كلية علوم الحاسوب وتكنولوجيا المعلومات

المرحلة الثانية

مادة الخورازميات

م.م فرح معاذ جاسم

4.1 Sorting Algorithm

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order.

Other Definition: **Sorting** is the process of arranging a set of graphical elements according to the value of a field (or fields) called a key (ascending) or descending.

4.2 Sorting Applications and Purposes

- Uniqueness testing
- Deleting duplicates: solve the problem of similarity restrictions.
- Prioritizing events :To simplify the processing of files
- Frequency counting
- Reconstructing the original order
- Set intersection/union
- Finding a target pair x, y such that x+y = z
- Efficient searching: To increase the efficiency of the search algorithm for an item.

3.3 Steps in the sorting process

The steps of the sorting algorithm are summarized in the following stages:

1- Reading the key field. which mean the input data that will sort it.

2- Inference (deduction) the location of the element in the new arrangement.

3- Move the sorted element to its new location.

4.4 Types of sorting algorithms

There are two types of internal sort (in the main memory) ,and external sort (the secondary storage).

A –Internal sort

The order that occurs in the main memory of the computer when the volume of data is appropriate (not large) for the storage of the memory.

Among the most important types:

- Selection Sort
- Bubble Sort (exchange)
- Insertion Sort
- Sort Quick Sort
- Radix Sort
- Heap Sort
- Shell Sort

B. External Sort

It is the arrangement of the data stored in the secondary storage media in the form of files when the volume of data is very large so that it cannot be absorbed all in memory at one time during the arrangement process and one of its most important types

- Tow-way-Merge Sort
- K-way-Merge Sort
- Balanced Two-way-Merge sort
- polyhase Tow -way-Merge sort

4.5 key determinants of the sorting algorithm selection

The testing of any of the ranking algorithms should be in light of a number of factors, the most important of which are:

1 - The volume of data stored.

2. Storage type (main memory, disk, and tape).

3. Degree of data order (unordered, semi-ordered)

4.6 Bubble Sort

The idea of this method involves testing the smallest values and placing them in the list (that is, the small value floats to the surface.)

1. In the first stage (first pass) :We compare the two elements in the two locations (n-1), (n) and we exchange their location to be the smallest before the other, and we continue to the top of the list until we reach the comparison of the element in the second location with the element in the first site.
2. In the second pass: We compare in the same way as the previous one, but from the element in the location (n) to the element in the second site because the first site was chosen where the least valuable element in the previous step
3. Mention the above steps for (n-1) stages.

Example: sort the list by 8, 3, 9, 7, 2 ascending:

The fourth pass	The third pass		The second pass			The first pass				The input
1	2	1	3	2	1	4	3	2	1	
2	2	2	2	2	2	**2**	8	8	8	8
3	3	3	**3**	**8**	8	**8**	**2**	3	3	3
7	7	**8**	8	**3**	3	3	**3**	**2**	9	9
8	8	**7**	7	**7**	**7**	9	9	**9**	**2**	7
9	9	9	9	9	**9**	7	7	7	**7**	2

Note1: that the number of elements in the list is (n = 5) and the number of stages (n-1 = 4) The number of steps in each stage decreases by one by the number of steps in the previous stage.

<u>Notes2:</u>

-Average no. of comparison is (n2 / 2) where (n) represents the number of menu items.

-The average no. of exchanges is ($n^{4/2}$).

The method is good when the elements are semi-arranged, the number is not large and does not need large and simple storage space.

- The execution time is O (n^2)

Bubble Worst-case

- Input is in descending order
- Running time remains the same: O(n^2)

Bubble Best-case

- Input is already in ascending order
- The algorithm returns after a single outer iteration
- Running time: O(n)

Bubble Sort function

```
 void bubble sort(int ar[n])
 {
  int i,j;
  int x;
  for(i=0;i<n;i++)
 {
 for(j=n-1;j>i;--j)
    {
  if(ar[j]<ar[j-1])
    { x=ar[j];
    ar[j]=ar[j-1];
     ar[j-1]=x;
     }
     }
}
}
```

4.7 Selection sort

The algorithm for this arrangement is summarized by the following steps:

1- Find the smallest item in the list and replace it from its location with the item in the first location in the list.
2- Find the smallest element in the remaining part of the list and replace it from its location with the element in the second location in the list.
3- We continue in this process until we reach the end of the list.

Example: sort the following list in ascending order (8 3 9 7 2 6 4).

6	5	4	3	2	1	القائمة الأصلية
2	2	2	2	2	2	8
3	3	3	3	3	3	3
4	4	4	4	9	9	9
6	6	6	7	7	7	7
7	7	8	8	8	8	2
8	8	7	6	6	6	6
9	9	9	9	4	4	4

Number of menu items n = 7
number of stages no. Passes n-1 = 6
Notes: - The average number of comparisons comparisons is n / 2 * (n-1)
Average number of substitutions is not average.
Exchanges average. is (n-1)

Selection sort function

```
void slctsort(int data[n],int s)
{
 int i,k,j,item,x,y;
 for(i=0;i<s-1;i++)
  {
   k=i;
   item=data[i];
   for(j=i+1;j<s;j++)
    {
     if(data[j]<item)
      {
       x=data[j];
       data[j]=item;
       item=x;
      }
    }
   y=item;
   item=data[k];
   data[k]=y;
  }
}
```