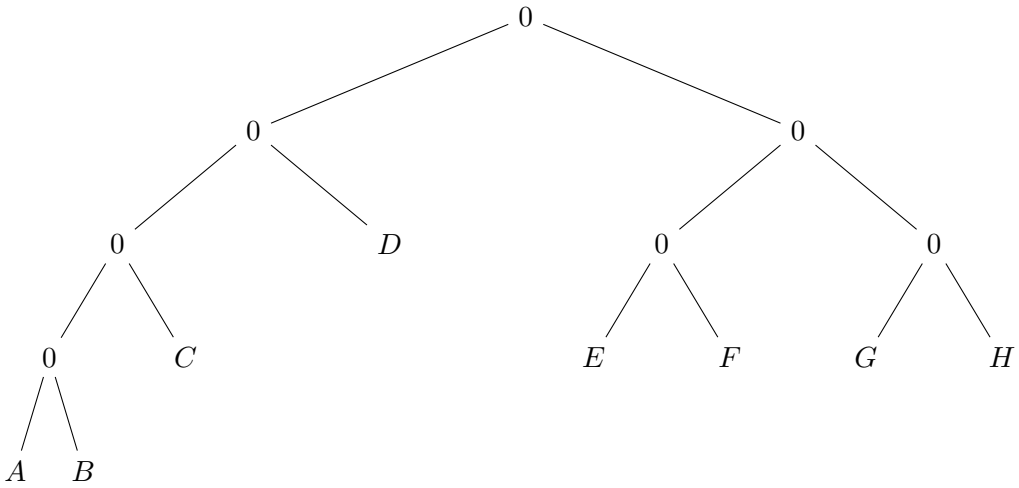
Tutorial 3, Design and Analysis of Algorithms, 2021

1. Solve the following instance of the *Fractional Knapsack Problem*, by applying the *Greedy Algorithm*:
   There are 8 items with profit of the items given by $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) = (14, 9, 11, 8, 13, 12, 4, 10)$, weight of the items given by
   $(w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8) = (3, 2, 4, 8, 5, 9, 7, 6)$, and the knapsack capacity given by $W = 16$.

2. (a) Find the prefix code corresponding to the following binary tree:



   (b) Draw the binary tree corresponding to the following prefix code:
   $A = 00, B = 0100, C = 0101, D = 011, E = 100, F = 1010, G = 1011, H = 11$.

   (c) Using Huffman's algorithm find the optimal prefix code for the alphabet $\{A, B, C, D, E, F, G, H\}$ for the following frequencies:
   $f_A = \frac{3}{40}, f_B = \frac{6}{40}, f_C = \frac{4}{40}, f_D = \frac{10}{40}, f_E = \frac{9}{40}, f_F = \frac{5}{40}, f_G = \frac{1}{40}, f_H = \frac{2}{40}$.

   (d) Find *Average Bit Length* of the optimal prefix code in 2(c).

3. Using Huffman's algorithm find the optimal prefix code and *Average Bit Length* of the optimal prefix code for the alphabet
   $\{ x_j \}_{j=1}^{n}$ for the following frequencies:
   $f_{x_j} = \frac{1}{2^j}, \quad \forall j \in [1..n-1]$,
   $f_{x_n} = \frac{1}{2^{n-1}}$.

4. Describe an efficient algorithm that, given a set $\{ x_1, x_2, \ldots, x_n \}$ 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.

5. There exists an $O(n)$-time deterministic algorithm (M) for finding median of n given numbers. Using this algorithm as a subroutine, design an $O(n)$-time deterministic algorithm for solving the fractional knapsack problem (items are $(I_i)_{i=1}^{n}$, weight of items are $(w_i)_{i=1}^{n}$, profit of items are $(p_i)_{i=1}^{n}$, and knapsack capacity is W), and also prove its time complexity.

6. Show that no compression scheme can expect to compress a file of randomly chosen 8-bit characters by even a single bit.

7. Suppose we have an optimal prefix code on a set $C = \{ 0, 1, \ldots, n-1 \}$ of characters and we wish to transmit this code using as few bits as possible. Show how to represent any optimal prefix code on C using only $2n - 1 + n\lceil \log n \rceil$ bits. Make a binary code for the optimal prefix code of problem $2(a)$.

8. Suppose that a data file contains a sequence of 8-bit characters such that all 256 characters are about equally common: the maximum character frequency is less than twice the minimum character frequency. Prove that Huffman coding in this case is no more efficient than using an ordinary 8-bit fixed-length code.

9. Let G be a directed graph with n vertices. Let $l(u, v)$ be the (non-negative) length of the directed edge $u \to u$. A path starting at a given vertex v_0, going through every other vertex exactly once, and finally returning to v_0 is called a *tour*. The length of a tour is the sum of the lengths of the edges on the path defining the tour. We are concerned with finding a tour of minimum length. A greedy way to construct such a tour is: let (P, v) represent the path so far constructed; it starts at v_0 and ends at v. Initially P is empty and $v = v_0$, if all vertices in G are on P, then include the edge $v \to v_0$ and stop; otherwise include an edge $v \to w$ of minimum length among all edges from v to a vertex w not on P. Show that this greedy method does not necessarily generate a minimum-length tour.

10. Consider the following scheduling problem: n jobs are given as input. Job j ($1 \leq j \leq n$) has a processing time p_j ($p_j > 0$) and a non-negative weight w_j ($w_j \geq 0$). We must construct a schedule for these jobs on a single machine such that at most one job is processed at each point in time, and each job must be processed nonpreemptively; that is, once a job begins to be processed, it must be processed completely before any other job begins its processing. The objective is to find a schedule that minimizes the weighted sum of completion times: $\sum_{j=1}^{n} w_j C_j$. Suppose that jobs are indexed such that $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \ldots \geq \frac{w_j}{p_j} \geq \ldots \geq \frac{w_n}{p_n}$. Then prove that it is optimal to schedule the jobs in the order $(1, 2, \ldots, j, \ldots, n)$ (job 1 first, job 2 second, and so on).