MALMÖ UNIVERSITY
Computer Science

Advanced Programming
Fall 2012

Advanced Programming Exam

10 January 2013 kl. 8.15–12.15

Instructions: Treat at most one question per sheet, write only on one page of each sheet and mark every page with your initials. Do not use red ink! Write legibly and comment your work extensively, this can give you extra marks even if the result is wrong. The exam contains 5 question and each question is worth 10 points.

Allowed Material: The course books (Dasgupta, Papadimitriou, Vazirani. *Algorithms*; Skiena, Revilla. *Programming Challenges*; Bentley. *Programming Pearls*), course notes and other printed material, pen or pencil and paper. **Good Luck!**

---

1. Let A be an array with n entries holding distinct numbers. The array is *unimodal*, i.e., there is an index k in the array such that the values increase up to index k and then decrease the remainder of the way to index $n-1$.

   Give an algorithm that finds the "peak entry" k using at most $O(\log n)$ reads of the entries of A. (10p)

   (Hint: Use divide and conquer.)

2. The number of combinations possible when picking k items among n distinct ones is denoted $C(n,k)$ or sometimes $\binom{n}{k}$. $C(n,k)$ is defined as:

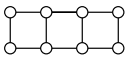
$$C(n,k) = \begin{cases} 1 & \text{if } n \geq 0 \text{ and } k = 0. \\ 1 & \text{if } n \geq 0 \text{ and } k = n. \\ C(n-1,k) + C(n-1,k-1) & \text{if } n > 0 \text{ and } 0 < k < n. \end{cases}$$

   (a) Construct an algorithm that computes $C(n,k)$ with the use of dynamic programming. (5p)

   (b) Analyze the complexity of your algorithm in terms of n and k. (2p)

   (c) Use your algorithm to compute $C(3,2)$. (3p)

3. A simple undirected graph is called *bipartite*, if the nodes of the graph can be partitioned into two subsets X and Y so that every edge connects a node in X with a node in Y. Graph (i) below is bipartite whereas graph (ii) is not.


(i)


(ii)

   Design an algorithm that tests if a graph is bipartite. (10p)

   (Hint: Use breadth first search.)

Please turn page!

4. In the INTERVAL SCHEDULING PROBLEM, we are given n jobs $u_1, u_2, \ldots, u_n$ where each job $u_i = (s_i, t_i)$ has a starting time s_i and a finishing time t_i, i.e., job u_i must be processed in the time interval between s_i and t_i. Note that only one job can be executing at any one time so if time intervals between two different jobs overlap, it is impossible to execute both of them. We say that a subset of the n jobs is *compatible*, if no two jobs have overlapping time intervals (except at the interval end points). The problem is to find the largest compatible subset of n jobs.

   EXAMPLE: $u_1 = (0, 4)$, $u_2 = (1, 3)$, $u_3 = (3, 5)$, $u_4 = (4, 6)$, $u_5 = (7, 9)$, are five jobs. The subset $\{u_1, u_4, u_5\}$ is compatible since no two jobs overlap.

   A greedy algorithm to find a compatible subset of jobs works by using a rule to select one job u_{i_1} to include in the solution. Those jobs that overlap with u_{i_1} are discarded. It then uses the rule to select the next job u_{i_2} for the solution, discard those that overlap this job and so on until every job has either been included in the solution or discarded.

   Consider the following three possible rules to use for the selection.

   1. Select the job u_i that has the earliest starting time s_i.
   2. Select the job u_i that has the earliest finishing time t_i.
   3. Select the job u_i that has the shortest processing time $t_i - s_i$.

   QUESTIONS:

   (a) One of these rules will guarantee that the greedy algorithm finds the largest compatible subset. Which rule is it? (2p)

   (b) Find counterexamples for the other two rules proving that they do not necessarily yield largest compatible subsets. (8p)

5. Given a convex polygon $\mathcal{P}$ in the plane, represented by an array of n vertices in their order along the boundary. Describe an algorithm that, given a query point q, decides whether q is inside $\mathcal{P}$ in $O(\log n)$ time. (10p)

   (Hint: Use divide and conquer.)