Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Dr. Alexander Souza
Thomas Janson

WS 2011/12
Sheet 06

Algorithm Theory

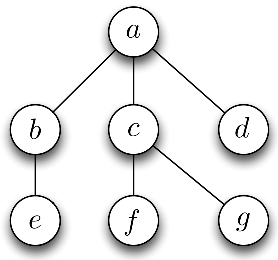
Exercise 1 (Union-find) [Points: 5]

Consider the union-find data structure with bottom-up trees introduced in the lecture (the root of the first two trees becomes the root) and with path compression (all nodes on a find-set-path become children of the root). For any n, state a sequence of n find-set operations and $\mathcal{O}(n)$ union operations starting on a sufficiently large universe that each find-set operation has running time $\Omega(\log n)$.

Exercise 2 (Union-find) [Points: 5]

Use the union-find data structure with bottom-up trees with the weighted union and path compression optimization.

(a) Write an operation sequence of make-set, union, and find-set which can produce the following data structure



(b) We are given a graph $G = (V, E)$ and a graph coloring $c : V \to \mathcal{C}$ where $\mathcal{C}$ is a set of colors.

- A node set M is *monochrome connected* if each pair of nodes in M is connected by a path where all nodes have the same color.
- A monochrome connected set M is *maximal* if there exists no monochrome connected set M with $M \subset M'$.

Create an algorithm based on a union-find data structure which computes the maximal monochrome connected sets of a graph $G = (V, E)$.

(c) Give a preferably small upper bound for the asymptotic running time of your algorithm. Justify your answer.

Algorithm 1 Maximum monochrome connected sets

Input: graph $G = (V, E)$, graph coloring $c : V \to \mathcal{C}$

Output: union-find data structure where find-set$\,(u)$ = find-set$\,(v)$ iff u and v are in the same maximal monochrome connected set

Exercise 3 (Greedy algorithm) [Points: 7]

We are given n jobs which can be executed on a given single machine. Each job j has weight w_j and execution time p_j. The jobs are processed in some sequence $S = (s[1], s[2], ..., s[n])$ where $s[i]$ denotes the job executed at i-th position. Suppose that job j is in position k, i.e., $j = s[k]$, then its completion time is

$$c_j = \sum_{i=1}^{k} p_{s[i]}$$

Our objective function is to minimize $\sum_j w_j \cdot c_j$ over all possible sequences.

(a) Assume we have equal weights $w_j = 1$ and for $n = 4$ the execution times are $p = (5, 7, 8, 9)$. Calculate the objective function for the sequences $S_1 = (1, 2, 3, 4)$, $S_2 = (1, 4, 2, 3)$, $S_3 = (4, 2, 3, 1)$, and $S_4 = (4, 3, 2, 1)$.

(b) Proof that it is an optimal solution to sort the n jobs in increasing order for the special case $w_j = 1$.

(c) Prove that it is in general optimal to sort the jobs according to increasing p_j/w_j ratio.

Exercise 4 (Greedy algorithm) [Points: 3]

Give a greedy algorithm which solves the activity selection problem optimally and sorts the n activities $a_i = [s_i, f_i)$, $i = 1, ..., n$ such that

$$s_1 \leq s_2 \leq ... \leq s_n.$$