Advanced Algorithms Lecturer: Mohsen Ghaffari

Graded Homework 1

Deadline: 11/05/2019, 11:59 pm

Note: Your solutions must be typeset in LaTeX and should be emailed to brandts@ethz.ch by 11:59 pm on November 5, 2019. Please make sure to use the subject "Advanced Algorithms 2019: GHW1" for your email, without the quotation marks.

1 DAG Subgraph (10 points)

You are given an n-node directed graph $G = (V, E)$. Your objective is to select a subset of edges $E' \subseteq E$ that maximizes $|E'|$ subject to the constraint that edges of E' should define a directed acyclic graph, i.e., there should not be any directed cycle in the edges of E'. Devise a polynomial-time 2-approximation algorithm for this problem.

2 Disjoint Triangles (10 points)

You are given an n-node undirected graph $G = (V, E)$. Devise a polynomial-time 3-approximation algorithm for the problem of outputting the maximum number of vertex-disjoint triangles in G. A triangle is a set of three nodes that are all neighbors of each other in G.

3 Machines (20 points)

In this task we ask you to generalize the algorithm for 2-approximation of the minimum makespan problem that you know from the lecture. You are given a sequence of n jobs with nonnegative integer *lengths* given by sequence $p_1 \leq p_2 \leq \ldots \leq p_n$. Your task is to schedule them to m machines. You are also given the sequence of *speeds* of these machines; this is a sequence of integer values $s_1 \geq s_2 \geq \ldots \geq s_m > 0$ such that the time needed for job i to finish on machine j is $\frac{p_i}{s_j}$. The cost of an assignment of jobs to machines is its makespan, i.e., the overall time needed for all machines to finish. Your task is to find a 2-approximation algorithm for this task that runs in time polynomial in the input size. Here is an idea for a procedure that you may find helpful to use in your algorithm (it is not necessarily the only path to the solution). If you decide to use it, you have to prove that it really works as claimed below. The procedure is given the sequences $p_i, 1 \leq i \leq n$ and $s_j, 1 \leq j \leq m$, as well as a value D. The procedure either finds a solution with makespan at most $2D$ or it claims that there is no solution with makespan at most D. The procedure iterates over machines in the reverse order (from machine m to machine 1). For j-th machine the procedure iterates the following. In one step it finds a maximum length job among those that were not assigned to a machine yet and that have length at most $s_j \cdot D$. This job is then assigned to machine j. If the total time of jobs assigned to this machine is bigger than D (this means their total length is bigger than $s_j \cdot D$) or there are no more jobs that can be assigned to machine j, the procedure proceeds with the next machine. When the procedure finishes, it returns the found assignment if all jobs were assigned. Otherwise, it claims there is no solution with time at most D.

4 Tolls (20 points)

You are given a directed graph G with n vertices and m directed edges. Each edge e connects two different vertices u, v of G and represents a path that you can take to travel from u to v.

Each edge has certain integer *length* $l(e) \geq 0$ and integer *cost* $c(e) \geq 0$: whenever you decide to use this edge, you have to pay $c(e)$ Swiss francs as a toll and it takes you $l(e)$ minutes to travel from u to v via e. Your goal is to find the length of a shortest path from a given vertex s to a given vertex t, subject to the constraint that the sum of costs over the edges of the path is at most the given integer value $C \geq 0$.

1. (10 points) Give an algorithm for the above problem with time complexity that is polynomial in n and L, where L denotes the maximum length $l(e)$ over all edges.
2. (10 points) *You can now assume that you have an algorithm from the first subtask even if you did not solve it.* Provide an FPTAS for this problem. This means device an algorithmn that is given $\varepsilon > 0$ as input and runs in time polynomial in the length of the input and $1/\varepsilon$. The algorithm outputs a solution of total cost at most C such that its length its at most $1 + \varepsilon$ times the optimum achievable length.

5 DNF Score (20 points)

The input is a DNF formula f with n Boolean variables $x_1, \ldots, x_n$, as well as positive integer weights a_i and b_i for each $i \in \{1, \ldots, n\}$. For each particular assignment of n Boolean variables, the *score* of the assignment is defined as the multiplication of the scores of its n variables, where for the i^{th} variable, we get a score a_i if that variable is set to true and score b_i otherwise. Devise an FPRAS for approximating the summation of the scores of all satisfying assignments.

6 Dominating Set Approximation (20 points)

You are given as input an n-node undirected graph $G = (V, E)$ with maximum degree Δ. The objective is to find a dominating set $S \subseteq V$ with minimum cardinality $|S|$. A set $S \subseteq V$ dominates a vertex $v \in V$ iff v or at least one of its neighbors is in S (We emphasize that the or is not exclusive). A set S is called a dominating set if it dominates all vertices in V.

1. (5 points) Formulate the minimum dominating set problem as a linear programming.
2. (10 points) Devise a randomized rounding that turns the fractional solution to an integral solution T which is at most $O(\log \Delta)$ factor larger, and that dominates all but at most n/Δ^2 vertices (say, in expectation).
3. (5 points) Explain how to augment the above integral solution T to a set S that dominates all vertices, while maintaining an $O(\log \Delta)$ approximation.