AD A1

Casper Bresdahl whs715
Torben Olai Milhøj vrw704
Sarah Willumsen zql291

Contents

1 Todo List

Todo list

2 Task 1

2.1 1

$p(n) = 8p(\frac{n}{2}) + n^2$ Vi bruger sætning 1 side 94 og får, at $a = 8$, $b = 2$ og $f(n) = n^2$. Dvs. vi har:

$$n^{log_b(a)} = n^{log_2(8)} = \Theta(n^{log_2(8)}) = \Theta(n^3)$$

Fordi $n^2 = O(n^{3-\varepsilon})$ hvor $\varepsilon \leq 1$ for alle $n \geq 0$, gælder det pr. sætning 1, at $p(n) = \Theta(n^3)$.

2.2 2

$p(n) = 8p(\frac{n}{4}) + n^3$ Vi bruger sætning 3 side 94 og får, at $a = 8$, $b = 4$ og $f(n) = n^3$. Dvs. vi har:

$$n^3 = \Omega(n^{log_4(8)+\varepsilon}) = \Omega(n^{\frac{3}{2}})$$

hvilket gælder for $\varepsilon \leq \frac{3}{2}$ og for alle $n \geq 0$. Derudover skal det gælde, at $8(\frac{n}{4})^3 \leq cn^3$ for $c < 1$ for alle $n \geq n_0$. Vi omskriver:

$$8(\frac{n}{4})^3 = 8 \cdot \frac{1}{4} \cdot n^3 = 2n^3$$

Dvs. for $c \geq 2$, da vil $2n^3 \leq cn^3$. Dvs. at $p(n) = \Theta(n^3)$.

2.3 3

$p(n) = 10p(\frac{n}{9}) + nlog_2(n)$ Vi bruger sætning 1 side 94 og får, at $a = 10, b = 9$ og $f(n) = nlog_2(n)$. Dvs. vi har:

$$nlog_2(n) = O(n^{log_9(10)-\varepsilon}) = O(n^{1.048-\varepsilon})$$

hvilket gælder for alle $\varepsilon \leq 0.48$ og for alle $n \geq 0$. Dette afledes af den generelle regel, at $log_a(x) = O(x^b)$ for $a > 1$ og $b > 0$. Fordi $nlog_2(n) = O(n^{1.048-\varepsilon})$ for $\varepsilon \leq 0.48$, gælder det pr. sætning 1, at $p(n) = \Theta(n^{1.048})$.

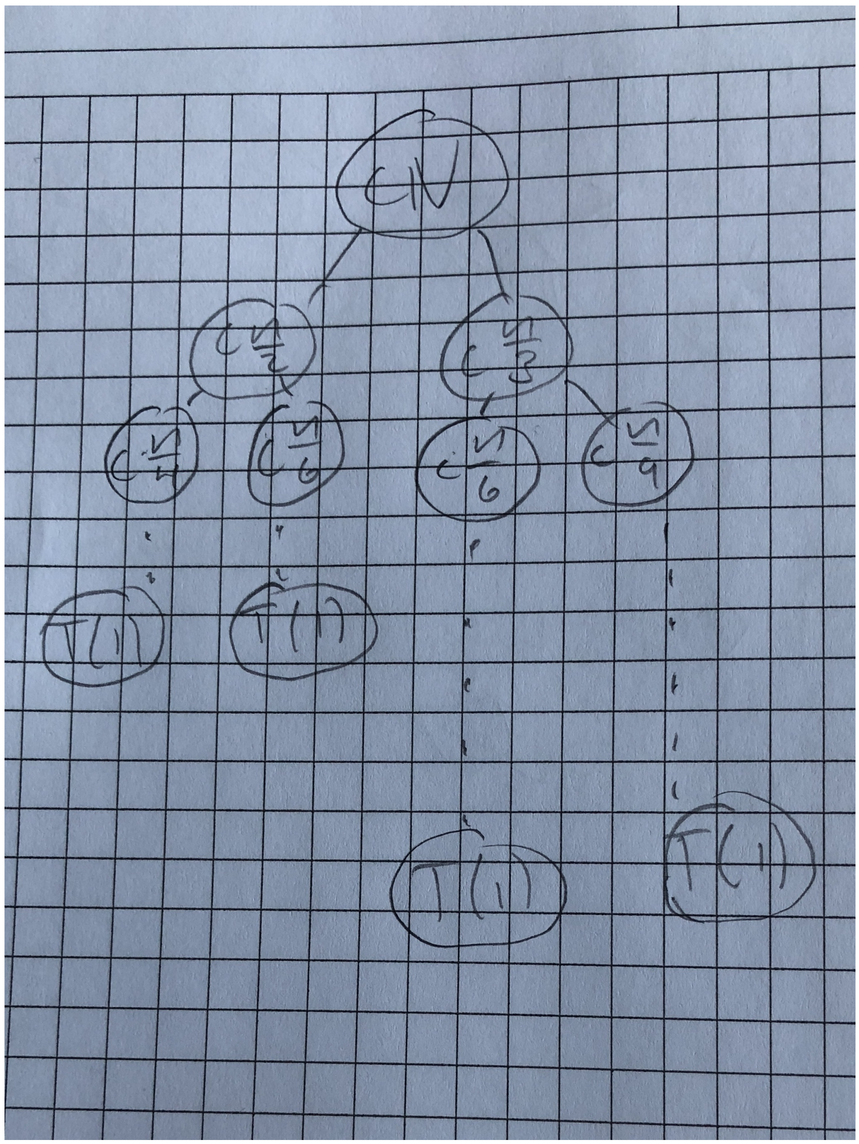
3 Task 2

3.1 1

Vi bliver bedt om at løse nedenstående rekursion ved substitution.

