Searching and Sorting

NOTES

Linear Search (Code)

```
// To search for key in arr using linear search
// Return index if found; otherwise return -1


int linearSearch(int arr[], int size, int key)
{
      int i;
      for (i=0; i<size; i++)
            if (key == arr[i])
                  return i;
      return -1;  // not found
}
```

Binary Search (Algorithm)

(Pre-condition: list is sorted in ascending order)

- Look for the key in the middle position of the list.
- Either of the following 2 cases happens:
  - If the key is **smaller** than the middle element, then "discard" the right half of the list and repeat the process.
  - If the key is **greater** than the middle element, then "discard" the left half of the list and repeat the process.
- Terminating condition: either the key is found, or when all elements have been "discarded".

Binary Search (Code)

```
// To search for key in sorted arr using binary search
// Return index if found; otherwise return -1
int binarySearch(int arr[], int size, int key)
{
    int low=0, high=size-1, mid=(low + high)/2;
    while ((low <= high) && (arr[mid] != key))
    {
        if (key < arr[mid])
            high = mid - 1;
        else
            low = mid + 1;
        mid = (low + high)/2;
    }
    if (arr[mid] == key)
        return mid;
    else
        return -1;
}
```

Selection Sort (Algorithm)

- **Step 1:**

  Find the smallest element in the list.

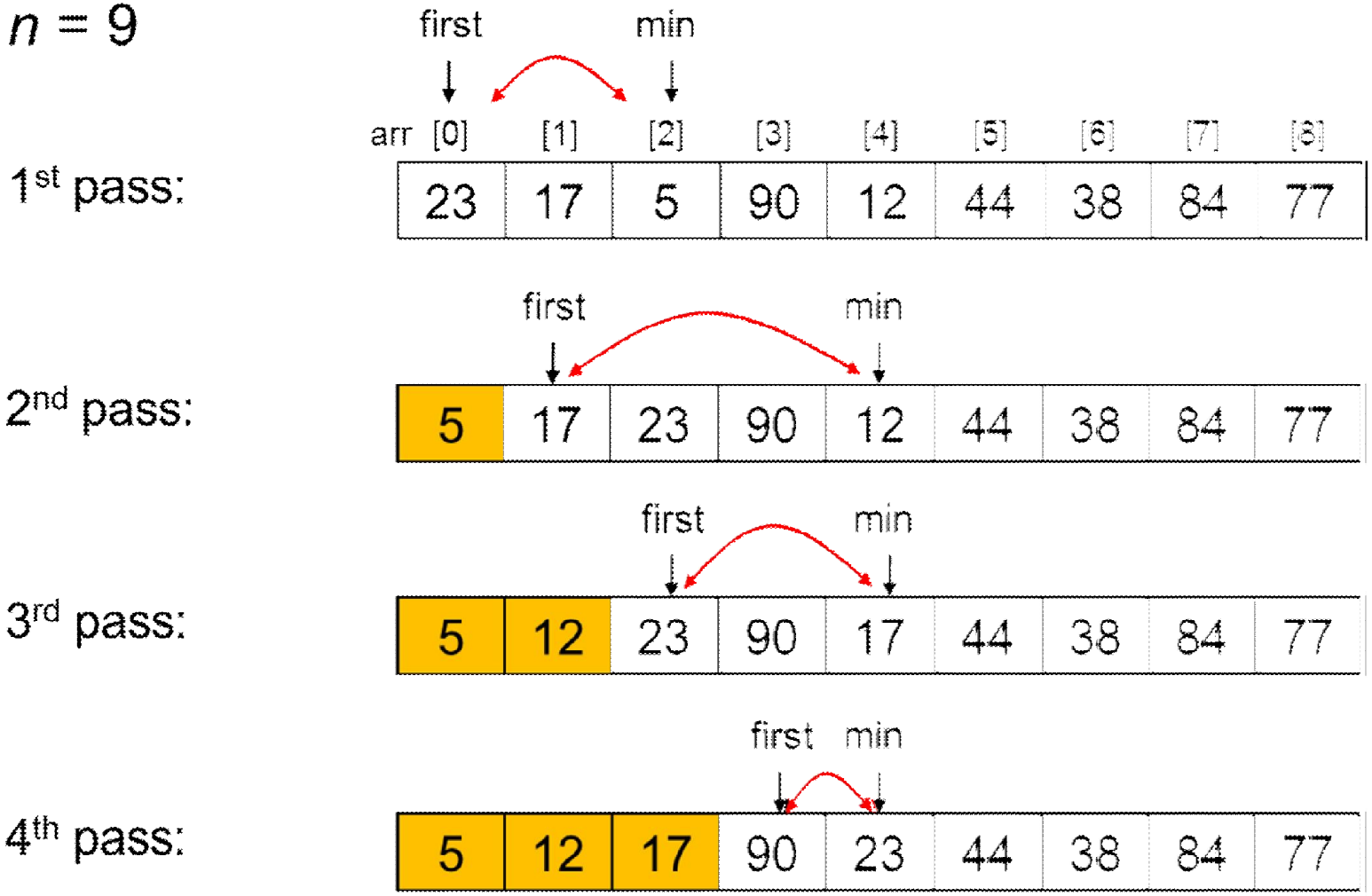
- **Step 2:**

  Swap this smallest element with the element in the first position. (Now, the smallest element is in the right place.)

