Homework Assignment 1 (Design techniques)

Solution

Problem 1

(a) Yes. After \(i\) iterations of the WHILE-LOOP, the largest \(i\) items are in the correct last \(i\) positions of the array. This is because the algorithm at the first run eventually picks up the largest element of the array and carries it through to the last position. In the second run, the second-largest element is placed correctly, etc.

(b) No! If, for example, the array is in inverse order, it will take the last (smallest) element \(n\) steps to arrive at the (correct) first position.

(c) In the best case, the array is already in correct order and the algorithm just passes through once doing \(n - 1\) comparisons; however, in the worst case, as we see from (b), it will take \(n\) iterations of \(n - 1\) comparisons each. Hence, we end up with a worst-case running time of order \(O(n^2)\).

Problem 2

(a) First, we sort the array of the \(x_i\) (and thereby also the weights) following some sorting algorithm which takes \(O(n \log n)\) time (Wikipedia). Thereafter, we just start adding up weights \(w_i\) from the "left" of the sorted array (that is, with increasing \(x_i\)'s) until we reach or exceed the value 0.5. The index at which this happens gives the weighted median. This procedure of summing up can take up to \(n\) additions and thus uses time \(O(n)\). In total, the consumed time is

\[
T(n) = O(n \log n) + O(n) = O(n \log n).
\]

(b) With worst-case time \(O(n)\), this problem can be solved using a prune-and-search technique. The procedure is the following:

- Find the median of the \(x_i\) array (linear-time algorithm).

- Place the median at the correct position in the array, such that to its left all items are no larger, and to its right all items are no smaller than the median itself (w.r.t. the \(x_i\) values; SPLIT algorithm, linear time). This completes the "prune" part.
• The "search" part: Initially, set $\rho = 0$. The variable $\rho$ will represent the weight of the cut-off part on the left of the (sub-)array at hand. Add up all weights of items to the left of the median, call the result $\Sigma$.

- If $\Sigma > 0.5 - \rho$, start over the procedure with the left part of the array and leave $\rho$ unchanged.
- If $\Sigma < 0.5 - \rho$, but $\Sigma + \text{(weight of median)} \geq 0.5 - \rho$, the median equals the weighted median. Thus, finish the algorithm and return the index of the median.
- If $\Sigma + \text{(weight of median)} < 0.5 - \rho$, start over the procedure with the right part of the array and update $\rho$ by $\rho = \rho + \Sigma$.

Since each subroutine takes linear time, their succession within each iteration is linear time also. However, each iteration halves the size of the array that needs to be investigated. Therefore, the recursion for the running time reads

$$T(n) = O(n) + T\left(\frac{n}{2}\right).$$

We solve this by induction: Assume $T(m) \leq cm$ for $m = 1, \ldots, n - 1$ and write $O(n) \leq \frac{cn}{2}$ for some constant $c > 0$. The latter holds if we choose $c$ large enough, and we have to show that $T(n) \leq cn$, which follows directly by inserting:

$$T(n) \leq \frac{cn}{2} + c \cdot \frac{n}{2} = cn.$$