Warm-up

Problem 1. Construct the Huffman tree of the following phrase: "data structure"

Problem 2. During the lecture we saw that the Fractional Knapsack algorithm's correctness depends on which greedy choice we make. We saw that there are instances where always picking the "highest benefit" or always picking the "smallest weight" doesn't give the optimal solution.

For each of these two strategies, give an infinite class of instances where the algorithm gives the optimal solution, thus showing that *you can't show correctness of an algorithm by using a finite number of example instances*.

Problem solving

Problem 3. Given vectors $a, b \in R^n$, consider the problem of finding a permutation a' and b' of a and b respectively minimizing their total difference

$$\sum_{1 \le i \le n} |a'_i - b'_i|.$$

Prove or disprove the correctness (i.e., that it always returns the optimal solution) of the following algorithm that greedily tries to pair up similar coordinates.

```
1:  function GREEDY-MATCHING-V1(a, b)
2:      A ← multiset of values in a
3:      B ← multiset of values in b
4:      a' ← new empty list
5:      b' ← new empty list
6:      while A is not empty do
7:          x, y ← a pair in (A, B) minimizing |x − y|
8:          Append x to a'
9:          Append y to b'
10:         Remove x from A and y from B
11:     return (a', b')
```

Problem 4. Given vectors $a, b \in R^n$, consider the problem of finding a permutation a' and b' of a and b respectively that minimizes

$$\sum_{1 \le i \le n} |a'_i - b'_i|.$$

Prove or disprove the correctness (i.e., that it always returns the optimal solution) of the following algorithm that greedily tries to pair up similar coordinates.

```
1: function GREEDY-MATCHING-V2(a, b)
2:     A ← multiset of values in a
3:     B ← multiset of values in b
4:     a' ← new empty list
5:     b' ← new empty list
6:     while A is not empty do
7:         x ← smallest in A
8:         y ← smallest in B
9:         Append x to a'
10:        Append y to b'
11:        Remove x from A
12:        Remove y from B
13:    return (a', b')
```

Problem 5. Suppose we are to construct a Huffman tree for a string over an alphabet $C = c_1, c_2, \ldots, c_k$ with frequencies $f_i = 2^i$. Prove that every internal node in T has an external-node child.

Problem 6. Design a greedy algorithm for the following problem (see Figure 1): Given a set of n points $\{x_1, ..., x_n\}$ on the real line, determine the smallest set of unit-length intervals that contains all points.

Figure 1: Covering points with unit intervals.

Problem 7. Suppose we are to schedule print jobs on a printer. Each job j has an associated weight $w_j > 0$ (representing how important the job is) and a processing time t_j (representing how long the job takes). A schedule σ is an ordering of the jobs that tells the printer in which order to process the jobs. Let C_j^σ be the completion time of job j under the schedule σ.

Design a greedy algorithm that computes a schedule σ minimizing the sum of weighted completion times, that is, minimizing $\sum_j w_j C_j^\sigma$.

Advanced problem solving

Problem 8. Show that greedy routing can still get stuck even if it remembers all previously visited vertices and ignores those when determining the next vertex on the path.

Problem 9. Show that compass routing doesn't always reach the destination in general graphs. For an extra challenge, try to construct an example where it cycles through a set of vertices larger than 3.