

Homework 1

Due: Thursday, Jan. 29 in class.

General remark: whenever you are asked to provide an α -approximation algorithm you need to prove that your algorithm outputs a feasible α -approximate solution.

1. **Greedy algorithms for Job Interval Scheduling** Recall that in the job interval scheduling problem we have a set J of jobs, and each job $j \in J$ has a release date r_j , a deadline d_j and processing time p_j . Job j is associated with a set \mathcal{I}_j of time intervals, the set of all length- p_j intervals contained in $[r_j, d_j]$. The goal is to schedule maximum possible number of jobs on one machine. We have shown in class that the following algorithm achieves a factor 2-approximation:

GREEDY

Among all available jobs, choose job j that has an interval $I \in \mathcal{I}_j$ with left-most right endpoint; schedule j on I and discard all job intervals overlapping with I .

A mirror reflection of GREEDY is an algorithm we call GREEDY':

GREEDY'

Among all available jobs, choose job j that has an interval $I \in \mathcal{I}_j$ with right-most left endpoint; schedule j on I and discard all job intervals overlapping with I .

Algorithm GREEDYx2 Runs GREEDY and GREEDY' and outputs the better of the two solutions.

- (a) Show that GREEDYx2 achieves a factor 2-approximation.
 - (b) Show that your analysis is asymptotically tight.
2. **Machine Minimization** The input to the Machine Minimization problem is the same as for Job Interval Scheduling. The goal is to schedule all jobs using minimum possible number of machines. Each machine can only execute one job at any given time moment. In Restricted Machine Minimization, each job j has $d_j - r_j = p_j$, so $|\mathcal{I}_j| = 1$ for all j .
 - (a) Show an efficient algorithm for solving Restricted Machine Minimization exactly.
For an instance I of the (unrestricted) Machine Scheduling, let $\rho(I) = \frac{\max_{j \in J} \{p_j\}}{\min_{j \in J} \{p_j\}}$.
 - (b) Show a constant factor approximation algorithm for $\rho(I) = 1$ (i.e., all processing times are identical).
 - (c) Show a constant factor approximation algorithm for $\rho(I) \leq 2$.
 - (d) Show an $O(\log(\rho(I)))$ -approximation algorithm for unrestricted Machine Minimization.

3. **Set Cover** Denote by GREEDY the greedy algorithm we defined in class for the unweighted Set Cover problem. Recall that the algorithm chooses at each step a set that covers maximum number of elements that are not covered yet.

Given a set cover instance I , let s_I denote the maximum set size, $s_I = \max_{S \in I} \{|S|\}$. Show that GREEDY achieves an $O(\log(s_I))$ -approximation.

Hint: Observe that $\text{OPT} \geq n/s_I$.

4. **Steiner Tree** In the directed Steiner Tree problem, we are given a directed edge-weighted graph $G = (V, E)$, a root vertex r and a subset $T \subseteq V$ of vertices called terminals. The goal is to find a minimum-cost subset E' of edges, such that in the graph induced by E' , there is a directed path from r to every terminal in T .

- (a) Show an approximation-preserving reduction from Set Cover to directed Steiner Tree.

Hint: Given an instance of set cover, construct a 3-layered instance of Steiner tree. First layer contains the root r , second layer contains Steiner vertices and third layer contains terminals.

- (b) Recall that there is an $O(\log n)$ -approximation algorithm for Set Cover, and that the problem is also hard to approximate up to $\Omega(\log n)$ factor. What is the implication of your reduction to the approximability of directed Steiner Tree?

5. **Local Ratio:** Given a weighted Set Cover instance $I = (U, \mathcal{S})$, where U is the set of elements and \mathcal{S} is a collection of subsets of U , the *frequency* of element i is the number of sets in \mathcal{S} containing i . Let f_I denote the frequency of the most frequent element. Show an f_I -approximation algorithm for weighted Set Cover using the Local Ratio technique.