Algorithm Design, Assignment 1

Akiko Takeda (Problems are given by Yutaro Yamaguchi)

Due date: December 9, 2014

Choose two problems from the below and solve them. You must choose one from 1–3, and the other from 4–6. The notation $\mathrm{O}(f(x))$ or $\Omega(f(x))$ means that $cf(x)$ is an upper-bound or lower-bound, respectively, for some constant c.

1. Consider the following local-improvement algorithm for finding a stable matching between two disjoint sets M and W of the same size.

   Step 0. Choose an arbitrary perfect matching $S \subseteq M \times W$.

   Step 1. While there exists an instability (blocking pair) with respect to S, repeat the following procedures: choose an arbitrary instability $(m, w) \in M \times W$, and update $S \leftarrow (S \setminus \{(m, w'), (m', w)\}) \cup \{(m, w), (m', w')\}$, where $m' \in M$ and $w' \in W$ satisfy $(m, w'), (m', w) \in S$.

   This algorithm does not always halt. Prove it by using the following instance.

   For $M = \{a, b, c\}$ and $W = \{x, y, z\}$, the preference of $m \in M$ is given as the total order $\prec_m$ on W, and that of $w \in W$ as $\prec_w$ on M as follows:

   $$\begin{array}{ccc} y \prec_a x \prec_a z, & x \prec_b z \prec_b y, & x \prec_c y \prec_c z \\ a \prec_x c \prec_x b, & c \prec_y a \prec_y b, & a \prec_z c \prec_z b. \end{array}$$

   Here, e.g., the relation $y \prec_a x$ means that a prefers y to x.

2. It is well-known that a maximum weight spanning tree of a connected undirected graph with nonnegative weight on each edge can be found by a greedy algorithm (check each edge e in decreasing order of the weights, and select e unless e and some selected edges form a cycle). Related to this, solve one of the following problems (a) and (b).

   (a) Relax the spanning-tree constraint so that a set of selected edges can contain at most one simple cycle (a cycle without intersecting the same vertex in between). Prove that a similar greedy algorithm (check each edge e in decreasing order of the weights, and select e unless e and the selected edges contain at least two different simple cycles) correctly returns an optimal solution in this case.

   (b) Relax the spanning-tree constraint so that a set of selected edges can contain at most two simple cycle. Show an example that a similar greedy algorithm (check each edge e in decreasing order of the weights, and select e unless e and the selected edges contain at least three different simple cycles) fails to return an optimal solution in this case.

3. Let a, b be positive integers with $1 < a < b$. Consider the situation when we pay an arbitrary positive-integer amount of money by using only three kinds of coins whose values are $1, a, b$. Show two pairs (a, b) such that for one pair the following greedy algorithm always returns a payment with the fewest coins, and for the other pair it does not always returns such a payment.

   Greedy Algorithm Use as many coins as possible in decreasing order of the values, i.e., for the payment x, use $\lfloor x/b \rfloor$ b-coins and $\lfloor (x - b\lfloor x/b \rfloor)/a \rfloor$ a-coins, and pay the rest by 1-coins.

4. Let n, N be positive integers. Given N coins among which exactly one is fake and lighter than the others, we want to find the fake coin by using a balance which can just compare the weights of two sets of coins. Show the maximum N such that n comparisons are sufficient to find the fake coin among N coins, and prove the correctness.

5. Let n, k be positive integers with $k \leq n$. The following algorithm is to find the k-th minimum integer among given n distinct integers.

   Find(S, k)

   Input: A set S of n distinct integers, and an integer k with $1 \leq k \leq n$.

   Output: The k-th minimum integer in S.

   Step 0. If $n < 25$, sort all integers in S in increasing order, and return the k-th integer of the sorted sequence.

   Step 1. Partition S into subsets $S_1, S_2, \ldots, S_l$ with $|S_1| = |S_2| = \cdots = |S_{l-1}| = 5$ and $1 \leq |S_l| \leq 5$.

   Step 2. For each $i = 1, 2, \ldots, l$, let m_i be the median of the integers in S_i (here, defined as the $\lceil |S_i|/2 \rceil$-th minimum integer), and $M \leftarrow \{ m_i \in S \mid i = 1, 2, \ldots, l \}$.

   Step 3. By Find$(M, \lceil l/2 \rceil)$, compute the median m of the integers in M.

   Step 4. Let $S_- \leftarrow \{ s \in S \mid s \leq m \}$, $S_+ \leftarrow \{ s \in S \mid s > m \}$.

   Step 5. If $|S_-| \geq k$, return the output of Find(S_-, k). Otherwise (i.e., if $|S_-| < k$), return the output of Find$(S_+, k - |S_-|)$.

   Prove that this algorithm computes the k-th minimum number in $\mathrm{O}(n)$ time, along the following procedures.

   (a) Confirm that the output is correct.

   (b) Show upper-bounds of $|S_-|$ and $|S_+|$ using n.

   (c) Let $T(n)$ denote the computational time of Find(S, k). Using this T, estimate the computational time of each step.

   (d) Based on the above estimations, show a recurrence relation (inequality) for $T(n)$, and prove $T(n) = \mathrm{O}(n)$.

6. Let n be a positive integer. Given a sequence $a_1, a_2, \ldots, a_n$ of n integers, we want to find a consecutive subsequence whose sum is maximum. If you check all patterns of the starts and ends of consecutive subsequences, the combination is $\binom{n+1}{2} = \Omega(n^2)$,

and hence it takes $\Omega(n^2)$ time. Based on the following divide-and-conquer idea, construct an algorithm (show concrete procedures) that returns a maximum-sum consecutive subsequence in $\mathrm{O}(n \log n)$ time, and prove the correctness.

Divide Partition the given sequence into two consecutive subsequences at the center (or near the center).

Conquer Find maximum-sum consecutive subsequences of the left and right consecutive subsequences separated above and one across the partitioning point, and adopt one with the sum maximum among the three.