CLRS, 2-2 Corectness of bubblesort

Jonas Brunsgaard

May 4, 2012

a. A' has to be a permutation of A.

b. **Loop invariant (2-4):** In the subarray $A[j...n]$, j are the minimum element. Also $A'[j...n]$ is a permutation of $A[j...n]$.

   i. **Initialization:** At this point $j = n$ and therefore $A[j...n] = A[j]$, which satisfy the loop invariant.

   ii. **Maintenance:** For each iteration we see that if $A[j] < A[j-1]$, then $A[j]$ and $A[j-1]$ are swapped, thus $A[j-1]$ will be a minimum element for the next iteration, where it will be referred to as $A[j]$. The loop invariant then holds for maintenance.

   iii. **Termination:** The loop will terminate when $j = i$, at this point $A[i]$ will be the minimum element in $A[i...n]$.

c. **Loop invariant (1-4):** The subarray $A[1...i-1]$ appended with $A[i...n]$ are a permutation of the original A. The subarray array $A[1...i-1]$ will be sorted, which follows from *b.*. Also, all the elements in $A[1...i-1]$ are smaller or equal to the elements in $A[i...n]$.

   i. **Initialization:** At this point $i = 1$ and thus $A[1...i-1] = A[1...0]$, which are an empty list. The substring $A[i...n]$ are, at this point, identical to the original A, so it is clearly a permutation of A. At this point the loop invariant holds and we are all good.

   ii. **Maintenance:** From the termination in sub-assignment *b.* we know, that i are the minimum element in $A[i...n]$ when the loop terminates, and because it increase by 1 for each iteration, we know that $A[1...i-1]$ are always in sorted order. Also in each iteration we move one element from the unsorted array to the sorted array, thus we maintain the requirement, that the two arrays appended are a permutation of A.

   iii. **Termination:** The loop will terminate at the point where $i = A.length = n$, and thus the whole list $A[1...n]$ is sorted.

d. To find the worstcase running time of bubblesort we analyse each line of the algorithm.

```
BUBBLESORT($A$)                                              cost      times
1  for $i = 1$ to $A.length - 1$                             $c_1$     $n$
2      for $j = A.length$ downto $i + 1$                     $c_2$     $\sum_{j=1}^{n-1} t_j$
3          if $A[j] < A[j-1]$                                $c_3$     $\sum_{j=1}^{n-1} (t_j - 1)$
4              exchange $A[j]$ with $A[j-1]$                 $c_4$     $\sum_{j=1}^{n-1} (t_j - 1)$
```

If we sum up each line, this can be written as

$$T(n) = c_1 n + c_2 \sum_{j=1}^{n-1} t_j + c_3 \sum_{j=1}^{n-1} (t_j - 1) + c_4 \sum_{j=1}^{n-1} (t_j - 1)$$

If we use a little math, and write up the worst case scenario, the following appear

$$T(n) = c_1 n + c_2 \frac{1}{2}(n^2 - n - 2) + c_3 \frac{1}{2}(n^2 - 3n + 2) + c_4 \frac{1}{2}(n^2 - 3n + 2)$$

We not really interested in c_i for each line(it does not have any effect on the growth), so we assume $c_1 = c_2 = c_3 = c_4 = 1$ and reduce the expression further

$$T(n) = \frac{3}{2} n^2 - \frac{5}{2} n + 1$$

Now we clearly see that bubblesort is a quadratic function of n. Thus, it has the same runtime as insertion sort.