Sorting

The Basic Problem

- Given an array

  x[0], x[1], ... , x[size-1]

  reorder entries so that

  x[0] <= x[1] <= . . . <= x[size-1]

  - List is in non-decreasing order.

- We can also sort a list of elements in non-increasing order.

Example

- Original list:

  10, 30, 20, 80, 70, 10, 60, 40, 70

- Sorted in non-decreasing order:

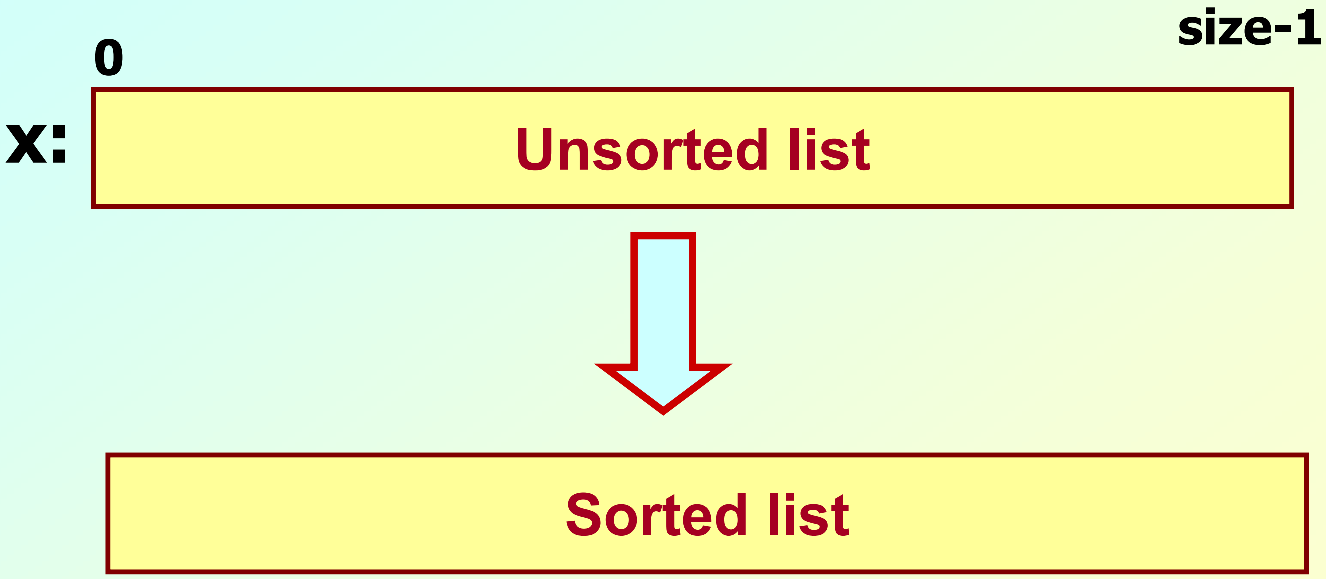
  10, 10, 20, 30, 40, 60, 70, 70, 80

- Sorted in non-increasing order:

  80, 70, 70, 60, 40, 30, 20, 10, 10

Sorting Problem

- What we want ?
  - Data sorted in order



size-1

0

x:

