CSE 102 Spring 2021
Advanced Homework Assignment 7

Jaden Liu
University of California at Santa Cruz
Santa Cruz, CA 95064 USA

May 29, 2021

1 AdvHW7

3. Suppose you are given a set of positive integers $A = \{a_1, a_2, \ldots, a_n\}$ and a positive integer B. A subset $S \subseteq A$ is called *feasible* if the sum of the numbers in S does not exceed B:

$$\sum_{a_i \in S} a_i \leq B.$$

The sum of the numbers in S will be called the *total sum* of S.

You would like to select a feasible subset S of A whose total sum is as large as possible.

Example. If $A = \{8, 2, 4\}$ and $B = 11$, then the optimal solution is the subset $S = \{8, 2\}$.

(a) Here is an algorithm for this problem.

```
Initially S = φ
Define T = 0
For i = 1, 2, ..., n

  If T + a_i ≤ B then
    S ← S ∪ {a_i}
    T ← T + a_i
  Endif
Endfor
```

Give an instance in which the total sum of the set S returned by this algorithm is less than half the total sum of some other feasible subset of A.

(b) Give a polynomial-time approximation algorithm for this problem with the following guarantee: It returns a feasible set $S \subseteq A$ whose total sum is at least half as large as the maximum total sum of any feasible set $S' \subseteq A$. Your algorithm should have a running time of at most $O(n \log n)$.

Solution for a. Assume we have A = {2,4,10} and B = 15, which is similar to the above problem. Then we first pick 2, then 4, but we cannot pick 8 since it's overloaded. Then we have our solution S = {2,4}, which is smaller than the optimal solution S_{opt} = {10,4}, and $S_{sum} = 6 < \frac{1}{2} S_{optsum} = \frac{1}{2} * 14$. □

Algorithm. 1. Order the inputs set A by descending value;
2. Put the next-largest input into the subset, as long as it won't over limit.
The time complexity for the first step(using quick sort or merge sort) is O(nlogn). The second step is O(n) since we need to add every input and compare them. Thus the total running time is at most O(nlogn).

□

Proof for 2 approximation. When this algorithm terminate, there are two situation:

1. We have put every element in A to the S, then obviously it has been the optimal solution.
2. We have an input element that cannot fit in the set S, hence we terminated.

For the second situation, we name the first such input as a_i. Then we have $a_1+a_2+\cdots+a_{i-1}+a_i > S_{optsum}$. Since A is in descending order, then $a_1+a_2+\cdots+a_{i-1} > a_i$, $2(a_1+a_2+\cdots+a_{i-1}) > a_1+a_2+\cdots+a_{i-1}+a_i > S_{optsum}$. Thus, we proved that $a_1+a_2+\cdots+a_{i-1} > \frac{1}{2}S_{optsum}$, which means our greedy solution will be at least half as large as the optimal maximum total sum. □