INTRO TO APPROXIMATION – HW1

Deadline: Monday **November 20, 2017 15:39**. Send your solutions preferably by email (in PDF, ODT …or just inside the email). I also accept scans or photos of quality high-enough to read everything without problems. Another possibility is to hand your solutions in on a paper at the beginning of a class. Please ask if any homework is not clear to you.

Homework one — **Warmup with tokens.** — **[3 points]**

Consider 10 tokens with values $1, 2, \dots 10$. If we split them into two groups – even valued and odd valued – the average of the odd group is 5 and the average of the even group is 6. Prove or disprove the following:

a) Can we reorder the tokens into two different groups so that the average of both groups increases? If not, argue why, and if yes, show such groups.

b) Can we reorder the tokens into two different groups so that the average of both is above 5.5?

Homework two — **Steiner tree** — **[5 points]**

In the *Steiner tree problem* we get on input a connected undirected graph $G = (V, E)$, an edge cost function $c : E \to \mathbb{R}^+$, and finally a list of *terminals* $S \subseteq V$. A feasible solution to our problem is any subset of edges $E' \subseteq E$ so that the graph $G' = (V, E')$ has all the terminals in one connected component. We aim to minimize the cost of E', i.e. $\sum_{e \in E'} c(e)$. Your task is to design a 2-approximation algorithm.

Hint: Start with the case in which edges satisfy triangle inequality (that is, the shortest path between vertices connected by an edge e is this edge e).

Homework three — **Scheduling on machines with speeds** — **[6 points]**

We have m machines with speeds $s_1 \geq s_2 \geq \cdots \geq s_m$ on which we want to schedule n jobs (so that one machine processes at most one job at a time). Now, processing the j-th job on machine i takes p_j/s_i units of time, where p_j is the length of the job. (The schedule starts at time 0 and two jobs cannot of course run at the same time on one machine.)

We say that a polynomial-time algorithm is a ρ-relaxed decision procedure if it gets a number D in addition to the intput and either (I) creates a schedule so that all jobs are finished till time ρD, or (II) outputs that there is no schedule that can finish all jobs till time D.

a) First show that one can use a ρ-relaxed decision procedure to find a ρ-approximation algorithm.

b) Then prove that the following algorithm is a 2-relaxed decision procedure: First for each job j we find its type i which is the number of the slowest machine which finishes job j till time D (that is, maximal i such that $p_j/s_i \leq D$). If for some job j there is no machine which finishes it till time D (and thus $p_j/s_1 > D$), the algorithm immediately stops outputting (II).
   After assigning types to jobs, each machine i starts processing jobs of type i so that after finishing a job j machine i continues as follows:
   - If job j finishes at time D or later, the machine stops and processes nothing from now on.
   - If there is no available job of the same type as job j (that is, not finished and not started) the machine starts processing a job of the next type (for which there is an available job). Machine i thus first processes jobs of type i, then of type $i + 1, i + 2, \dots$
   - Otherwise the machine starts a job of the same type as job j.

   If not all n jobs are processed by this procedure, the algorithm stops outputting (II), otherwise it outputs the created schedule containing all jobs.

Homework four — **Looking for TSP instances** **[6 points]**

a) Find an infinite class of graphs showing that the Christofides' algorithm for metric TSP is no better than a 3/2-approximation.
More precisely, for infinitely many n's construct a graph G_n with n vertices so that

$$\frac{\mathrm{ALG}(G_n)}{\mathrm{OPT}(G_n)} \to \frac{3}{2}$$

for $n \to \infty$ where $\mathrm{ALG}(G_n)$ is the cost of the algorithm's solution and $\mathrm{OPT}(G_n)$ is the optimum cost. Similarly as in the class, you can choose the behavior of the algorithm if there are more possibilities (for example, if the graph has more minimum spanning trees).

b) In *asymmetric TSP* our task is to find a minimum-length walk that visits each vertex in a directed graph with edge lengths. While triangle inequality still holds, symmetry $d(u,v) = d(v,u)$ may not hold. Show that a spanning tree tree algorithm cannot be used for any good approximation (constant-, or even $\mathcal{O}(\log n)$-approximation). In this algorithm minimum spanning tree is found in the graph in which we forget edge orientations.
Hint: Find an example of a directed graph with k really long edges (and the rest of edges with length 1). An optimum solution uses only one long edge, while a DFS traversal of a minimum spanning tree (and taking shortcuts) uses all k long edges. How long can these long edges be? (Of course, you can solve this exercise using a different approach.)

Homework five — **Bonus exercise: cubic graphs without bridges [5 points]**

Consider a cubic 2-edge-connected graph G. The word *cubic* means that every degree of the graph is equal to 3. The word *2-edge-connected* means that the graph does not contain a bridge, which is an edge whose removal disconnects the graph. The graph is not weighted, so all the edges have distance one.

1. Show that any such graph has a TSP tour of length at most $4|E|/3$.
2. Prove that the point $(1/3, 1/3, 1/3, \ldots, 1/3)$ lies always in the perfect matching polytope of G.

The perfect matching polytope is this one (it is the set of all feasible solutions of the following linear program):

$$\begin{aligned}
&\forall v \in V\colon && \sum_{e=vx} x_e = 1\\
&\forall S \subsetneq V, S \neq \emptyset, |S| \text{ odd}\colon && \sum_{e \in E(S, V\setminus S)} x_e \geq 1\\
&\forall e \in E\colon && x_e \geq 0
\end{aligned}$$

You can use the following fact: This polytope is equal to the convex hull of (characteristic vectors of) perfect matchings.