

# CS222 Homework 8

Algorithm Analysis & Deadline: 2020-12-05 Saturday 24:00

Exercises for Algorithm Design and Analysis by Li Jiang, 2020 Autumn Semester

1. Suppose you're acting as a consultant for the Port Authority of a small Pacific Rim nation. They're currently doing a multi-billion-dollar business per year, and their revenue is constrained almost entirely by the rate at which they can unload ships that arrive in the port.

Here's a basic sort of problem they face. A ship arrives, with  $n$  containers of weight  $w_1, w_2, \dots, w_n$ . Standing on the dock is a set of trucks, each of which can hold  $K$  units of weight. (You can assume that  $K$  and each  $w_i$  is an integer.) You can stack multiple containers in each truck, subject to the weight restriction of  $K$ ; the goal is to minimize the number of trucks that are needed in order to carry all the containers. This problem is NP-complete (you don't have to prove this).

A greedy algorithm you might use for this is the following. Start with an empty truck, and begin piling containers 1, 2, 3, ... into it until you get to a container that would overflow the weight limit. Now declare this truck "loaded" and send it off; then continue the process with a fresh truck. This algorithm, by considering trucks one at a time, may not achieve the most efficient way to pack the full set of containers into an available collection of trucks.

- (a) Give an example of a set of weights, and a value of  $K$ , where this algorithm does not use the minimum possible number of trucks.
- (b) Show, however, that the number of trucks used by this algorithm is within a factor of 2 of the minimum possible number, for any set of weights and any value of  $K$ .

**Solution.**

- (a)  $n = 3, (w_1, w_2, w_3) = (3, 5, 2), K = 5$ . This algorithm use 3 trucks, however the minimum possible number of trucks is 2.
- (b) Denote the solution of this algorithm as  $f(w_{\leq n}, K)$ , and denote the optimal solution as  $f^*(w_{\leq n}, K)$ .

We prove  $f(w_{\leq n}, K) \leq 2f^*(w_{\leq n}, K)$  by induction:

- **Base Step:**  $f(w_1, K) = 1 \leq 2f^*(w_1, K) = 2$ .
- **Inductive Step:** We have  $f(w_{\leq n}, K) \leq 2f^*(w_{\leq n}, K)$ , and need to show  $f(w_{\leq n+1}, K) \leq 2f^*(w_{\leq n+1}, K)$ . We claim that  $f(w_{\leq n+1}, K) = f(w_{\leq n}, K) \vee f(w_{\leq n}, K) < 2f^*(w_{\leq n}, K) \vee f^*(w_{\leq n}, K) < f^*(w_{\leq n+1}, K)$ .

**Proof:**

Otherwise,  $f(w_{\leq n+1}, K) = f(w_{\leq n}, K) + 1 \wedge f(w_{\leq n}, K) = 2f^*(w_{\leq n}, K) \wedge f^*(w_{\leq n}, K) = f^*(w_{\leq n+1}, K)$ .

We say  $f^*(w_{\leq n}, K) = t$ . Then we have  $\sum_{i=1}^{n+1} w_i \leq K \times f^*(w_{\leq n+1}, K) = Kt$  (1). Assume in the greedy algorithm, the  $p_i$ -th container is the first one to be added to the  $i$ -th truck. Then  $\forall 1 \leq i \leq 2t, \sum_{j=p_i}^{p_{i+1}} w_j > K$ . Sum over  $i$ , we have  $\sum_{i=1}^{2t} \sum_{j=p_i}^{p_{i+1}} w_j = \sum_{j=1}^{n+1} w_j + \sum_{i=1}^{2t} w_{p_{i+1}} > 2Kt$ . However, by (1),  $\sum_{i=1}^{2t} w_{p_{i+1}} \leq \sum_{j=1}^{n+1} w_j \leq Kt$ . Contradictive! ■

So from the claim above:

