Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

WS 1999/2000
Problem Set 6
December 7, 1999

Efficient Algorithms and Data Structures

Problem 1 (8 credits)

Let $G = (V, E)$ be a connected undirected graph with weight function $w : E \to \mathbb{R}$, and suppose that $|E| \geq |V|$. Assume that G has at least two different spanning trees and that not all of them have the same weight.

a) Let T be a minimum spanning tree of G. Prove that there exist edges $(u, v) \in T$ and $(x, y) \notin T$ such that $(T \setminus \{(u, v)\}) \cup \{(x, y)\}$ is a second-best minimum spanning tree of G. (A spanning tree T' is called second-best if its weight is strictly larger than the weight of the minimum spanning tree but minimum among all spanning trees with this property.)

b) Let T be a minimum spanning tree of G and, for any two vertices $u, v \in V$, let $\max[u, v]$ be an edge of maximum weight on the unique path between u and v in T. Describe an $O(|V|^2)$-time algorithm that, given T, computes $\max[u, v]$ for all $u, v \in V$.

c) Give an efficient algorithm to compute the second-best minimum spanning tree of G.

Problem 2 (8 credits)

With the minimum tardy task scheduling problem, we are given a set $T = \{t_1, t_2, \ldots, t_n\}$ of n tasks, with a length $\ell(t_i)$, a weight (penalty) $w(t_i)$ and a deadline $d(t_i)$ associated with each task $t_i \in T$. We are required to find a feasible subset S of T (each scheduled task in S finishes before its deadline and at most one task is executed at a time) such that the sum of the weights of the unscheduled tasks (those that are in $T \setminus S$) is minimum. An obvious greedy algorithm for constructing S, for the special case where $\ell(t_1) = \ell(t_2) = \cdots = \ell(t_n) = 1$, is to add at each step the task with the highest weight among those not yet considered, provided that the chosen set of tasks remains feasible.

Consider the greedy procedure *Min_Tardy_Task*(T) which assumes that the tasks are numbered in such a way that $w(t_1) \geq w(t_2) \geq \cdots \geq w(t_n)$.

> **procedure** *Min_Tardy_Task*(T)
> $S \leftarrow \emptyset$
> **for** $i = 1$ **to** n
>   **if** $S \cup \{t_i\}$ is feasible
>   **then** $S \leftarrow S \cup \{t_i\}$
> **return** $W = \sum_{t \in T \setminus S} w(t)$

Recall that the tasks in $S \cup \{t_i\}$ are first sorted by increasing deadlines and then scheduled in order to check feasibility.

a) What is the growth rate of *Min_Tardy_Task*(T)?

b) Can you try and make it more efficient by choosing better data structures?

c) Prove that *Min_Tardy_Task*(T) gives the optimal schedule.

Problem 3 (8 credits)

Let $P_1, P_2, \ldots, P_n$ be n programs to be stored on a disk. Program P_i requires s_i kilobytes of storage, and the capacity of the disk is D kilobytes, where $D < \sum_{i=1}^{n} s_i$.

a) We want to maximize the number of programs held on the disk. Prove or give a counter-example: we can use a greedy algorithm that selects programs in order of nondecreasing s_i.

b) We want to use as much of the capacity of the disk as possible. Prove or give a counter-example: we can use a greedy algorithm that selects programs in order of nonincreasing s_i.

Problem 4 (8 credits)

Let $P_1, P_2, \ldots, P_n$ be n programs to be stored on a tape. Program P_i requires s_i kilobytes of storage; the tape is long enough to hold all the programs. We know how often each program is used: a fraction π_i of requests concern program i (and so $\sum_{i=1}^{n} \pi_i = 1$). Information is recorded along the tape at constant density, and the speed of the tape drive is also constant. After a program is loaded, the tape is rewound to the beginning. If the programs are held in the order $i_1, i_2, \ldots, i_n$ the average time to load a program is therefore

$$\bar{T} = c \sum_{j=1}^{n} \left(\pi_{i_j} \sum_{k=1}^{j} s_{i_k} \right),$$

where the constant c depends on the recording density and the speed of the drive. We want to minimize $\bar{T}$ using a greedy algorithm. Prove or give a counter-example for each of the following:

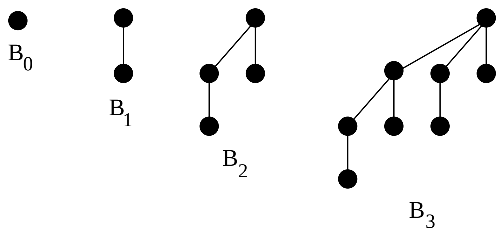
a) We can select the programs in order of nondecreasing s_i.

b) We can select the programs in order of nonincreasing π_i.

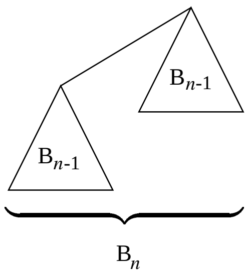
c) We can select the programs in order of nonincreasing π_i / s_i.

Problem 5 (8 credits)

Binomial trees are defined recursively by



and in general:



(1) a) Show that the root node of B_n has degree n.

b) Show that if the children of the root node are numbered from left to right by $n-1, n-2, \ldots, 1, 0$, then child i is the root of a subtree B_i.

(2) Define the path length of B_n to be the sum of the levels of all its nodes, and denote it by P_n. Find a closed form for P_n.