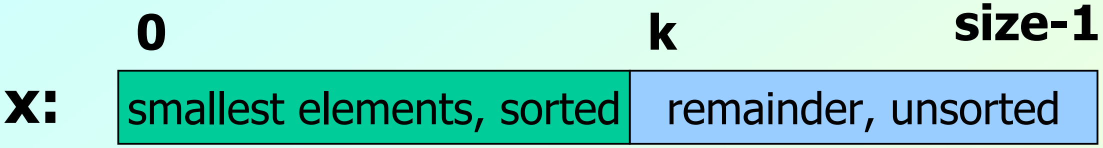
Unsorted list

Sorted list

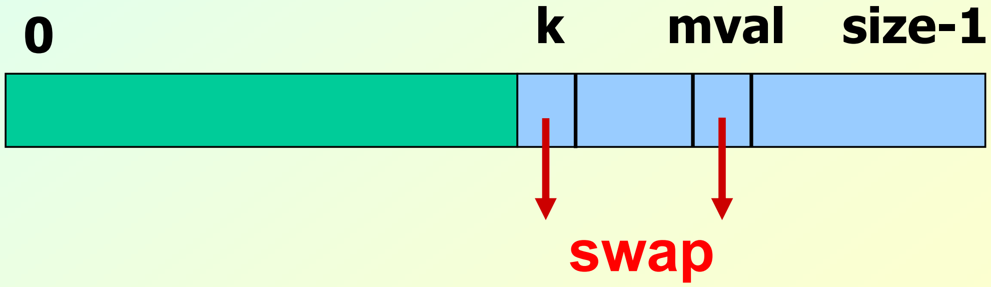
Selection Sort

How it works?

- **General situation :**



- **Step :**
  - **Find smallest element, mval, in x[k..size-1]**
  - **Swap smallest element with x[k], then increase k.**



Subproblem

```
/* Yield index of smallest element in x[k..size-1];*/

int min_loc (int x[], int k, int size)
{
    int j, pos;

    pos = k;
    for (j=k+1; j<size; j++)
        if (x[j] < x[pos])
            pos = j;
    return pos;
}
```

The main sorting function

```
/* Sort x[0..size-1] in non-decreasing order */

int selsort (int x[], int size)
{   int k, m;

    for (k=0; k<size-1; k++)
    {
       m = min_loc (x, k, size);
       temp = a[k];
       a[k] = a[m];
       a[m] = temp;
   }
}
```

```
int main()
{
  int x[ ]={-45,89,-65,87,0,3,-23,19,56,21,76,-50};
  int i;
  for(i=0;i<12;i++)
      printf("%d ",x[i]);
  printf("\n");
  sel_sort(x,12);
  for(i=0;i<12;i++)
      printf("%d ",x[i]);
  printf("\n");
}
```

```
-45 89 -65 87 0 3 -23 19 56 21 76 -50
-65 -50 -45 -23 0 3 19 21 56 76 87 89
```

Example

x:	3	12	-5	6	142	21	-17	45

x:	-17	12	-5	6	142	21	3	45

x:	-17	-5	12	6	142	21	3	45

x:	-17	-5	3	6	142	21	12	45

x:	-17	-5	3	6	142	21	12	45

x:	-17	-5	3	6	12	21	142	45

x:	-17	-5	3	6	12	21	142	45

x:	-17	-5	3	6	12	21	45	142

Analysis

- How many steps are needed to sort n things ?
  - Total number of steps proportional to n^2
  - No. of comparisons?

