Problem1

Substitution method

1)

IH: Assume $T(k)>=c(n+2)lg(n+2)$ for all $k<n$

IS:

$T(n)=2T(\lfloor n/2 \rfloor)+n$

$>=2c(\lfloor n/2 \rfloor+2)lg(\lfloor n/2 \rfloor+2)+n$

$>=2c((n/2-1+2)lg(n/2-1+2)+n$

$=2c(n+2)/2lg((n+2)/2)+n$

$=c(n+2)lg(n+2)-c(n+2)lg2+n$

$>=c(n+2)lg(n+2)$ when $0<c<1$

when n=2,T(2)=3>=c2lg2

when n=3 T(3)=9>=c3lg3

pick c=0.5 is enough for all n>1

so c(n+2)lg(n+2) can be the omega, which is cnlgn

so $T(n)= \Omega(n \cdot lg\ n)$

2)

IH: Assume $T(k)<=cnlgn$ for all $k<n$

IS:

$T(n)=2T(\lfloor n/2 \rfloor)+n$

$<=2c(\lfloor n/2 \rfloor lg \lfloor n/2 \rfloor)+n$

$<=2c(n/2lg(n/2))+n$

$=cnlg(n/2)+n$

$=cnlgn-cnlg2+n$

$=cnlgn-cn+n$

$<=cnlgn$ when $c>=1$

when n=2 T(2)=3<=c2lg2

when n=3 T(3)=9<=c3lg3

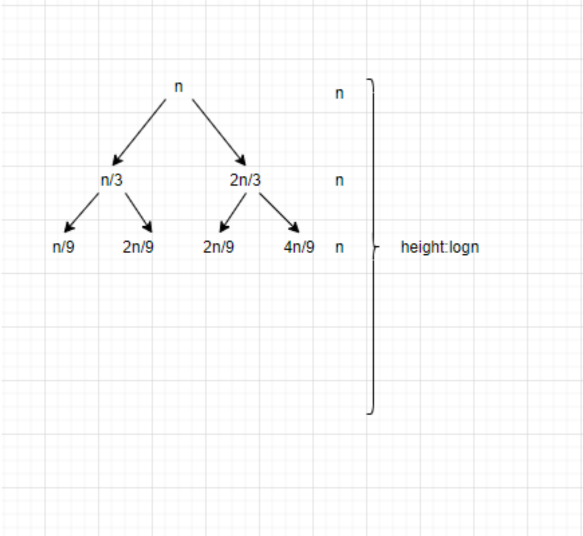
pick c=3, is enough for all n>1

so $T(n) = O(n \cdot lg\ n)$

so $T(n)= \Theta(n \cdot lg\ n)$

Problem2:

Step1:Recursion tree to get reasonable guess



the depth depends on right most branch ,which decreases slowliest.

The height should be $\log_{2/3}n$ =logn

and the sum of each row(recursion)=n

so T(n) =nlogn is a reasonable guess

Step2: Substitution method

1)

IH: Assume T(k)>=cnlgn for all k<n

IS:

T(n)=T(n/3)+T(2n/3)+n

>=c(n/3)lg(n/3)+c(2n/3)lg(2n/3)+n

=c(n/3)lgn-c(n/3)lg3+c(2n/3)lgn+c(2n/3)lg(2/3)+n

=cnlgn+cnlg(2/3)-cnlg3+n

=cnlgn+cnlg(2/9)+n

>=cnlgn when c=1

so T(n)= $\Omega(n \cdot lg\ n)$

2)

IH: Assume T(k)<=cnlgn for all k<n

IS:

T(n)=T(n/3)+T(2n/3)+n

<= c(n/3)lg(n/3)+c(2n/3)lg(2n/3)+n

=cnlgn+cnlg(2/9)+n

<=cnlgn when c=5

so T(n) = $O(n \cdot lg\ n)$

so T(n)= $\Theta(n \cdot lg\ n)$

Problem3:

$f(n)=n^2<n^{\log_2 7}=n^{2.8}$

so it is case 1, $T(n)=\Theta(n^{2.8})$

for A', it should still be case1, then we can get largest integer value

$n^{\log_4 a}<n^{\log_2 7}$

$\log_4 a< \log_2 7$

$\log_4 a< \log_4 49$ //log computation

$a=48$

Problem4:

This is the pseudo code of partition, even in the worst case, which every if statement is true(A[j]<=x for every j), the steps in for loop is still fixed, which can be considered as O(1), and there is only one 'for loop' which depends on array length n, so the running time is Θ(n)

