## 1.

### Solution:

Sort the set A, B using merge sort. Then for each a ∈ A separately use binary search to check if integer x – a exists in B. Sorting takes time Θ(nlogn). Binary search takes time O(logn) and is executed n times. Thus the total time is Θ(nlogn).
If A, B are sorted, the problem can also be solved in linear time by scanning the list A, B at the same time forward and backward directions:

### Pseudocode:

Input: list A,B sorted in ascending order, a,b
Output: true if there exist two elements (a in A and b in B) whose sum is exactly x, otherwise false.

```
i ← 1, j ← n
while i ≤ j do
    if A[i] +B[ j] = x then
        return true
    if A[i] +B[ j] < x then
        i ← i+1
    else
        j ← j −1
return false
```

### 2. Solution:

(a): a=2,b=2,$\log_b a$=1, Here f(n)=$(n \log n)^4$,k=4,p=4, a<$b^k$ ,so T(n) = Θ($n^4(\log n)^4$).

(b): a=2,b=2,$\log_b a$=1, Here f(n)=$\log^2 n$,k=0,p=2. a>$b^k$, so T(n) = Θ($n^{\log_b a}$)= Θ(n).

(c): a=9,b=3,$\log_b a$=2, Here f(n)=$n^2$,k=2,p=0. a=$b^k$, so T(n) = Θ($n^{\log_b a} \log^{p+1} n$)
= Θ($n^2 \log n$).

(d): a=9,b=3,$\log_b a$=2, Here f(n)=$n^3$,k=3,p=0. a<$b^k$, so T(n) = Θ($n^k \log^p n$) = Θ($n^3$).

(e): a=7,b=3,$\log_b a=\log_3 7$, Here f(n)=$n^2$,k=2,p=0. a<$b^k$, so T(n) = Θ($n^k \log^p n$) = Θ($n^2$).

### 3. Solution:

The pivot element will always be the smallest element in each partition. Therefore, each partition will produce two subarrays.The first subarray contains only one element( which is the smallest element) and this element is in its right position. And the second subarray contains all the remaining elements. In this problem , the running time recurrence will be T(n)=T(n-1)+n. Thus,T(n)= Θ($n^2$).

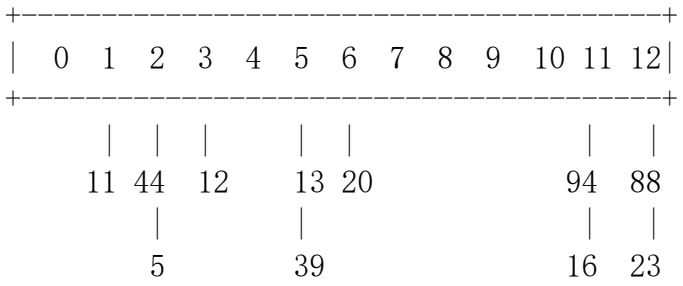
### 4. Solution:

Assume that n is odd and k is even. Sort the set S using merge sort. Sorting takes time Θ(nlogn). If the set S was sorted, then the median is in position n/2 and the k numbers in S which is closest to the median are in positions (n − k)/2 through (n + k)/2. First we use linear time selection to find the (n − k)/2, n/2, and (n + k)/2 elements and then go

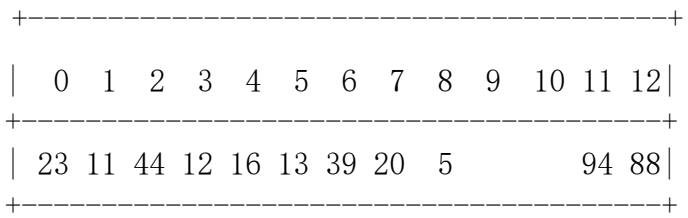
through the set S to find the numbers less than (n+k)/2 element, greater than the (n−k)/2 element, and not equal to the n/2 element. This algorithm takes O(n) time as we use linear time selection exactly three times and traverse the n numbers in S once.

## 5. Solution:

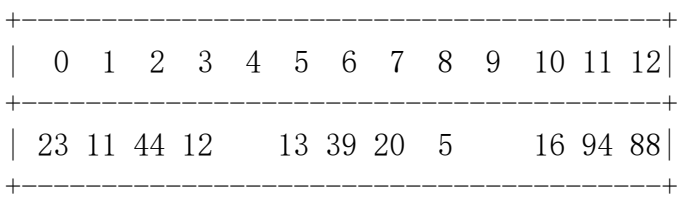
(1):




(2):




(3):




(4):