

## Discussion 10: Sorting

Caijie Zhang

<http://www.cs.ucsb.edu/~caijie>

*Mar. 10 2005*

7.33 Prove that any algorithm that finds an element  $X$  in a sorted list of  $N$  elements requires  $\Omega(\log N)$  comparisons.

Prove:

7.48 We are given an array that contains  $N$  numbers. We want to determine if there are two numbers whose sum equals a given number  $K$ . For example, if the input is 8, 4, 1, 6 and  $K$  is 10, then the answer is yes (4 and 6). A number may be used twice. Do the following:

- Give an  $O(N^2)$  algorithm to solve this problem.
- Give an  $O(N \log N)$  algorithm to solve this problem. (Hint: Sort the items first. After that is done, you can solve the problem in linear time.)

Solution:

a.

```
bool func(int a[], int n, int K)
{
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++){
            if (a[i]+a[j] == K)
                return true;
        }
    return false;
}
```

Time Complexitiy:  $O(n^2)$

b.

```
bool func(int a[], int n, int K)
{
    sort a[n] using Heapsort;                // O(nlog(n))

    int x = 0;
    int y = n-1
    while( x <= y ){                          // O(n)
        if( a[x] + a[y] < K ){
            x++;
        } else if( a[x] + a[y] > K ){
            y--;
        } else{
            return true;
        }
    }
    return false;
}
```

Time Complexitiy:  $O(n \log n)$