1.T,F,F,T, F,T,T

2.

IH: assume that for some c>0,We have $T(k)>=cn^2+n$ for all $k<n$

IS:$T(n)=16T(n/4)-2n$

$>=16c*n^2/16+16n/4-2n$

$>=cn^2+4n-2n$

$>=cn^2+2n$

cause n>0,so $T(n)>=cn^2+2n>cn^2$

Boundary Check: $T(1)=3>=c*1+2$

- ⇨ $0<c<=1$
- ⇨ Thus, with c=0.5 $T(n)>=cn^2+2n$ for all $n>=1$
- ⇨ Thus, $T(n) = \Omega(n^2)$

3.

a)

we just need to pick the first k elements in a1.a2……ak…..an

use a1.a2,….ak and b1.b2,…bk to form a new array

if k=O(1),, Then the size of new array is still O(1)

The time need to sort such array is still O(1)

just sort and pick the k smallest element

b)

pick first k elements a1.a2….ak,

use a1.a2,….ak and b1.b2,…bk to form a new array

Then the size of new array should be O(logn)

Then apply SELECT algorithm ,the Time coplexity will be O(logn)

c) just combine two sets of numbers, and apply SELECT algorithmm to find the kth smallest elements in whole array, the size is still O(n), so the time complexity will be O(n)

1. Divide $n$ elements into $\lfloor n/5 \rfloor$ group of 5 elements, and one group that can have less than 5 elements.

2. Find the median of each group.

3. Use Select recursively to find the median $x$ of the $\lceil n/5 \rceil$ values.

4. Partition the input around $x$.

5. Continue as in the Select algorithm recursively.

4.

STEP1,Firstly, sort the set S and re-label them

STEP2.Create a new empty set K for the unit intervals

STEP3:For each xi in S

    if xi is included in the newest interval of K, then skip

    else if xi isn't included in the newest interval, add a new interval [xi,xi+1] into the set K

STEP4:return S

Proof: The first interval will be [X1,X1+1], assume we use [y,y+1] to include X1, where y <X1, cause X1 is smallest number(S is sorted) , there is no number between [y,x1), we can use [X1,X1+1] to replace [y,y+1].

Then for the next interval is same reason,

[Xi,Xi+1]. No number between [X1+1,Xi),then we can use [Xi,Xi+1] to replac

…………………

It is optimal and greedy because every step we choose the best solution by local information, and the subproblems are the same type as original problem

Time complexity is O(nlogn) because of sort

5.

5.It is still minimum spanning tree
Use Kruskals method for example
Assume the original weighted edges are sorted in an array
X1,X2….Xn
Then use the Kruskals method, we need to test X1,X2 to Xn, if meets cycle, skip

Now it becomes
X1+k,X2+k,…..Xn+k,
The order of this sorted array won't be changed.
So we still use Kruskals to add edges in same order, and cycle will be the old cycle since the location of edge hasn't be changed.

And it will finally be same minimum spanning tree with different weight