$$p(n) = p(\frac{n}{2}) + p(\frac{n}{3}) + n$$

For at komme frem til et gæt på en løsning vil vi tegne rekursionstræet:



Som det ses på træet, så har vi en geometrisk serie. Denne kan omskrives til en sum som følger[1]:

$$
\begin{aligned}
T(n) &= \left(cn\left(\frac{5}{6}\right)^0 + cn\left(\frac{5}{6}\right)^1 + \cdots + cn\left(\frac{5}{6}\right)^{log_2(n)} + cn\left(\frac{2}{6}\right)^{log_2(n)+1} + \cdots + cn\left(\frac{2}{6}\right)^{log_3(n)}\right) \cdot log(n) \\
&\leq \left(cn\left(\frac{5}{6}\right)^0 + cn\left(\frac{5}{6}\right)^1 + \cdots + cn\left(\frac{5}{6}\right)^{log_3(n)}\right) \cdot log(n) \\
&= \sum_{i=0}^{log_3(n-1)} \left(\frac{5}{6}\right)^i cn \cdot log(n) \\
&\leq \sum_{i=0}^{\infty} \left(\frac{5}{6}\right)^i cn \cdot log(n) \\
&= \frac{1}{1-\frac{5}{6}} cn \cdot log(n) \\
&= 6n \cdot log(n) \\
&= O(nlog(n))
\end{aligned}
$$

Vi har nu et gæt og kan gå i gang med vores bevis.
Vi vil nu gerne vise at for $P(n) = P(n/2) + P(n/3) + n$:

$$P(n) \leq cnlog(n)$$

Vi kan nu begynde beviset ved at substituere ind:

$$
\begin{aligned}
P(n) &\leq c(n/2)log(n/2) + c(n/3)log(n/3) + n \\
&\leq c(n/2)log(n/2) + c(n/3)log(n/2) + n \\
&\leq cnlog(n/2) + n \\
&= cnlog(n) + cnlog(2) + n
\end{aligned}
$$

Vi kan nu se at hvis $cnlog(2) + n \geq 0$, så holder vores ulighed. Ved at løse for c finder vi:

$$
\begin{aligned}
cnlog(2) + n &\geq 0 \\
cnlog(2) &\geq -n \\
c &\geq \frac{-1}{log(2)}
\end{aligned}
$$

Vi har dermed bevist at for $c \geq \frac{-1}{log(2)}$ holder vores ulighed, og er derfor $P(n) = O(nlog(n))$.

Usikker på om uligheden skal vendes

3.2 2

$p(n) = \sqrt(n)p(\sqrt(n)) + \sqrt(n)$ Vi omskriver $p(n)$ til noget simplere ved at lade $m = lg(n)$:

$$p(2^m) = 2^{\frac{m}{2}} p(2^{\frac{m}{2}}) + 2^{\frac{m}{2}}$$

[1] Introduction to algorithms - s. 90

Dernæst definerer vi en ny reccurence givet ved $s(m) = p(2^m)$:

$$s(m) = 2^{\frac{m}{2}} p(\frac{m}{2}) + 2^{\frac{m}{2}}$$

Vi genkender, at $s(m)$ har løsningen $s(m) = O(2^m lg(m))$. Dvs:

Reference til bogen?

$$p(n) = p(2^m) = s(m) = O(2^m lg(m)) = O(nlg(lg(n)))$$

4 Task 3

```
Sort(A)
    maxLen = 2 log (A.Len)
    IntroSort(A, maxLen)
    return(A)

IntroSort(A,maxDepth)
    n = A.Len
    if (maxDepth == 0)
        HeapSort(A)

    elif(n < c)
        InsertionSort(A)

    else
        p = RandomPartition(A)
        IntroSort(A[0:p], maxDepth-1)
        IntroSort(A[p+1:n], maxDepth-1)
```

5 Task 4

Vi er blevet bedt om at redegøre for køretiden af introsort.

Heapsort-sort kører i worst-case O(n log n). Insertion-sort kører i worst-case $O(n^2)$, men da vi kun kører insertion-sort når $j - i = n < c$ hvor c er en konstant som typisk er 16 eller 32, så vil insertion-sort i forhold til de andre algoritmer køre i konstant tid. Derfor er worst-case køretiden for insertion-sort i vores algoritme konstant dvs. $O(1)$.
Ved hver opsplitning i sub-arrays, i quick-sort, bliver der udført en operation på hvert element (sammenligning med pivot element) i hver af de nye under arrays, dvs. at der ved hvert niveau er n-operationer i alt. Da vores recursions dybde maks vil være 2log n, og der på hvert niveau er n-operationer, vil quick-sort i worst-case køre O(2n log n).

Vi har derfor at worst-case køretiden for intro-sort er O(n log n).

6 Task 5

Vi bruger heap-sort snarere end en anden $O(\text{n log n})$ sorterings-algoritme, fordi heap-sort er en *in-place sorterings algoritme*, dvs. den omarrangerer tallene inden for arrayet, så der, til en hver tid kun er lagret et konstant antal af array elementerne udenfor input arrayet, modsat f.eks. merge-sort som ikke sorterer *in-place*.