Omtenta i Algoritmer[1]

Jan. 15, 2005
kl 0830 - 1230
byggnad V

Ansvarig:
Devdatt Dubhashi Tel. 073 932 2226 Rum 6479 EDIT

Poäng:	60	
Betygsgränser:	Chalmers	5:48, 4:36, 3:24
	GU	VG:48, G:28
	Doktorander	G:36
Hjälpmedel:	kursboken, anteckningar	

- Recommended: First look through all questions and make sure that you understand them properly. In case of doubt, do not hesitate to ask.
- **Answer all questions in the given space on the question paper (the stapled sheets of paper you are looking at). The question paper will be collected from you after the exam. Only the solutions written in the space provided on the question paper will count for your points.**
- Use extra sheets only for your own rough work and then write the final answers on the question paper.
- Answer concisely and to the point. (English if you can and Swedish if you must!)
- Code strictly forbidden! Motivated pseudocode or plain but clear English/Swedish description is fine.

Lycka till!

[1] 2004 läsperiod 1, INN200 (GU) / TIN090 (CTH).

Problem 1 Minmax pair [**10**] The input is a set S of n integers $x_1, \cdots, x_n$ where n is even. Design an algorithm to partition the input into $n/2$ pairs such that the following is true. For each pair (x_i, x_j), we compute the sum $x_i + x_j$. Denote these sums by $s_1, s_2, \cdots, s_{n/2}$. Your algorithm should find a partition that *minimizes* the *maximum* sum. Give a brief argument to justify why your algorithm is correct and state its running time.

Problem 2 Spanning trees vs Shortest paths [**10**] Hacker Kalle has been assigned some tasks by his boss. In each of the following two cases, say whether he is correct or not. If correct, sketch a proof for him highlighting the key points. If incorrect, give a *concrete* counter-example to convince him.

(a) Given is a map of cities represented by an undirected graph $G = (V, E)$ with distances between cities given by weights $w_e, e \in E$. Kalle is required to find the shortest distance (as per the weights $w_e, e \in E$) between two given cities u and v. Kalle proeceeds as follows: he computes a minimum cost spanning tree T of G and then computes the weight of the unique path between u and v in T

(b) Kalle now wants to compute a minimum spanning tree of G. For this he applies Dijkstra's algorithm for shortest paths (regarding each undirected edge of G as two directed edges oppositely oriented). He makes the observation that Dijkstra's algorithm computes a directed tree rooted at the source. Let T be this tree and now erase the orientations of the edges. Kalle claims the result is a minimum cost spanning tree of the original undirected graph G.

Problem 3 Investment [**10**] You are a rich venture capitalist and have SEK N million to invest in any of m start-ups. Assume that N is an integer and that all investments are in units of a million SEK. Your chief financial officer has prepared a table `invReturn` of expected returns on your investments: `invReturn[d,j]` gives the expected return from an investment of SEK d million in start-up j (you may assume that the columns of `invReturn` are non-decreasing i.e. investing more money in any of the start-ups won't decrease your return). You want to find a strategy of investment that maximizes your total return. Fill in the following outline of a dynamic programming solution to the problem.

(a) State the optimal substructure property in this example.

(b) Let $R(d, i)$ denote the maximum return possible with a total of SEK d invested in start-ups $1 \cdots i$ (for $0 \leq d \leq N$ and $1 \leq i \leq m$. Write a recurrence for $R(d, i)$. Give the base case as well as the recurrence.

(c) Based on the recurrence in (b), give an algorithm to find the maximum possible return. Give the time and space complexity of your algorithm.

(d) Show how to modify your algorithm to also output the optimal investment plan i.e. how much to invest in each start-up. Give the time and space complexity of the modified algorithm.

(e) Does your algorithm work if we do not assume that the columns of `invReturn` are non-decreasing i.e. if investing more in some start-up *could* reduce the return?

Problem 4 Significant inversions [10] The input is an array $A[1 \dots n]$ of `comparable` elements. A pair (i, j) is called a *significant inversion* if $i < j$ but $a_i > 2a_j$. Give an efficient Divide-and-Conquer algorithm to count the number of significant inversions in a given array. You should state the algorithm clearly, say in pseudocode giving the divide step and the conquer step. Analyse the running time of your algorithm by writing a recurrence for it.

Problem 5 Improving Prim? [10] Recall that Prim's algorithm using the binary heap data structure to implement the interface consisting of the `extractMin()` and `decreaseKey()` methods runs in time $O(|E| \log |V|)$ on an input graph $G = (V, E)$. Using the *Fibonacci Heap* data structure, this can be improved to $O(|E| + |V| \log |V|)$. (You don't need to know the Fibonacci Heap data structure for this problem!) Show that this running time is optimal i.e. *any* implemenattion of Prim's algorithm *must* take time $\Omega(|E| + |V| \log |V|)$.

Problem 6 Cybercommunities [10] An interesting problem in internet algorithmics is to find a collection of densely connected web sites i.e. set of web sites which have a lot of hyperlinks between each other. Such collections are called *cybercommunities* - for example, the set of sites discussing *Harry Potter* films is one such cybercommnunity. In graph theory terms, the concept is captured by the notion of a *clique*: a clique is an undirected graph $G = (V, E)$ is a subset $U \subseteq V$ such that for any two vertices $u, v \in U$, $(u, v) \in E$. Consider the optimization problem of finding the size of the largest clique in a given graph.

(a) Formulate a decision problem corresponding to whether or not a graph has a clique of a certain size. Show that the decision problem and the optimization problem are polynomial-time equivalent i.e. if one can be solved in polynomial time, so can the other.

(b) Show that the decision problem is in $\mathcal{NP}$.

(c) Show that the decision problem is $\mathcal{NP}$-complete by giving a reduction from the *Independent Set* problem (which is know to be $\mathcal{NP}$-complete).