Algorithm Design - Homework 1

Academic year 2017/2018

Instructor: Prof. Stefano Leonardi

November 15, 2017

Due date: December 10, 11.59pm

Make sure that the solutions are typewritten or clear to read. A complete answer consists of a clear description of an algorithm, followed by an analysis of its running time and a proof that it works correctly. Exercises 1 and 2 require an implementation of the algorithm. Each exercise can be awarded a maximum of 6 points.

Send your solutions to *schwiegelshohn"at"dis.uniroma1.it* (replace "at" with @) and keep a copy for yourself. Office hours are held for the duration of the assignment weekly on Wednesday 2.00pm to 3.00pm in Via Ariosto 25, Room B118. Solutions will be posted or presented after due date. In the final exam, you will be asked to explain your solutions and/or to go over your mistakes.

Collaboration policy. You may discuss the homework problems with other members of the class, but you must write up the assignment alone, in isolation. Also, you must understand your solutions in detail and be able to discuss your choices and their motivations in detail with the instructor. Finally, you should cite any sources you use in working on a homework problem.

Late policy: Every homework must be returned by its due date. Homeworks that are late will lose 10% of the grade if they are up to 1 day (24h) late, 20% if they are 2 days late, 30% if they are 3 days late, and they will receive no credit if they are late by more than 3 days.

Please refer to course's Web page for detailed information about above aspects.

Exercise 1. You are given an array A of n positive real numbers. The task is to find a subarray $A[i, j]$, i.e. starting at $A[i]$ and ending at $A[j]$ such that

$$\prod_{k=i}^{j} A_k$$

is maximized. Your algorithm only has to output the *value* of $\max_{1 \le i \le j \le n} \prod_{k=i}^{j} A_k$. Finding i and j is not conceptually different, but requires (slightly) more effort.

In addition to arguing correctness and running time, implement your algorithm in either Java, Python, or C++ and supply us with a link or a print out of your code. For full marks the algorithm must run in linear time ($O(n)$).

Exercise 2. Assume we are given a power network $G = (V, E)$. We assume that G is a tree, i.e. a connected graph with no cycles. We say a hub conductor is a node v, such that upon removing v, the induced subgraph $G' = (V \setminus \{v\}, E \cap (V \setminus \{v\} \times V \setminus \{v\}))$ consists of connected components of size at most $\frac{1}{2} \cdot |V|$. The task is to find a hub conductor. If multiple hub conductors exist, the algorithm has to find only one.

In addition to arguing correctness and running time, implement your algorithm in either Java, Python, or C++ and supply us with a link or a print out of your code. For full marks the algorithm must run in linear time ($O(|V| + |E|)$).

Exercise 3. Let $G = (V, E)$ be a graph. Show that the follwing two problems are NP-hard.

1. G has a spanning tree where every node has at most k neighbors and k is part of the input.
2. G has a spanning tree where every node has at most 5 neighbors.

Hint: First consider a suitable hard problem for some appropriate choice of k. Then try to generalize your reduction for $k = 5$.

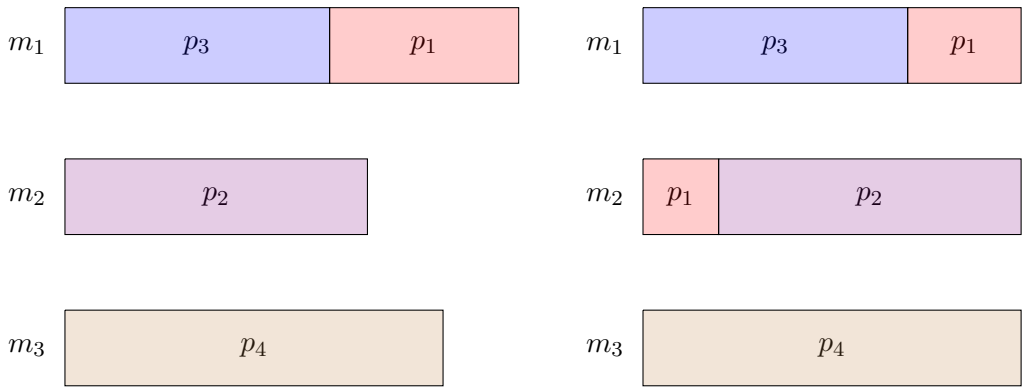
Exercise 4. We are given n jobs with processing times $p_1, \dots p_n$ and m machines. We assume that our machines support interruptions, e.g. a job j with $p_j = 5$ may be processed on a machine m_1 for 3 units of time, interrupted, and subsequently resumed processing for the remaining 2 units of time on machine m_2. A job *cannot* be executed simultaneously on two different machines.

1. We want to schedule the jobs on the machines such that the completion time $C_{\max}$ of the final job is minimized.

   For full marks the algorithm must run in linear time ($O(n)$). In particular the running time must be *independent* of the size of the jobs.

2. Show that the problem is NP hard even on 2 machines if interruptions are *not* allowed.

As an example, consider four jobs with processing times $p_1 = 5$, $p_2 = 8$, $p_3 = 7$, and $p_4 = 10$ to be scheduled on three machines. The schedule on the left is *not* optimal and has $C_{\max} = C_{p_1} = 12$. The schedule on the right is optimal with a completion time of $C_{\max} = 10$.



Exercise 5. A board game expert is using an algorithm to optimize his playing strategy. The game is based on a set of n challenges. In order to complete a challenge, the board game player has to acquire a set of resources. Challenge C_i, $i \in \{1, .., n\}$, requires resources $r_1, \dots, r_{n_i}$.

Challenge C_i gives p_i points when the game player has acquired all the resources needed to complete the challenge.

In order to obtain the resources, the board game expert needs to buy resource cards. The resource card r_j, $j \in \{1, \ldots, m\}$, will cost c_j points. Once the game player holds a resource, it can be used for *all* challenges that require this resource, i.e. resource r_1 is bought only once, and not for every completed challenge. You may assume that resources and points are positive integers.

The net revenue of the game player is equal to the total revenue of the challenges he wins minus the total cost of the cards he buys.

Design an algorithm based on max-flow/min-cut which provides an optimal solution to this problem. Provide a formal proof of the optimality of the algorithm.

Hint: Design a network with:

- a source node s connected to a node representing the resource card r_j with an edge of cost c_j,
- a node representing challenge C_i connected to sink t with an edge of cost p_i,
- an edge of cost infinity connecting each node with resource r_j to any challenge C_i that requires r_j for completion.