- \*  $f(w_{\leq n+1}, K) = f(w_{\leq n}, K) \Rightarrow f(w_{\leq n+1}, K) = f(w_{\leq n}, K) \leq 2f^*(w_{\leq n}, K) \leq 2f^*(w_{\leq n+1}, K)$ .
- \*  $f(w_{\leq n}, K) < 2f^*(w_{\leq n}, K) \Rightarrow f(w_{\leq n+1}, K) \leq f(w_{\leq n}, K) + 1 \leq 2f^*(w_{\leq n}, K) \leq 2f^*(w_{\leq n+1}, K)$ .
- \*  $f^*(w_{\leq n}, K) < f^*(w_{\leq n+1}, K) \Rightarrow f(w_{\leq n+1}, K) \leq f(w_{\leq n}, K) + 1 \leq 2f^*(w_{\leq n}, K) + 1 < 2f^*(w_{\leq n+1}, K)$ .

So by induction we prove that given a fixed  $K$ ,  $\forall n \in \mathbb{N}_+$ ,  $f(w_{\leq n}, K) \leq 2f^*(w_{\leq n}, K)$ .

2. Consider an optimization version of the Hitting Set Problem defined as follows. We are given a set  $A = \{a_1, \dots, a_n\}$  and a collection  $B_1, B_2, \dots, B_m$  of subsets of  $A$ . Also, each element  $a_i \in A$  has a weight  $w_i \geq 0$ . The problem is to find a hitting set  $H \subseteq A$  such that the total weight of the elements in  $H$ , that is  $\sum_{a_i \in H} w_i$ , is as small as possible. (We say that  $H$  is a hitting set if  $H \cap B_i$  is not empty for each  $i$ .) Let  $b = \max_i |B_i|$  denote the maximum size of any of the sets  $B_1, B_2, \dots, B_m$ . Give a polynomial-time approximation algorithm for this problem that finds a hitting set whose total weight is at most  $b$  times the minimum possible.

3. Suppose you're consulting for a biotech company that runs experiments on two expensive high-throughput assay machines, each identical, which we'll label  $M_1$  and  $M_2$ . Each day they have a number of jobs that they need to do, and each job has to be assigned to one of the two machines. The problem they need help on is how to assign the jobs to machines to keep the loads balanced each day. The problem is stated as follows.

There are  $n$  jobs, and each job  $j$  has a required processing time  $t_j$ . They need to partition the jobs into two groups  $A$  and  $B$ , where set  $A$  is assigned to  $M_1$  and set  $B$  to  $M_2$ . The time needed to process all of the jobs on the two machines is  $T_1 = \sum_{j \in A} t_j$  and  $T_2 = \sum_{j \in B} t_j$ . The problem is to have the two machines work roughly for the same amounts of time—that is, to minimize  $|T_1 - T_2|$ .

A previous consultant showed that the problem is NP-hard (by a reduction from Subset Sum). Now they are looking for a good local search algorithm. They propose the following. Start by assigning jobs to the two machines arbitrarily (say jobs  $1, \dots, n/2$  to  $M_1$ , the rest to  $M_2$ ). The local moves are to move a single job from one machine to the other, and we only move jobs if the move decreases the absolute difference in the processing times. You are hired to answer some basic questions about the performance of this algorithm.

- (a) The first question is: How good is the solution obtained? Assume that there is no single job that dominates all the processing time—that is, that  $t_j \leq \frac{1}{2} \sum_{i=1}^n t_i$  for all jobs  $j$ . Prove that for every locally optimal solution, the times the two machines operate are roughly balanced:  $\frac{1}{2}T_1 \leq T_2 \leq 2T_1$ .
- (b) Next you worry about the running time of the algorithm: How often will jobs be moved back and forth between the two machines? You propose the following small modification in the algorithm. If, in a local move, many different jobs can move from one machine to the other, then the algorithm should always move the job  $j$  with maximum  $t_j$ . Prove that, under this variant, each job will move at most once. (Hence the local search terminates in at most  $n$  moves.)
- (c) Finally, they wonder if they should work on better algorithms. Give an example in which the local search algorithm above will not lead to an optimal solution.

Remark: You need to upload your .pdf file.