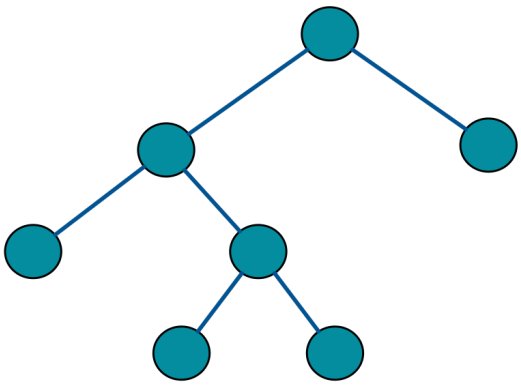
Lab 2: Part 2



Group 26
Klas Mannberg klaman-8@student.ltu.se
D0012E Algorithms and data structures

LULEÅ UNIVERSITY OF TECHNOLOGY

January 9, 2020

Contents

1 Introduction

We try to construct a divide and conquer algorithm that returns the 4th element from a list. The algorithm should take at most linear time. After the algorithm is constructed into a recurrence algorithm we assume it runs in linear time or less followed by proving it using two different methods.

2 Theory

Divide and conquer algorithm, assuming input array has to be atleast of length 4.
Step 1: We **divide** the array in constant time $\mathcal{O}(1)$
Step 2: we **conquer** the array with recursive calls taking a time of $4T(\frac{n}{4})$ since we choose to divide into 4 length sub-arrays.
Step 3: We **combine** the sorted arrays taking a constant of 4 comparisons since its the set size for each sub-array. $\mathcal{O}(4)$
The combined worst case cost of this algorithm (Denoted $T(n)$) when $n = 2^k$ where k is positive and $n \geq 4$ is the following recursive formula:

$$T(n) = \begin{cases} 6, & \text{if } n = 4. \\ 4T(\frac{n}{4}) + 4, & \text{if } n > 4. \end{cases} \tag{1}$$

Lets try to prove that $T(n) <= n$:
Method 1: Subsitution Lets assume the algorithm runs in linear time or less:
$T(n) \leq cn$.
$T(n) = 4T(\frac{n}{4}) + 4 = 4(4T(\frac{n}{16}) + 4 + 4 = 16T(\frac{n}{16}) + 16 + 4 = 4(16T(\frac{n}{16}) + 20) + 4 = 64T(\frac{n}{64}) + 80 + 4$
Were starting to see a pattern here: $4^kT(\frac{n}{4^k}) + 4^k + 4^{k-1} + ... + 4^{k-n} = 4^kT(\frac{n}{4^k}) + \sum_{i=1}^{k} 4^i$ How do we get $\frac{n}{4^k} = 4$ for the base case? By setting $\frac{n}{4} = 4^k$ and $log\frac{n}{4} = k$ and using gemoteric sum $\sum_{i=1}^{k} 4^i = \frac{4}{3}(\frac{n}{4} - 1)$ which gives us $4^{log4(\frac{n}{4})}T(4) = \frac{n}{4} * 6 + \frac{4}{3}(\frac{n}{4} - 1) = \frac{1}{6}(11n - 8)$ which is our exact number of comparisons. Big-O is $\mathcal{O}(n) \leq cn$

Since the base case is correct: $\frac{1}{6}(11 * 4 - 8) = 6$ if we assume the sub-lists created are sorted correctly by chosen sorting algorithm. And also we assume that the merge operation works since its getting two sorted arrays and hence should return a combined sorted array. If the previous steps work correctly it must then follow that the resulting array after merge step is also correctly sorted into the four smallest elements from the input array.
Method 2: Master's Theorem
$T(n) = aT(\frac{n}{b}) + f(n)$ when applied to our $T(n)$ we get $a = 4, b = 4, f(n) = 4$.
$n^{\log_b a} = n^{\log_4 4} = n^1 = n$. Now this does not equal our $\mathcal{O}(f(n))$ unless we **remove** 1, we have $\epsilon = 1$ for $\theta(n^{\log_b a - \epsilon}) = \theta(n^{\lg_4 4 - 1})$ which then confirms that $\theta(n^{1-1}) = \theta(n^0) = \theta(1) = \mathcal{O}(f(n))$ which means that case 1 applies and $T(n) = \theta(n^{log_4(4)}) = \theta(n^1) = \theta(n) \implies$ linear time.