please grade

CSE 548 – Analysis of Algorithms, Spring 2013

Assignment #3

Duo Xu (#108662210)

partner: Yu-Yao Lin (#109090793)

<u>Problem 1</u>

(a) T

As We multiply the weight of each in edge in G by a positive, non-zero constant, assuming we have already got all the paths from v to u, now every path's total weight will be multiplied by this constant, thus the shortest path, which has the smallest weight, still has the smallest weight.

(b) F

$f(n) = o(g(n))$ means for sufficiently large n, $f(n) < Cg(n)$ for every fixed positive number C.

$f(n) = \Theta(g(n))$ means for sufficiently large n, $C1g(n) < f(n) < C2g(n)$ for some fixed positive number $C1$ and $C2$.

The above two definitions are inconsistent.

(c) F

$2^{\log^2 n} = n^{\log n}$, thus when n is sufficiently large, the statement will be false.

(d) T

$(\log n)^{2\log n/\log\log n} = (\log n)^{\log_{\log n} n^2} = n^2$

(e) T

Let $t = \sqrt[9]{n}$, then $\log n = 9\log t$. It is obvious that $t = \Omega(9\log t)$.

Problem 2

(a) Each time we divide the existing length-n string into two sub-strings with length $n/2$ until the sub-strings are of length 1.

Then we merge two by two, sorting the two sub-strings of length $n/2$ into a length-n string by using reversals.

See (c) as an example.

(b) $T(n) = 2T(n/2) + O(n)$

$T(n) = n \log n$

(c) 01101000

01010100

00110001

00000111

Problem 3

$T(n) = 2T(n/2) + O(\sqrt{n})$

$T(n) = 2(2T(n/4) + O(\sqrt{n/2})) + O(\sqrt{n})$

$T(n) = 2^2(2T(n/8) + O(\sqrt{n/4})) + O(\sqrt{2n}) + O(\sqrt{n})$

$T(n) = 2^3(2T(n/16) + O(\sqrt{n/8})) + O(\sqrt{4n}) + O(\sqrt{2n}) + O(\sqrt{n})$

...

$T(n) = n + O(\frac{n}{2}\sqrt{2}) + ... + O(\sqrt{4n}) + O(\sqrt{2n}) + O(\sqrt{n})$

$T(n) = O(n)$

Problem 4

(a) $T(m) = O(1) + O(\sqrt{2}) + O(\sqrt{4}) + O(\sqrt{8}) + ... + O(\sqrt{m}) = O(\sqrt{m})$ where $m/2 < n \leq m$

1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	1

The boxes with value 1 stands for checking positions, if the current box has value 1, stop. Otherwise double the checking index and check the next possible position.

(b) $T(m) = O(\sqrt{m} \log m)$ where $m/2 < n \leq m$

0	0	0	0	0	0	0	1	0	3	4	2	0	0	0

We use binary search between the $\frac{m}{2}$th and mth positions. Every time we check the current position, if it is 1 and its previous neighbour is 0, this is the exact value of n. Otherwise we continue searching.

The above is an example. The numbers in boxes (1-4) show the search order.

As it has $\frac{m}{2}$ numbers, so it has $O(\log m)$ levels in recursion tree, and in each level the probe cost $O(\sqrt{m})$. The total cost is $O(\sqrt{m} \log m)$.

Problem 5

(a) $T(k, N) = T(k/2, N) + O(\log^2 N)$

We use divide and conquer. Thus $M^k modN = [(M^{\frac{k}{2}} modN)(M^{\frac{k}{2}} modN)]modN$. In each level, the cost is multiplying two $O(\log N)$-bit numbers, which takes time $O(\log^2 N)$.

(b) $T(k, N) = O(\log k \log^2 N)$

In the recursion tree, there are $\log k$ levels, and in each level it costs $O(\log^2 N)$.

Problem 6

Recursive relation: $T(N) = T(N/2) + 1$

Base case of recursion: $T(N) = 1$ when $N \leq B$

Cost of recursion: $T(N) = O(\log \frac{N}{B})$

N is the number of elements in array $\mathcal{D}$, and B is the size of the block in one transfer.