Algorithms & Data Structures 2018/19 Coursework

clvp22

December 26, 2018

Question 6.

(b)

The worst case input for an input size of 16 is: $1, 5, 9, 13, 3, 7, 11, 15, 2, 6, 10, 14, 4, 8, 12, 16$

Our *msort* recursively splices the list into 4 sublists,

$$1, 5, 9, 13|3, 7, 11, 15|2, 6, 10, 14|4, 8, 12, 16$$

Selection sort is not an adaptive sorting algorithm (every input of size n takes the same amount of time). So the sorted list 1,5,9,13 would take the same time as any other permutation of 1,5,9,13, using selection sort. Therefore it doesn't matter if our sublists are sorted or not we always take $O(n^2)$, or more specifically $n(n-1)/2$ comparisons.

Merging 2 lists of size n takes atleast n and at most $2n-1$ comparisons. It takes $2n-1$ comparisons if the left and right lists store alternating elements - an element is added to the new list from one list and then the next element is added from the other list and so on until one of the lists in empty. Clearly the 2 lists, 1,5,9,13 and 3,7,11,15 store alternating elements so $2n-1$ comparions will be needed to merge these 2 lists. Again, the 2 lists 2,6,10,14 and 4,8,12,16 store alternating elemenst so $2n-1$ comparisons will be needed to merge these 2 lists.

After we unwind one level of recursion we get the 2 sublists,

$$1, 3, 5, 7, 9, 11, 13, 15|2, 4, 6, 8, 10, 12, 1, 4, 16$$

These 2 sublists store alternating elements so $2n-1$ comparisons are needed again to merge these 2 lists. Because we always require the maximum amount of comparisons $(2n-1)$ when merging 2 lists together this is a worst case input for an input size of 16. We need 53 comparisons $= 4(4(4-1)/2) + 2(2(4)-1) + 2(8) - 1$