Selection Sort Algorithm

This type of sorting is called "Selection Sort" because it works by repeatedly element. It works as follows: first find the smallest in the array and exchange it with the element in the first position, then find the second smallest element and exchange it with the element in the second position, and continue in this way until the entire array is sorted.

Algorithm: Selection Sort -

```
SELECTION_SORT (A)

for i ← 1 to n-1 do
min j ← i;
min x ← A[i]
for j ← i + 1 to n do
if A[j] < min x then
min j ← j
min x ← A[j]
A[min j] ← A [i]
A[i] ← min x
```

Selection sort is among the simplest of sorting techniques and it work very well for small files. Furthermore, despite its evident "naïve approach "Selection sort has a quite important application because each item is actually moved at most once, Section sort is a method of choice for sorting files with very large objects (records) and small keys.

The **worst case** occurs if the array is already sorted in descending order. Nonetheless, the time require by selection sort algorithm is not very sensitive to the original order of the array to be sorted: the test "if A[j] < min x" is executed exactly the same number of times in every case. The variation in time is only due to the number of times the "then" part (i.e., min j ← j; min x ← A[j] of this test are executed.

The Selection sort spends most of its time trying to find the minimum element in the "unsorted" part of the array. It clearly shows the similarity between Selection sort and Bubble sort. Bubble sort "selects" the maximum remaining elements at each stage, but wastes some effort imparting some order to "unsorted" part of the array. Selection sort is**quadratic** in both the worst and the average case, and requires no **extra memory**.

For each i from 1 to n - 1, there is one exchange and n - i comparisons, so there is a total of n -1 exchanges and (n -1) + (n -2) + . . . + 2 + 1 = n(n -1)/2 comparisons. These observations hold no matter what the input data is. In the worst case, this could be quadratic, but in the average case, this

quantity is O(n log n). It implies that the running time of Selection sort is quite insensitive to the input.

Implementation

```
void selectionSort(int numbers[], int array_size)
{
int i, j;
int min, temp;
for (i = 0; i < array_size-1; i++)
{
min = i;
for (j = i+1; j < array_size; j++)
{
if (numbers[j] < numbers[min])
min = j;
}
temp = numbers[i];
numbers[i] = numbers[min];
numbers[min] = temp;
}
}
```

Source:

http://www.learnalgorithms.in/#