Exercise II 04

Sunday, May 5, 2024 5:01 PM

Problem 1 to hand in: *Greedy Proof Attempt: Find the Error*

Consider *Selection Sort*, a greedy sorting algorithm that runs in quadratic time in the input size (even if the input is already sorted).

`selection_sort`$(A[0..n-1], n)$:

```
1 for j ← 0 to n − 1 do
2     find index min of the minimum element in A[j..n − 1]
3     swap A[j] and A[min]
4 return A
```

Take a look at the following proof attempt that claims to show why the algorithm works. Determine and explain three crucial errors. Repair the errors.

> **Loop invariant:** In loop iteration j, the current element $A[j]$ is swapped with the minimum of the remaining subarray $A[j..n-1]$.
>
> **Proof by induction over j:**
>
> **Base case:** For $j = 0$, the minimum index of the whole array $A[0..n-1]$ is found (line 2). This minimum swaps positions with $A[0]$ (line 3).
>
> **Induction step:** In later iterations, the minimum is only found in the remaining subarray $A[j..n-1]$ (line 2). This is sufficient, because the beginning of the array is already sorted, and therefore saves running time. The minimum is swapped with $A[j]$ (line 3) to save the minimum at the current position j.
>
> All in all, by this loop invariant we obtain the following **termination case:** After n steps, the array is completely sorted.

1. Loop invariant shall address the current state at each iteration j rather than stating the steps in the current iteration.
   Correction: In loop iteration j, the array A[0,…,j-1] has the smallest j elements in A and it is correctly sorted.
2. Base case then needs to be tweaked according to the loop invariant:
   Correction: In loop iteration 0, the array A is unsorted. It has the smallest 0 element in A and therefore is already correctly sorted.
3. In the induction step, we need to prove that the swapping operation we did guarantees the local best choice, leading to a global optimum.
   Correction: In loop iteration j, A[0,…,j-1] fulfills the loop invariant, then the elements are correctly sorted.
   Assume A[j] is not chosen correctly in iteration j. Then the j-th element of the sorted array is not our chosen A[j], which means there exists a smaller element A[min'] in A[j,…,n-1] than swapped A[j]. This leads to A[min']<A[min] which contradicts the minimum search operation in line 2. The assumption is then wrong, A[j] equals the j-th element of the sorted array.

   Then in all, each entry of A after n steps corresponds to the entry of sorted array. A is sorted correctly.