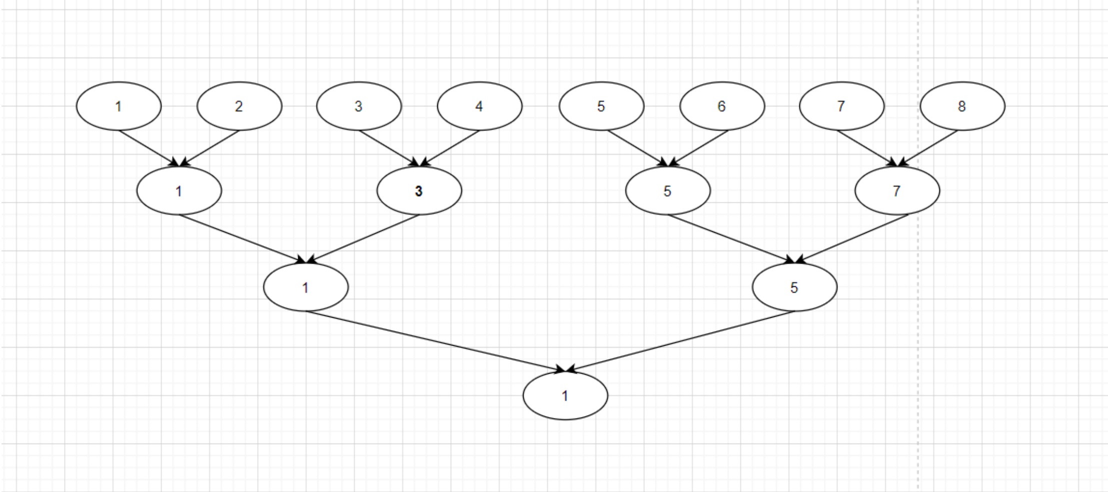
```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Problem 5:

n^2, it can be considered as sorted array, no matter you choose first index or last index as pivot, the partition will always divide the array to n-1 and 0 . Then if you draw a tree for this T(n), the depth of this tree will be n

Problem6

Step1: Determine the minimum as a tournament



$1+2+4+8+\cdots n/2$ //geometric

$=(n/2*2-1)/1=n-1$

Step2:

The only possible second smallest is the elements that have been compared to smallest(1) directly, because if it hasn't been compared to 1, it will compare to other element, if it is bigger than the other one, it will not be the second smallest. If it is smaller, then it will finally be compared to smallest(1)

And at each height, there will be one element compared to the smallest element(1), so the number of alternative second smallest = the depth of tree

//height1: 2 , height 2: 3 , height 3:5

the height $=\lceil \lg n \rceil$ = the number of alternative second smallest

Then at the second tournament, there will be $\lceil \lg n \rceil-1$ comparisons

So there are total $n+\lceil \lg n \rceil-2$ comparisons

Problem7

1/

This is the original formula

$$T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) \text{ if } n > n_0$$

if they are group into groups of 7

There will be ⌈n/7⌉ groups ,we need T(⌈n/7⌉) to choose the median

This is the original elements greater/less than pivot x

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6 .$$

4(⌈1/2*⌈n/7⌉⌉-2)>=2n/7-8

So T(n)<=T(⌈n/7⌉)+T(5n/7+8)+O(n) 到这一步没有大问题

guess T(k)<=cn for all k<=n

这里要把上界拆出来，
n/7
最多小于等于n/7+1

T(n)<=c⌈n/7⌉+5cn/7+8c+O(n)

<=6cn/7+8c+O(n)

if(O(n))<= cn/7-8c, it is still linear

2/

2(⌈1/2*⌈n/3⌉⌉-2)>=n/3-4

So(Tn)<= T(⌈n/3⌉)+T(2n/3+4)+O(n)

cause 1/3+2/3=1

so if you draw a tree, the sum of every height will be >=n

Then the result T(n) will be nlogn

Problem 8

Create a new int array **temp** whose size= max element of (XUY), all elements are 0 by default

Then use 2n to loop these two arrays,take X for example

```
for(int i=0;i<n,i++){
    temp[X[i]]++;
}
```

for example ,if the second element of X is 5, then the data in index 5 of temp will ++, which can count how many 5 in these two arrays.

Then use a for loop to add data of **temp** , if the data=2n/2=n, then the corresponding index will be the median. Which can be done in O(n)

Then the algorithm complexity will be O(n)