



Algorithm Theory

Exercise Sheet 14

Due: Friday, 20th of February, 2026, 10:00 am

Exercise 1: Set Sums

(8 Bonus Points)

Suppose you are given a list of N positive integers $L = [a_1, a_2, \dots, a_N]$, and a positive integer C . The problem is to find a subset $S \subseteq \{1, 2, \dots, N\}$ such that

$$T(S) = \sum_{i \in S} a_i \leq C,$$

and $T(S)$ is as large as possible.

- (a) Someone proposes the following greedy algorithm for obtaining a 2-approximate solution to this maximization problem:

Algorithm 1 Greedy Algorithm for Bounded Set Sum

Require: List of integers $[a_1, \dots, a_N]$, capacity C

Ensure: A subset S such that $T(S)$ is maximized under the constraint $T(S) \leq C$

```
1:  $S \leftarrow \emptyset, T \leftarrow 0$ 
2: for  $i = 1$  to  $N$  do
3:   if  $T + a_i \leq C$  then
4:      $S \leftarrow S \cup \{i\}$ 
5:      $T \leftarrow T + a_i$ 
6: return  $S$ 
```

Show that the algorithm is not a 2-approximation algorithm.

(2 Points)

- (b) Modify the greedy algorithm to obtain a 2-approximation algorithm for this maximization problem.
(6 Points)

Exercise 2: Cover Many Sets

(6 Bonus Points)

We consider the following variant of the set cover problem discussed in the lecture. We are given a set of elements X and a collection $\mathcal{S} \subseteq 2^X$ of subsets of X such that $\bigcup_{S \in \mathcal{S}} S = X$.

Instead of finding a collection $\mathcal{C} \subseteq \mathcal{S}$ of the sets which covers all elements, the goal is to find **at most** 2 sets $S_1, S_2 \in \mathcal{S}$ such that the number of covered elements $|S_1 \cup S_2|$ is maximized.

We consider the greedy set cover algorithm from the lecture, but we stop the algorithm after adding 2 sets.

Show that the described greedy algorithm has approximation ratio at most $4/3$.

(6 Points)

Exercise 3: Greedy Knapsack

(6 Bonus Points)

In the lecture, we have considered the Knapsack problem: There are n items with positive weights w_1, \dots, w_n and values v_1, \dots, v_n and a knapsack (a bag) of capacity W such that $w_i \leq W$ for all $1 \leq i \leq n$. A feasible solution to the problem is a subset of the items such that their total weight does not exceed W . The objective is to find a feasible solution of maximum possible total value.

Consider the following greedy algorithm:

1. Sort the n items such that $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \dots \geq \frac{v_n}{w_n}$.
2. Fill the knapsack sequentially with items in the above sorted order starting with the item with largest value per weight. The algorithm stops either if there are no more items left or it reaches an item $k \leq n$ which does not fit, i.e., $w_k > W - \sum_{i=1}^{k-1} w_i$.

- (a) Show that the solution of the greedy algorithm can be arbitrarily bad compared to an optimal solution. (2 Points)
- (b) Let S_g be the set computed by the above greedy algorithm and let S_s be the set containing only one element, i.e., the element with the largest value that fits into the knapsack. Now let our modified algorithm return S_g if $v(S_g) > v(S_s)$ and return S_s otherwise. Show that this modified algorithm computes a 2-approximation. (4 Points)

Note: As usual $v(S)$ denotes the sum over the values of the items in S .

Hint: You can use that the following variant of the above greedy algorithm is optimal for the fractional knapsack problem: The knapsack is sequentially filled with items in the above (greedy) sorted order starting with the item with largest value per weight. When reaching an item $k \leq n$ which does not fit, we add a fraction of the item so that the knapsack is filled to its capacity W .