Fall 2021, CS 583: Approximation Algorithms

Homework 0

Due: 09/02/2021 in Gradescope

Collaboration Policy: This homework is to test your knowledge of pre-requisite material and will not be officially graded. Try to work out the problems on your own but feel free to talk to other students. Everyone should write up their own solutions and clearly indicate who they collaborated with.

What to turn in: Solutions to two or more problems. We want to get a sense of how you write formally and whether you have sufficient background.

I expect you to think about all the problems since future homeworks might make use of some of the observations.

Problem 1 The classical $0, 1$ knapsack problem is the following. We are given a set of n items. Item i has two positive integers associated with it: a size s_i and a profit p_i. We are also given a knapsack of integer capacity B. The goal is to find a maximum profit subset of items that can fit into the knapsack. (A set of items fits into the knapsack if their total size is less than the capacity B.) Use dynamic programming to obtain an exact algorithm for this problem that runs in $O(nB)$ time. Also obtain an algorithm with running time $O(nP)$ where $P = \sum_{i=1}^{n} p_i$. Note that both these algorithms are not polynomial time algorithms. Do you see why?

Problem 2 Consider the following multi-processor scheduling problem. The input consists of n jobs $J_1, J_2, \ldots, J_n$ and m identical machines $M_1, M_2, \ldots, M_m$. Each job J_i has a non-negative size s_i. The goal is to assign the jobs to the machines to minimize the maximum load over all machines. The load of a machine is the sum of the sizes of the jobs assigned to it.

- Prove that the problem is NP-Hard even when $m = 2$. More formally, consider the decision problem that consists of the job sizes, m and and a bound B and the goal is check if the jobs can be scheduled with load at most B. You should be able to use a canonical NP-Hard number problem.
- Now consider the setting where there are only 3 distinct job sizes $\{a, b, c\}$. That is, $s_i \in \{a, b, c\}$ for $1 \leq i \leq n$. Describe a polynomial-time algorithm for this problem. You need not answer this part but can you obtain a polynomial time algorithm when the number of distinct job sizes is at most k where k is some fixed constant?

- Prove that there is a *pseudopolynomial-time* algorithm for the problem when m is a fixed constant (prove it for $m = 2$ if it is easier to think about it).
- Prove that the problem is *strongly* NP-Hard when m is part of the input. In case you are not familiar with strong NP-Hardness it means that the problem is NP-Hard even when numbers are given in unary. *Hint:* Consider the canonical strongly NP-Hard problem 3-Patition (read about it in case you are not familiar).

Problem 3 Ball and bins. Consider throwing n balls into n bins where each ball is thrown independently and uniformly at random into a bin.

- What is the probability that a given bin (say the first bin) is empty?
- What is the expected number of bins that are empty?
- If you know Chernoff bounds prove that the maximum number of balls in any bin is $O(\log n/\log\log n)$ with high probability (with probability at least $1 - 1/\text{poly(n)}$). If you do not know Chernoff bounds, using more elementary methods, show a weaker bound of $O(\log n)$.

Problem 4 Let σ be a uniformly random permutation of $\{1, \ldots, n\}$. That is $\sigma(1), \sigma(2), \ldots, \sigma(n)$ is a permutation and it is chosen uniformly from one of the $n!$ permutations. We say that position i is a peak in σ if $\sigma(i)$ is the maximum number amongst $\sigma(1), \sigma(2), \ldots, \sigma(i)$. For instance if σ is the permutation $3, 4, 1, 2, 5$ then positions $1, 2, 5$ are peaks and positions 3 and 4 are not. Note that position 1 is always a peak. Let σ be a uniform random permutation of $\{1, 2, \ldots, n\}$.

- What is the probability that position i is a peak in σ?
- What is the expected number of peaks in σ?

Problem 5 In the (maximum-cardinality) matching problem, given a graph $G(V, E)$, the goal is to find a largest subset of edges E' such that no two edges in E' share a common vertex. (Equivalently, each vertex must be adjacent to at most one edge in E'.)

1. Write a Linear Program (LP) for the matching problem in bipartite graphs.
2. Write a linear program for the matching problem in general graphs.
3. Write the duals to the primal linear programs from parts 1 and 2.
4. Give an example to show that there is a fractional solution to the LP of part 2 with value strictly greater than that of an optimal integral solution. (That is, the *integrality gap* of this linear program is greater than 1.)

5. Prove that the optimal value to the LP of part 1 is an integer.
   Hint: You may use the fact that the in a network with integer capacities, the value of the maximum flow is integral.

Problem 6 Let G be a complete graph with non-negative edge weights. One can compute in polynomial time a minimum weight perfect matching in G (assuming G has an even number of vertices). We want to use the matching algorithm to solve a problem on directed graphs. Let $H = (V, E)$ be a *directed* graph with non-negative arc weights given by $w : E \to \mathcal{R}^+$. We wish to find a minimum weight collection of vertex-disjoint directed cycles in H such that every vertex is in exactly one of those cycles. Show that one can solve this problem by reducing it to the minimum weight perfect matching problem.
Hint: Split each vertex v in H into two vertices v^- and v^+ with v^- for incoming arcs into v and v^+ for outgoing arcs from v.