- **Step 3:**

  Repeat steps 1 and 2 with the list having one fewer element (i.e. the smallest element just found and placed is "exempted" from further processing).

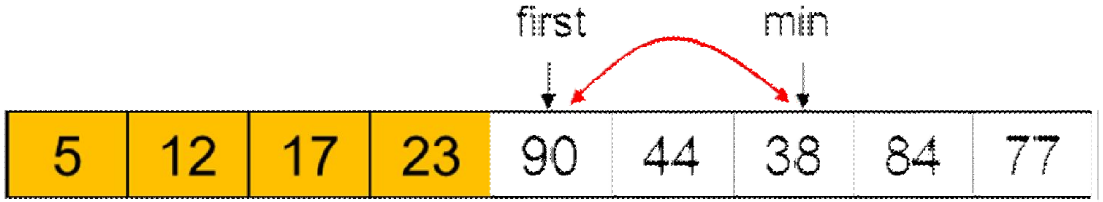
Selection Sort (Part 1)



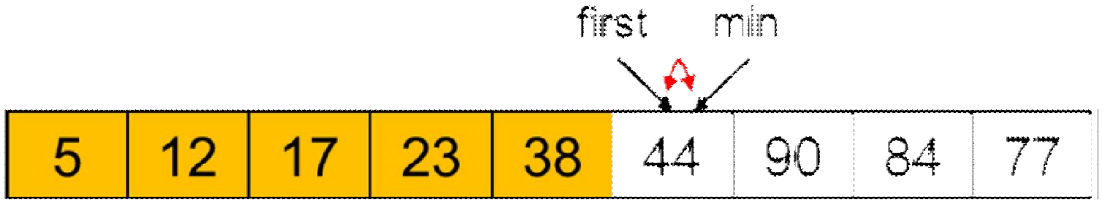
Selection Sort (Part 2)

$n = 9$

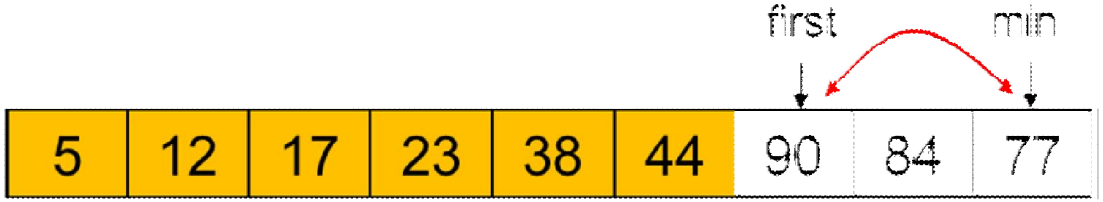
5th pass:



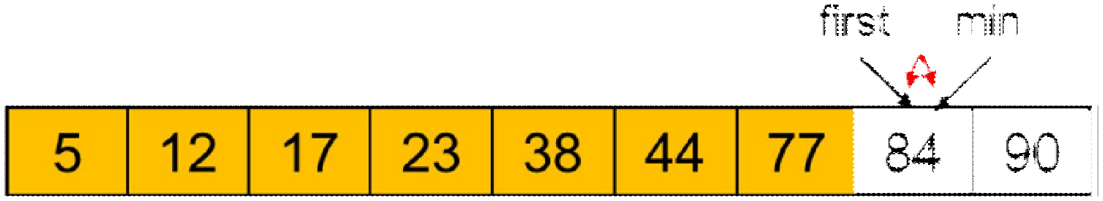
6th pass:



7th pass:



8th pass:



Final array:

Selection Sort (Code)

