Selection Sort (Selection)

Selection Sort

Selection sort works by *selecting* the smallest remaining element of the input array and *appending* it at the end of all the elements that have been inserted so far.

Just like Insertions sort, it does this by partitioning the array into a sorted part at the front and an unsorted part at the end.

Initially the sorted part is empty and the unsorted part is the whole input array.

In each pass it finds the smallest element of the unsorted part and swaps it with the first element of the unsorted part of the array. Then it moves the split position between the sorted and the unsorted parts of the array on by one cell.

Example of a Selection Sort run

1. $\big|\, 5, 12, 6, \underline{3}, 11, 8, 4$
2. $3 \,\big|\, 12, 6, 5, 11, 8, \underline{4}$
3. $3, 4 \,\big|\, 6, \underline{5}, 11, 8, 12$
4. $3, 4, 5 \,\big|\, \underline{6}, 11, 8, 12$
5. $3, 4, 5, 6 \,\big|\, 11, \underline{8}, 12$
6. $3, 4, 5, 6, 8 \,\big|\, \underline{11}, 12$
7. $3, 4, 5, 6, 8, 11 \,\big|\, \underline{12}$
8. $3, 4, 5, 6, 8, 11, 12 \,\big|$

Selection Sort (pseudocode)

```
selectionsort(a, n){
  for ( i = 0 ; i < n-1 ; i++ ) {
      k = i
      for ( j = i+1 ; j < n ; j++ )
          if ( a[j] < a[k] )
              k = j
      swap a[i] and a[k]
  }
}
```

Selection Sort Complexity

The outer loop is iterated $n-1$ times

In the worst case, the inner loop is iterated $n-1$ times for the first outer loop iteration, $n-2$ times for the 2nd outer iteration, etc. Note that this is for best, worst and average cases. Thus, the total number of comparisons is:

$$\begin{aligned}\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 &= \sum_{i=0}^{n-2}(n-1-i) \\ &= (n-1) + \cdots + 2 + 1 \\ &= \frac{n(n-1)}{2}.\end{aligned}$$

Hence best, average and worst case complexity is $O(n^2)$

Selection Sort Stability

Consider what happens when two elements with the same value are in the array to be sorted.

For example, consider when the input array contains $2_1, 2_2, 1$, where the subscript indicates the order of appearance of the two copies of the value 2.

In the first pass, we find the smallest element, in this case the 1, and swap it with the first element in the array, the 2_1. This results in $1, 2_2, 2_1$. In other words, the 2 copies of 2 have changed order.

No matter how we change the condition for which element of a set of elements of the same (smallest) value we then select, we can easily produce counterexamples that show that the order of elements with the same value can change.

Hence selection sort is *unstable*.