$$(n-1)+(n-2)+\ldots\ldots+1= n(n-1)/2$$

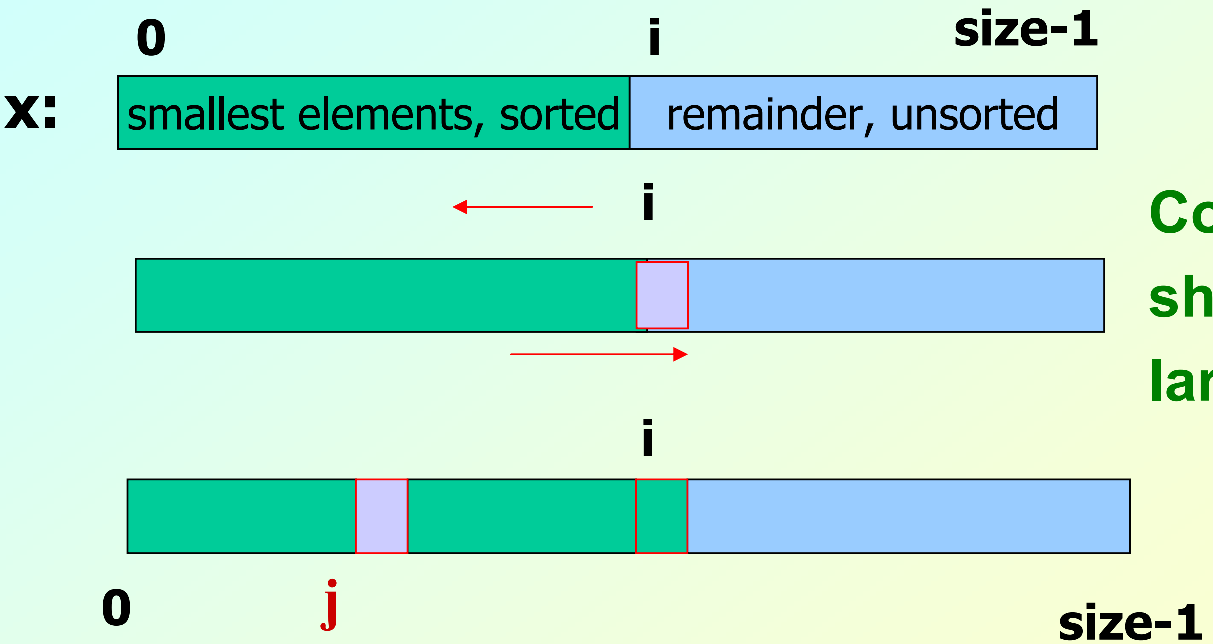
Of the order of n^2

  - Worst Case? Best Case? Average Case?

Insertion Sort

How it works?

- **General situation :**



Compare and shift till x[i] is larger.

Insertion Sort

```
void InsertSort (int list[], int size)
{
  int i,j,item;

  for (i=1; i<size; i++)
     {
      item = list[i] ;
      for (j=i-1; (j>=0)&& (list[j] > i); j--)
            list[j+1] = list[j];
      list[j+1] = item ;
  }
}
```

Time Complexity

- Number of comparisons and shifting:
  - Worst case?

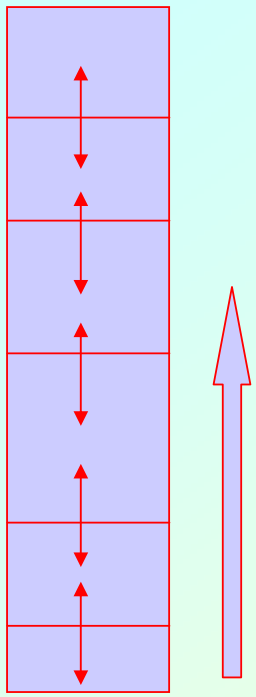
    1 + 2 + 3 + …… + (n-1) = n(n-1)/2
  - Best case?

    1 + 1 + …… + (n-1) = (n-1)

Bubble Sort

How it works?

Bubble Sort



In every iteration heaviest element drops at the bottom.

The bottom moves upward.

Bubble Sort

```
void swap(int *x, int *y)
{
  int tmp = *x;
  *x = *y;
  *y = tmp;
}
```

```
void bubble_sort
         (int x[], int n)
{
  int i,j;

  for (i=n-1; i>0; i--)
    for (j=0; j<i; j++)
      if (x[j] > x[j+1])
        swap(&x[j],&x[j+1]);
}
```

```
int main()
{
  int x[ ]={-45,89,-65,87,0,3,-23,19,56,21,76,-50};
  int i;
  for(i=0;i<12;i++)
      printf("%d ",x[i]);
  printf("\n");
  bubble_sort(x,12);
  for(i=0;i<12;i++)
      printf("%d ",x[i]);
  printf("\n");
}
```

```
-45 89 -65 87 0 3 -23 19 56 21 76 -50
-65 -50 -45 -23 0 3 19 21 56 76 87 89
```

Time Complexity

- Number of comparisons :
  - Worst case?

    $1 + 2 + 3 + \ldots\ldots + (n-1) = n(n-1)/2$

  - Best case?

    Same

- How do you make best case with (n-1) comparisons only?
  - By maintaining a variable `flag`, to check if there has been any swaps in a given pass.
  - If not, the array is already sorted.