Illinois Institute of Technology
Department of Computer Science

Homework 2 Solutions

CS 430 Introduction to Algorithms
Spring Semester, 2014

1. **Problem 1 Solution:**

   (a) $T(n) = T(\log n) + 3$
   $= T(\log\log n) + 3 + 3$
   $= T(\log\log\log n) + 3 + 3 + 3$
   on the ith step we have
   $= T(\log\log .... \log n) + 3i$
   $= log(k)n + 3k$
   Here, k is the number of times the recurrence occurs before the size of the problem becomes 1, and $log(k)n$ represents $\log\log .... \log n$ k times
   $= O(log^* n)$
   $log^* n$ is the iterated log of n, which is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1

   (b) $T(n) = T(\log n) + 3n$
   $= T(\log\log n) + 3\log n + 3n$
   $= T(\log\log\log n) + 3\log\log n + 3\log n + 3n$
   on the ith step we have
   $= T(\log\log .... \log n) + 3(\log\log .... \log n) + ... + 3(\log n) + 3n$
   $= O(n)$

2. **Problem 2**. **Solution:**
   $T(n) = T(n - c) + T(c) + f(n)$

   (a) $T(n) = \log\log(n) + [\log\log(n - c) + c] + [\log\log(n - 2c) + c] + ... + [\log\log(n - lc) + c]$
   $= T(n) = \log\log(n) + \log\log(n - c) + \log\log(n - 2c) + ... + \log\log(n - lc) + lc$
   Here l is the number of times the recursion occurs.
   $= O(n \log\log n)$

   (b) $T(n) = \sqrt{n} + [\sqrt{n - c} + c] + [\sqrt{n - 2c} + c] + ... + \sqrt{n - lc} + c$
   $= \sqrt{n} + (k - 1).c + \sqrt{n - c} + \sqrt{n - 2c} + .. + \sqrt{n - lc}$
   $= O(n\sqrt{n})$

   $T(n) = T(n) = 2T(n/2) + f(n)$

   (a) This recursion can be solved using Master's theorem:
   $f(n) = \log\log n$
   $af(n/b) = 2\log\log n/2 = 2\log(\log n - \log 2)$
   $= 2\log(\log n(1 - \frac{1}{n}))$
   $= 2\log\log n + 2\log(1 - \frac{1}{\log n})$
   Using taylor series expansion on $2\log(1 - \frac{1}{\log n})$ and taking the first term we get
   $= 2\log\log n - \frac{2}{\log n}$
   Using Master's theorem:
   $T(n) = \theta(n^{log 2}) = \theta(n)$

(b) This recursion can be solved using Master's theorem:
$f(n) = \sqrt{n}$
$af(n/b) = 2\sqrt{n/2} = \sqrt{2}\sqrt{n}$
Using Master's theorem:
$T(n) = \theta(n^{log2}) = \theta(n)$

3. **Problem 3**. **Solution:** Best Possible partititon is $\frac{n}{3}$ So the tree will be of degree 3i.e each node will have 3 children. The height of the tree
$i = \log_3 n$ As we keep increasing the number of partitions the tree height decreases but the contant factor increases as the the number of children increase(we will have 2 comaparisions at each level instead of one). So it is not better than a binary tree. Thus time complexity will be $O(n \log_3 n)$ Algorithm will be the same in the book except we will be partitioning the array into 3 parts instead of 2 in every step.

4. **Problem 4**. **Solution:** $(a + bi)(c + di)$
Let $product_1 = (a - b)(c + d)$,
$product_2 = bc$ and
$product_3 = ad$
So $product_1 + product_2 - product_3 = ac - bd$
and $product_2 + product_3 = ad + bc$
so we get
$(ac - bd) + (ad + bc)i$
As we can see above there are 3 multiplications required at each step.

5. **Problem 5**. **Solution:** Let M_w be the maximum number of leaf nodes of cost w.
cost(T)=$ma + nb$, where m and n are non-negative integers. Then we have

$$M_w = M_{w-a} + M_{w-b}$$