Algorithms and Data Structures (WS15/16)

Example Solutions for Unit 3

Problem 1

Algorithm: for the simplicity let's assume that $n = 2k$, algorithm can be easily modified if $n = 2k + 1$ by using one additional comparison . Let $A_1, A_2, ...A_n$ be the elements for which we want to find the maximum and the minimum. We also use two sets W and L, which are initially empty. First, for every $1 \leq i \leq k$ we compare A_{2i-1} and A_{2i}, if w.l.o.g we have that $A_{2i-1} < A_{2i}$, we add A_{2i-1} to L and we add a_{2i} to W. After k comparisons we will have that $|W| = |L| = k$. For the set W, while $|W| > 1$ we compare any $A_i, A_j \in W$ and remove the element which is smaller from W . For the set L, while $|L| > 1$ we compare any $A_i, A_j \in L$ and remove the larger element from L. In the end we will get that element in W is the maximal element and the element in L is the minimal element. In total we have used $k + (k - 1) + (k - 1) = 3n/2 - 2$ comparisons.

Lower Bound: We use an adversary argument to prove that any comparison-based algorithm for finding the maximum and the minimum must make at least $\lceil 3n/2 \rceil$ - 2 comparisons. For the simplicity we again assume that $n = 2k$. The adversary uses four sets : B, P_{max}, P_{min} and C. Also, partial order is kept for the comparisons already made. Initially $B = \{A_1, A_2, ..., A_n\}$, and P_{max}, P_{min} and C are empty. The adversary uses the following rules to answer the comparison question about two elements A_i and A_j :

1. if $A_i \in B$ and $A_j \in B$, the adversary answers that $A_i < A_j$. Then A_i is moved to P_{min} and A_j is moved to P_{max}.

2. if $A_i \in P_{max}$ and $A_j \in B \cup C$, the adversary answers that $A_i > A_j$. Then if $A_j \in B$, it is moved to P_{min}.

3.if $A_i \in P_{min}$ and $A_j \in B \cup C$, the adversary answers that $A_i < A_j$. Then if $A_j \in B$, it is moved to P_{max}.

4. if $A_i \in P_{min}$ and $A_j \in P_{min}$, the adversary answers that $A_i < A_j$. Then A_j is moved to C.

5. 4. if $A_i \in P_{max}$ and $A_j \in P_{max}$, the adversary answers that $A_i > A_j$. Then A_j is moved to C.

6. if $A_i \in C$ and $A_j \in C$ the adversary answers consistanly with the partial order. If A_i and A_j are incomparable according to the partial order, the adversary answers that $A_i < A_j$.

After every comparison partial order is updated if necessary. When the algorithm has finished, adversary tries to assign values to the elements in such way that they are consistent with the answered comparisons and the answer algorithm gave is incorrect. Observe that, if there is an element left in B, the adversary can pick it to be the maximal(minimal) element depending on the answer of algorithm. Also, if $|P_{max}| > 1$ the adversary can pick any of the elements contained in P_{max} to be the maximal element. Analogously, if $|P_{min}| > 1$ the adversary can pick any of the elements contained in P_{min} to be the minimal element. So, for the algorithm to give the correct answer , we must have that $|B| = 0$, $|P_{max}| = |P_{min}| = 1$ and $|C| = n - 2$. Elements are removed from B only by the rules 1, 2 and 3. Each time at most 2 elements are removed from B, so the algorithm must make at least k comparisons by rules 1,2 and 3. Elements are moved to C only by the rules 4 and 5, and only one element is moved to C, so

the algorithm must make at least $n-2$ comparisons by rules 4 and 5 . Comparisons made by rules 1,2,3 are different from the comparisons made by rules 4 and 5. Thus , the algorithm will need at least $k+n-2 = 3n/2-2$ comparisons.

Problem 2

to be added

Exercises for Unit 3

(3.) "$a+b \neq c$" - problem

input : sorted sets A, B, C of $\Theta(n)$ numbers

question : $a_i + b_j \neq c_k \quad \forall a_i \in A, b_j \in B, c_k \in C$?

a) this problem can be solved using $O(n^2)$ comparisons

naive way = consider each triple (a_i, b_j, c_k) and check whether $a_i + b_j = c_k$

$\rightsquigarrow$ number of triples is $n \cdot n \cdot n$

$\rightsquigarrow$ runtime $\Theta(n^3)$

smarter way = sorted sequences $\rightarrow$ exploit this fact

w.l.o.g. assume A, B, C contain exactly n elements

$A = \{a_1, \dots, a_n\}$, $a_1 < a_2 < \dots < a_n$

$B = \{b_1, \dots, b_n\}$, $b_1 < b_2 < \dots < b_n$

$C = \{c_1, \dots, c_n\}$, $c_1 < c_2 < \dots < c_n$

$A_1 := \{a_1+b_1, a_1+b_2, \dots, a_1+b_n\} \rightsquigarrow a_1+b_1 < a_1+b_2 < \dots < a_1+b_n$

$A_2 := \{a_2+b_1, a_2+b_2, \dots, a_2+b_n\} \rightsquigarrow a_2+b_1 < a_2+b_2 < \dots < a_2+b_n$

$\vdots$ $\qquad\qquad$ $\vdots$

$A_n := \{a_n+b_1, a_n+b_2, \dots, a_n+b_n\} \rightsquigarrow a_n+b_1 < a_n+b_2 < \dots < a_n+b_n$

- now we need to check whether A_i and C [redacted] contain a same element ; do this check for $\forall i$
- do this using a linear scan because A_i and C are both sorted
  - $\hookrightarrow$ we start from the smallest elements in both sets, meaning a_i+b_1 and c_1 : ask $a_i+b_1 \stackrel{?}{=} c_1$, then $a_i+b_1 \stackrel{?}{<} c_1$ and move to the next element in the set whose current element was smaller $\Rightarrow O(n)$ to inspect everything

$$T(n) = n \cdot O(n) = O(n^2)$$

b) we need to ask $\Omega(n^2)$ queries to be sure that no bad triple with $a+b=c$ exists

let us consider the following sets:

$$A = \{-10n + 4i \mid 0 \le i < n\}$$
$$B = \{10n + 4j \mid 0 \le j < n\}$$
$$C = \{4k + 1 \mid 0 \le k < 2n\}$$

- now, every $a+b = -10n + 4i + 10n + 4j = 4(\underbrace{i+j}_{\in\{0,1,\dots,2(n-1)\}})$ and every $c = 4k + 1$

$\Rightarrow a + b \neq c$ for all $a \in A, b \in B, c \in C$

- we claim that the algorithm has to consider all pairs (a_i, b_j) to be able to decide whether a bad triple exists or not
- if an algorithm does not consider all pairs (a_i, b_j) and
  - a) answers YES (there is a triple s.t. $a+b=c$), then the alg. is wrong on our example
  - b) answers NO (there is no triple s.t. $a+b=c$), then we change the instance: $a_i' = a_i + \frac{1}{3}$, $b_j' = b_j + \frac{1}{3}$ where (a_i, b_j) was the pair the alg. did not consider and we set $C_{k}' = C_k - \frac{1}{3}$, where $k = i+j$

$\Rightarrow a_i' + b_j' = C_k'$, so the alg. is again wrong

→ meaning that it [redacted] asks less than n^2 queries

$\Rightarrow$ alg. has to consider all pairs and there are n^2 of them