Sorting and Searching

selection sort

- Find the smallest element and exchange it with a[0],then find the smallest element in subarray a[1]...a[n-1], and swap it with a[1].
- **2， 1， 5， 4， 9**

  1， 2， 5， 4， 9　first pass

  1， 2， 5， 4， 9　second pass

  1， 2， 4， 5， 9　third pass

  1， 2， 4， 5， 9　forth pass
- implementing code

```
 int v = {1,2,4,5,9};
for(int i=0; i<length-1; i++){
        int min = v[i];
        int temp;
        int index = i;
        for(int j=i+1;j<v.length;j++){
            if(v[j] < min){
                min = v[j];
                index = j;
            }
        }

        temp = v[i];
        v[i] = min;
        v[index]= temp;
}
```

  exe:P333 8題

- Note:
  - for an array of n elements,the array is sorted after n-1 pass.
  - after the kth pass , the first k are in their final sorted position.

Insertion Sort

- definition
- for an array of n elements,the array is sorted after n-1pass.
- The worst case for insertion ----stored in reverse order
- the best case----already sorted in increasing order.

Recursive sorts: Mergesort and Quicksort

- inefficient:when selection and insertion sorts for large n,requiring approximately n pass.
- efficient: **divide-and-conquer** approach

Mergesort

how to do?

1. break the array into two halves
2. Mergesort the left half
3. Mergesort the right half
4. merge the two subarrays into a sorted array.

   Note

   - the major *disadvantage* is it needs a temporary array, this could be a problem if space is a factor.
   - Mergesort is not affected by the initial ordering of elements, best and worst and average cases have similar run times.

Quicksort

how to do ?

1. choose a pivot(random from the array)
2. placed all item to the left of pivot which are less or equal to the pivot.
3. whereas those to the right are greater than or equal to it.

Note

- for the *fastest run time*,the array should be partitioned into two parts of roughly the **same size**.
- if pivot happens to be the **smallest or largest** element in the array,one of the subarray is empty!!!!

  then quicksort becomes a version of selection sort
- so u can shuffle up the given array. or examining several elements of the array and taken median.

Sequential Search

start at the first element and compares the key to the each element in turn until the key is found or not found

1. the best case is the key in the first slot
2. the worst case is the key in the last case.
3. on average, there will be n/2 comparisons

binary search

if the elements are in a **sorted** array.

Note

- the best case,the key is found in the first try.
- the worst case, the key is not in the list or is at either end of a sublist.
- 2^n
- exe:9 10 11 12 13 28 30