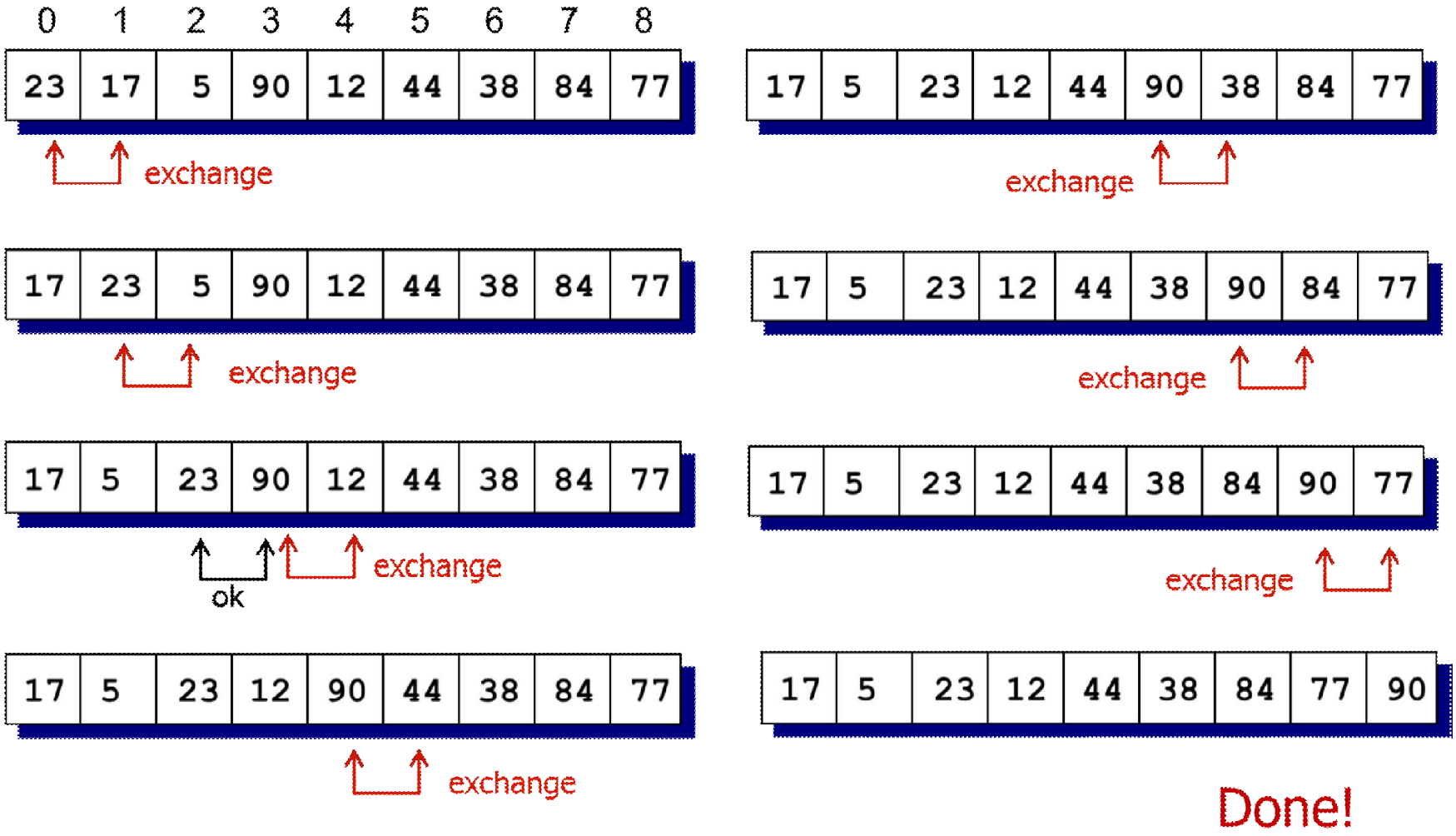
```
// To sort arr in increasing order
void selectionSort(int arr[], int size)
{
     int i, start_index, min_index, temp;

     for (start_index = 0; start_index < size-1; start_index++)
     {
          // each iteration of the for loop is one pass

          // find the index of minimum element
          min_index = start_index;
          for (i = start_index+1; i < size; i++)
               if (arr[i] < arr[min_index])
                    min_index = i;

          // swap minimum element with element at start_index
          temp = arr[start_index];
          arr[start_index] = arr[min_index];
          arr[min_index] = temp;
     }
}
```

Bubble Sort (Algorithm)



Bubble Sort

```
// To sort arr in increasing order
void bubbleSort(int arr[], int size)
{
     int i, limit, temp;

     for (limit = size-2; limit >= 0; limit--)
     {
          // limit is where the inner loop variable i should end

          for (i=0; i<=limit; i++) // one pass
          {
               if (arr[i] > arr[i+1])  // swap arr[i] with arr[i+1]
               {
                  temp = arr[i];
                  arr[i] = arr[i+1];
                  arr[i+1] = temp;
               }
          }
     }
}
```

```
// To sort arr in increasing order
void bubbleSort(int arr[], int size)
{
     int i, limit, temp;
     int done = 0;

     for (limit = size-2; limit >= 0 && !done; limit--)
     {
          // limit is where the inner loop variable i should end
          done = 1;

          for (i=0; i<=limit; i++) // one pass
          {
               if (arr[i] > arr[i+1])  // swap arr[i] with arr[i+1]
               {
                  temp = arr[i];
                  arr[i] = arr[i+1];
                  arr[i+1] = temp;
                  done = 0;
               }
          }
     }
}
```