Institut for Matematik og Datalogi
Syddansk Universitet, Odense

01. May 2025

Exercise Sheet Week 11

DSK814

A. Solve in the practice sessions

1. Specify a Huffman tree for a string with the following alphabet and associated frequencies.

Character	a	b	c	d	e	f	g
Frequency	300	150	75	125	200	50	100

2. Cormen et al., 4. Edition: Excercise 15.1-4 (page 425) [Cormen et al., 3. Edition: Excercise 16.1-4 (page 422)].

   You are given a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. More precisely, each lecture has a start time s and end time f and thus covers the time interval $[s, f)$. You wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall.

   Hint: Let a_t be the number of activities in progress at time t, more formally let $a_t = |\{i | s_i \leq t < f_i\}|$ using the notation from page 418 [page 415]. Let t' be a point in time for which a_t is maximal. Argue that $a_{t'}$ is a lower bound on the number of rooms that should be used. Then find a simple algorithm that makes one pass from left to right and makes greedy/obvious choices.

3. Cormen et al., 4. Edition: Excercise 15.1-3 (page 425) [Cormen et al., 3. Edition: Excercise 16.1-3 (page 422)].

The greedy algorithm for the activity-selection problem produces a maximum-size set of mutually compatible activities, as the "greedy choice" is the activity that ends last. Other natural suggestions for a greedy choice could be:

(a) The activity that is shortest.

(b) The activity that overlaps with the fewest other (remaining) activities.

(c) The activity that starts first.

For each of the other suggestions above, provide a counterexample (i.e. a collection of activities). More precisely, show that the suggestion for a greedy choice will not always find an optimal solution and therefore, does not constitute a correct greedy algorithm.

4. Cormen et al., 4. Edition: Excercise 14.1-2 (page 372) [Cormen et al., 3. Edition: Excercise 15.1-2 (page 370)].

   Consider the following greedy algorithm for the gold chain problem (the root cutting problem in the book): Set the *value density* of a piece of gold chain of length i to be p_i/i, i.e. the value per length. To find a cut of a gold chain of length n, choose the length i, $1 \leq i \leq n$, which has the highest value density (a "greedy" choice). Then continue applying the greedy strategy to the rest of the gold chain of length $n-i$.

   Provide a counterexample that shows that this algorithm will not always find an optimal solution, i.e. a gold chain length n and a collection prices p_i. Therefore, it is not a correct algorithm.

5. Cormen et al., 4. Edition: Excercise 15.2-3 (page 430) [Cormen et al., 3. Edition: Excercise 16.2-3 (page 427)].

   Suppose that in a 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem, and argue that your algorithm is correct.

6. ($*$) Cormen et al., 4. Edition: Excercise 15.2-5 (page 431) [Cormen et al., 3. Edition: Excercise 16.2-5 (page 428)].

   Describe an efficient algorithm that, given a set $S = \{x_1, x_2, \ldots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct and running time.

Hint: the algorithm is simple and similar to the one that solves the activity selection problem. To prove correctness of the algorithm, use the following invariant: the intervals selected so far are a subset of an optimal coverage.

B. Solve at home before tutorial in week 12

The last of the tasks below is the warm-up for the project part III.

1. Consider the alphabet with the seven characters a,b,c,d,e,f,g. The table below shows how often each character appears in a given text.

Character	a	b	c	d	e	f	g
Frequency	9	7	24	10	55	15	25

   Draw a Huffman-tree representing the Huffman codes for this example.

2. Cormen et al., 4. Edition: Excercise 15.3-3 (page 439) [Cormen et al., 3. Edition: Excercise 16.3-3 (page 436)]:

   What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

Character	a	b	c	d	e	f	g	h
Frequency	1	1	2	3	5	8	13	21

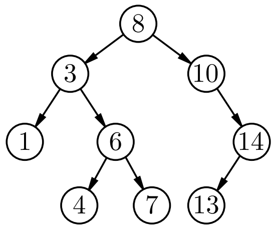
   Find a Huffman tree for the above input. Then argue for what the Huffman tree looks like for such inputs of arbitrary size n.

3. Cormen et al., 4. Edition: Excercise 15.3-7 (page 439) [Cormen et al., 3. Edition: Excercise 16.3-8 (page 436)]:

   For $n = 256$, look at the input to Huffman's algorithm where frequencies don't differ by more than a factor of two from each other. More precisely, the largest frequency is less than twice as large as the smallest frequency. What does the Huffman tree look like for such input?

   Hint: Can you say something about the magnitude of the frequencies of the subtrees during the first 128 merge steps (which turns the 256 original trees (just leaves) to become 128 trees)? And again during the next 64 merge steps?

4. Warm-up for project part III: In your `DictBinTree` from part II of the project, add a method that prints a description of the paths to all nodes in the tree. For the following tree



the output should be

```
Key 1: LL
Key 3: L
Key 4: LRL
Key 6: LR
Key 7: LRR
Key 8:
Key 10: R
Key 13: RRL
Key 14: RR
```

Hint: adjust the code for the inorder pass-through appropriately. Test the method on a tree built either directly in the code, or via some calls to `insert`.