. To show it we use the substitution method (without showing the initial step). So assume $p(n) \leq cn\log_2(n) - n$

$$\begin{aligned} p(n) &= p(n/2) + p(n/3) + n \\ &=\leq c\frac{n}{2}\log_2(n/2) - \frac{n}{2} + c\frac{n}{3}\log_2(n/3) - \frac{n}{3} + n \\ &\leq c\frac{n}{2}\log_2(n/2) - n + c\frac{n}{3}\log_2(n/3) \\ &= (5/6)cn\log_2(n) - n - c\frac{n}{2}\log_2(2) - c\frac{n}{3}\log_2(3) \\ &\leq cn\log_2(n) - n \end{aligned}$$

where the inequalities hold for $c > 0$. Hence

$$p(n) = O(n\log_2(n))$$

2. $p(n) = \sqrt{n}p(\sqrt{n}) + \sqrt{n}$

When describing the recurrence tree that we use to create a guess for the asymptotic cost, we will neglect cases where n is not a power of two. For level i of the tree, the number of nodes will be the number of nodes in the previous levels time the square root of the current level. Hence the number of nodes on level i is $\prod_{j=0}^{(} i-1)n^{1/2^{(}j+1)}$, the cost of each node in level i of the tree is $n^{1/2^{(}i+1)}$. So the the total cost of each level i in the tree is

$$\prod_{j=0}^{i-1} n^{1/2^{j+1}} n^{1/2^{i+1}} = \prod_{j=0}^{i} n^{1/2^{j+1}} = \prod_{j=1}^{i+1} n^{(1/2)^j} = n^{\sum_{j=1}^{i+1}(1/2)^j} \leq n$$

So the cost of all nodes in a level is a most n. Our guess are again that the asymptotic behavior is a most the height of the tree times the cost of the worst level in the tree. So lets try to determine the height of the tree. Note that after each recursive call the square root is taken, hence after k calls the size is $n^{1/2^k} = 2^{\log_2(n)/2^k}$. Solving this for a size 2, yields

$$2^{\log_2(n)/2^k} = 2$$

taking $\log_2$

$$\frac{\log_2(n)}{2^k} = 1$$

multiply by 2^k

$$\log_2(n) = 2^k$$

taking $\log_2$

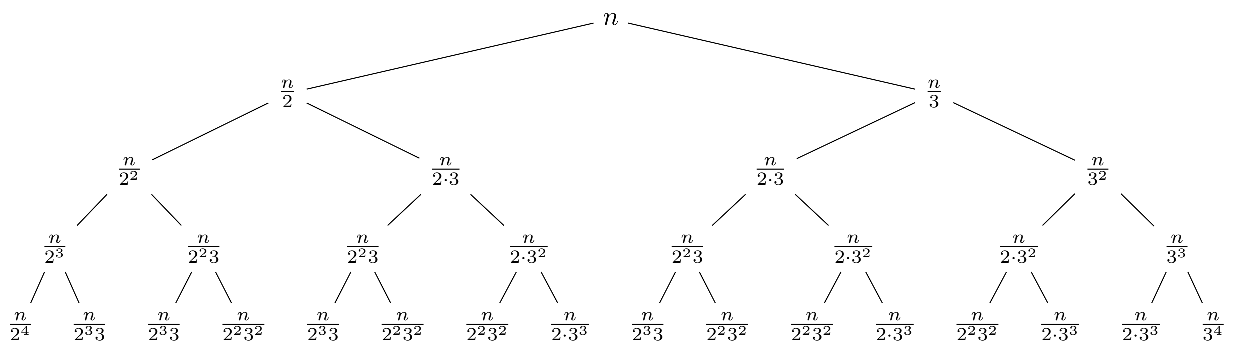
$$k = \log_2(\log_2(n))$$

So our guess to use in the substitution method is that $p(n) \leq n \log_2 \log_2(n) - d$. So

$$\begin{aligned} p(n) &= \sqrt{n}p(\sqrt{n}) + \sqrt{n} \\ &\leq \sqrt{n}\left(\sqrt{n}\log_2 \log_2(\sqrt{n}) - d\right) + \sqrt{n} \\ &\leq n \log_2 \log_2(n) - d\sqrt{n} + \sqrt{n} \end{aligned}$$

The last line in the equation above, hold if there exist a lower order term $d > 0$ where $d\sqrt{n} + \sqrt{n} = d$. Solving for d yields $d = \frac{\sqrt{n}}{\sqrt{n}-1}$ which is indeed a lower order term, larger than 0 from $n > 2$. Hence

$$p(n) = O(n \log_2 \log_2(n))$$



Figur 1: Three structure for the recurrence $p(n) = p(n/2) + p(n/3) + n$. The node indicates how much work need to be done on that node.