1. (1)$\Theta(n^{\log_3 2})$

   (2) $\Theta\left(n^{\log_4 5}\right)$

   (3)$\Theta(nlogn)$

   (4)$\Theta(n^2 logn)$

   (5) $\Theta(n^3 logn)$

   (6) $\Theta(n^{\log_2 7})$

   (7)$\Theta(logn)$

   (8)$\Theta(n^2)$

2 .$IH: T\ (k) \geq dk^2$ k<n and for positive constant d

$$T(n) \geq \max_{0\leq q\leq n-1}(T(q) + T(n-q-1)) + c\cdot n$$
$$\geq \max_{0\leq q\leq n-1}(dq^2 + d(n-q-1)^2) + c\cdot n$$

and when 0≤ $q \leq n-1$, the maximum value of $dq^2 + d(n-q-1)^2$ is $d(n-1)^2$

so $T(n) \geq d(n-1)^2 + c\cdot n$

$$= dn^2 - 2dn + 1 + c\cdot n$$
$$\geq dn^2 \quad \text{if } -2dn + 1 + c\cdot n \geq 0$$

therefore $T(n) = \Omega(n^2)$

3.

At first, i=0,j=13,x=13.

After the first iteration, i=1,j=11,A={6,19,9,5,12,8,7, 4,11,2,13,21}

After the second iteration, i=2,j=10, A={6,2,9,5,12,8,7, 4,11,19,13,21}.

After the third iteration, i=10,j=9, A= {6,2,9,5,12,8,7, 4,11,13,19,21}

4.If the array has many duplicate items, Hoare will be a better partitioning algorithm, because in that case there will be less operations to change two items.

```
5.def    MIN_MAX(A)
    if A.length=1
        return min=max=A[0]
    divide A into two equal subsets A1,A2
    (min1,max1)=MIN_MAX(A1)
    (min2,max2)=MIN_MAX(A2)
    if(min1<min2)
    return min=min1
else return min=min2
if(max1<max2)
    return max=max1
else
    return max=max2
```

T(n)=2T(n/2)+2 and T(2)=1, so T(n)=$\frac{3n}{2} - 2$

6. I think it does not have any flaw. In the closest pair algorithm presented in class, there are at most 8 dots in that area, but the dots in the line are calculated twice, if we reside those two dots to the left area, then we only need to compare with at most five dots.

7.

Regular algorithm: divide the n delegates into several sections, every function calculate (candidate, count), which candidate represents party which has the most delegates in this section, and count is the number of delegates in that party. Then I merge every section in order to find the party which has the most delegates. When I find the majority party, I take one of delegates in that party, let him greet with every delegates then I cound find all members in majority party.

Faster algorithm: I need two variables: candidate and count, candidate represents the party. Count=0. Candidate =first delegate, let candidate greet with the rest of delegates, if they are from the same party, count++, else count--; if count<0, assign candidate into the current delegate and continue. In the end, the candidate's party is majority party, and then let this candidate greet with every delegates to find all members in that party. In that case, the running time is O(n).

8. In this algorithm, I divide the predict price array into two equal section, in every section I find the maximum profit and store the minimum price and maximum price whose difference is the maximum profit, then I merge two sections and calculate the difference between the minimum price of the first section and the maximum price of the second section, the max of the three difference is the maximum profit.

9.Like the algorithm in the lecture, firstly divide the area into two pieces, then recursively find the closest pair in both pieces, then find if the dots pair which are in different pieces have minimum distance, So $T(n)=T(n/2)+O(n)$, $T(n)=O(n\log n)$.