COMP9101 ASS1

YIZHENG YING   Z5141180

1. I will use Merge sort algorithm to sort the array because the complexity is O(n log n). For one query earch L with binary search and the complexity is O(log n) and the same search for R. So for n queries the total complexity is n(O(log n)+O(log n)). So the complexity is O(nlog n).

2. (a) Firstly we sort the array of number by Quick sort in ascending order and complexity is O(nlogn).

Then take two element which index is i and j, i =0 and j =n-1 and sum the two elements. If the result r greater than x then keep i and decrease j by 1. If r is smaller than x then keep j and increase i by 1. If r=x then increase i by 1 and decrease j by 1 and sum the new elements again and compare with x. When i>j then stop it.

So the complexity is O(nlogn).

(b)I use hashmap. From the first number to the last number, if the number N in hashmap then return N and x-N, otherwise add N to hashmap. This algorithm time complexity is O(n).

3. We can ask the question: If X knows Y and if the answer is YES so Y could be one candidate but X could not. And if answer is NO so X could be a candidate.

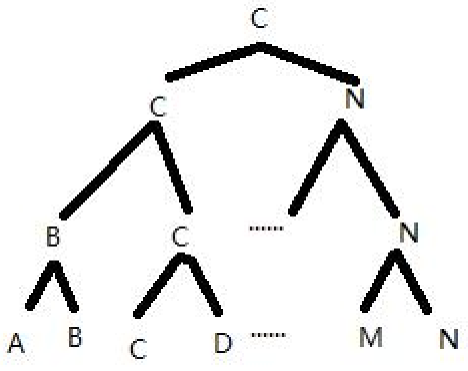
(a) Firstly the first person asks the second person the question in order to find the candidate. Then the candidate asks another person to choose the new candidate and so on. This question could be asked (n-1) times.

Secondly, ask the rest person if they know the candidate. If anyone answers NO, there could be none candidate. If all person answers are YES so the person could be the candidate. This could cost (n-1) times.

Thirdly, ask the candidate if he knows each other from the rest people. If any answer isYES, there could be none candidate. If all answers are NO so the person mut be the candidate. It will cost (n-1) times.

So the question will be asked no more than (3n-3) times.

(b) Make people as leaves of a tree and each node has 0 or 2 children.Like the follow construction.



It will take n-1 questions to get the candidate C.Then ask the rest of people whether they know candidate C or not, it will take n-1times. Then ask candidate C whetehr he konws any rest of people or not. If he knows any one so he is not candidate. If he does not know any one he is the candidate. Logn questions has been asked before so it will take $n-1$-$\lfloor \log n \rfloor$ times.So the total time is no more than $3n-\lfloor \log n \rfloor$-$3$.

4. (1) $f(n)=(\log_2 n)^2,\ \ g(n)=\log_2(n^{\log_2 n})+2\log_2 n$, we let $\log_2 n$ equals to x so

$f(n)=x^2,\ \ g(n)=\log_2(n^x)+2x=x\log_2 n+2x=x^2+2x$,

$f(n)\le g(n)\le 2f(n)$ so $f(n)=\Theta(g(n))$

(2) $f(n)=n^{100},\ \ g(n)=2^{\frac{n}{100}}$, because g(n) is a monotonically

decreasing function and f(n) is an increasing function,so $n^{100}\ge 2^{\frac{n}{100}}$

whatever n is, so $f(n)=\Omega(g(n))$

(3) $f(n)=\sqrt{n},\ \ g(n)=2\sqrt{\log_2 n}$,

$\lim_{n\to\infty} f(n)=\infty, \lim_{n\to\infty} g(n)=\infty, so$

$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{f'(n)}{g'(n)}=0$ so $f(\mathrm{n})<g(n)$ so

$f(n)=O(g(n))$

(4) $f(n)=n^{1.001},\ \ g(n)=n\log_2 n$, $f(n)=n\bullet n^{0.001}$ so

$f1(n)=n^{0.001}$ and $g1(n)=\log_2 n$,

$f1'(n)=1/(1000n^{0.999}), g1'(n)=1/(n\ln 2)$, so $g1'(n)>f1'(n)$ so g(n) growth rate is larger than f(n), so $f(n) = \mathrm{O}(g(n))$

(5) $f(n) = n^{(1+\sin(\pi n/2))/2}$, $\mathrm{g}(n) = \sqrt{n}$, for $\sin(\pi n/2) \in [-1,1]$, $f(n)$ is oscillation function so it is not comparable.

5. (a) $T(n) = 2T(n/2) + n(2 + \sin n)$

a = 2 and b = 2, $n^{\log_b a} = n^{\log_2 2} = n$, $f(n) = 2n + n\sin n \in [n, 3n]$,

When $T(n) = 2T(n/2) + n$, $f(n) = n, n^{\log_b a} = n$ so it matches case2.

When $T(n) = 2T(n/2) + 3n$, $f(n) = 3n, n^{\log_b a} = n$ it matches case2.

so $T(n) = \Theta(n \log_2 n)$

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

a = 2 and b = 2, $n^{\log_b a} = n^{\log_2 2} = n$,

$f(n) = \sqrt{n} + \log n = \mathrm{O}(n^{\frac{1}{2}}) < O(n^{\log_b a - \varepsilon})$, so for master theorem case1, when ε >0 then $f(n) = O(n^{\log_b a - \varepsilon})$, so

$T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$

(c) $T(n) = 8T(n/2) + n^{\log n}$

a = 8 and b=2, $n^{\log_b a} = n^{\log_2 8} = n^3$, $f(n) = n^{\log n}$,

Because $\mathrm{O}(n^{\log n}) > \mathrm{O}(n^3)$, it does not meet case1 .

Because $f(n) \neq \mathrm{O}(n^3)$, it does not meet case2 .

$af(n/b) \le cf(n), 8f(n/2) \le cf(n)$,

$8(n/2)^{\log n - \log 2} \le 8(n/2)^{\log n} \le 8n^{\log n}$, so it is case3,

$T(n) = \Theta(n^{\mathrm{logn}})$

(d) $T(n) = T(n-1) + n$

$T(n) = T(n-1) + n = T(n-2) + n - 1 = T(n-3) + n - 3$

$T(1) = T(0) + 1, T(0) = 0$, $\mathrm{T(n)} = 1 + 2 + 3 + ... + n = n(n-1)/2$ So

$T(n) = \mathrm{O}(n^2)$