CentraleSupélec

1re année 2018–2019
ST2, campus de Saclay
Vendredi 25 janvier 2018, matin

Example of a Written Exam of

Algorithmics and Complexity

Duration: 3 hours

Student number:	
First name:	Family name:

— All documents are allowed.
— Calculators, computers and phones are not allowed.
— All your algorithms can be written in pseudo-code or in Python.
— The points scale is given on an indicative basis
— The examination is 6 pages long.
— You can use the extra pages at the end if you need more space.

This is a fabricated exam that was composed from the last 3 years exams at Centrale and Supélec/Gif

It aims to give an idea of the type of subject that will be proposed in 2018-2019.

Exercise 1 : Huffman's algorithm 5 point(s)

We are interested in lossless compression of character sequences. In computing, each character is represented using n bits according to an encoding table, the most well-known being the ASCII table, on 7 bits. For example, symbol "a" is coded `1100001` in ASCII.

Huffman's idea to compress a sequence of characters, is to encode the characters on a variable number of bits depending on the frequency of occurrence of the character. For example, the most frequent letter in French is the letter "e" : it should thus be coded with as few bits as possible, unlike the infrequent "w", which can be coded with more.

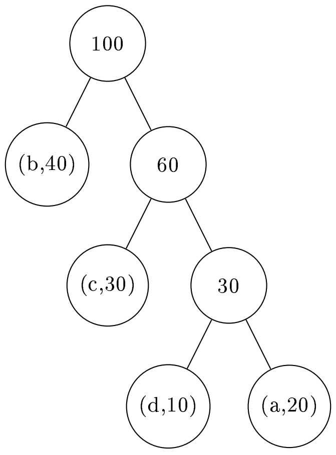
Huffman's compression algorithm is based on the use of a compression binary tree to calculate the binary code of each character. The construction of the tree is as follows :

1. Calculate the frequency of appearance of each character and build a list of pairs (c, f_c) where c is a character and f_c its frequency of occurrence ;
2. Sort the list in terms of increasing f_c ;
3. As long as the list is not reduced to a single item :
   (a) Remove the two smallest elements c_1 and c_2 from the list ;
   (b) Build a binary tree a of height 1 whose two leaves are c_1 and c_2 and whose root is labeled by the sum of the frequencies of these elements : $f_a = f_{c_1} + f_{c_2}$;
   (c) Insert (a, f_a) in the list (maintaining the sorting of $f\_$ values).

Let's imagine, for example, the following frequencies of occurrence :

character	a	b	c	d
frequency	20%	40%	30%	10%

The algorithm will first take the characters "a" and "d" in order to group them together. It will get a "30%" frequency tree and put in the list. In the next round, it will assemble this tree with the character "c" to make a frequency tree of "60%", etc. The compression tree obtained will be the following :



The binary code of each character is then defined by its path in the compression tree, putting a 0 when you take the left branch and a 1 when you take the right branch. In our example, character "b" is coded as "0" while character "a", less frequent, is coded as "111".

Question 1 1 point(s)

Suggest a data structure to represent the data in the built coding tree. You are required to specify precisely how the structure can be used.

Decoding

Question 2 1 point(s)

To begin, we will decode a single character. Write, in pseudo-code or Python, a decompression function that takes as input a list of bits and a compression tree and that returns the character represented by said list of bits, or None if the list of bits is not a valid Huffman code for that compression tree.

Question 3 1 point(s)

Based on the previous code, write, in pseudo-code or Python, a decompression function that takes as an input a list of bits representing a list of several characters and a compression tree, and returns the initial list of characters.

Question 4 0.5 point(s)

What is the complexity of this algorithm ?

Encoding

Question 5 1.5 point(s)

Write a function that takes a compression tree as input and returns a data structure to associate each character with its Huffman code.

Exercise 2 : Safest Route Problem **5 point(s)**

Our objective is to calculate the safest route for cash transport from point S to point T. Think of graph-oriented modeling the road network, where the nodes are the crossroads and the arcs represent the roads. An arc $A \xrightarrow{p} B$ represents a road that allows traffic to flow from junction A to junction B with a risk of steering corresponding to a probability $0 < p < 1$.

Question 1 1 point(s)

What is the probability that a road will be used without encountering a turn based on the probabilities of turning carried by its arcs ?

Question 2 3 point(s)

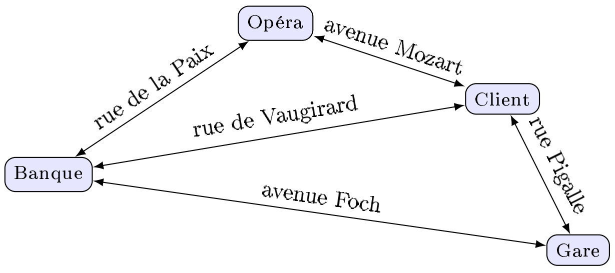
Determine and then adapt the most effective algorithm (seen in class) to solve the problem of the safest path.

Provide the complexity according to the inputs of the problem.

Question 3 1 point(s)

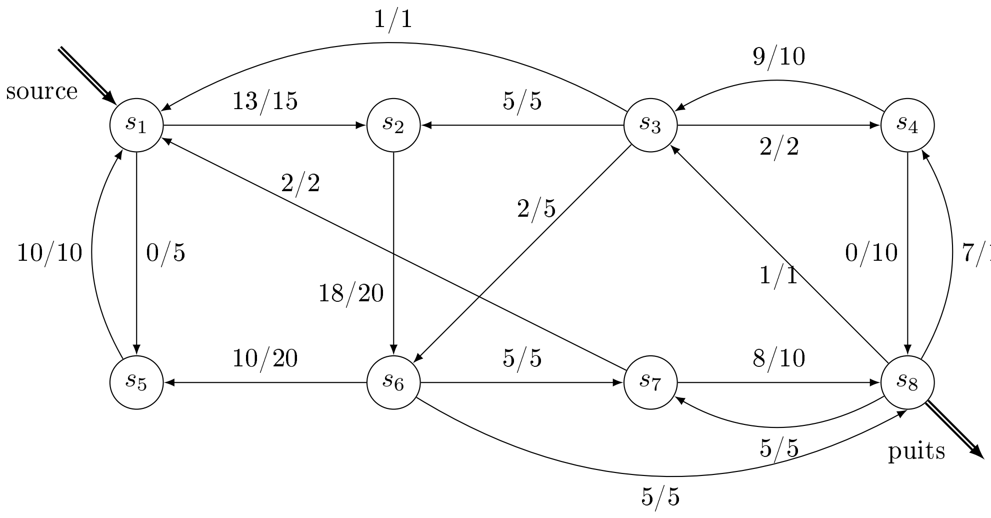
Unroll this algorithm on the next instance by describing the intermediate steps (you have to go from the bank to the customer). The arcs can be used in both directions with the same probability of steering.

— Rue de la Paix : $p = 0.1$
— Rue de Vaugirard : $p = 0.1$
— Avenue Mozart : $p = 0.05$
— Rue Pigalle : $p = 0.05$
— Avenue Foch : $p = 0.01$



Exercise 3 : Flow problem **3 point(s)**

The following graph is initialized with a non-zero flow, the label $\varphi(a)/c(a)$ on the arc a indicating that a flow $\varphi(a)$ is flowing on the arc a which has the capacity $c(a)$.



Complete the execution of the Ford-Fulkerson algorithm and give the value of a maximum flow by showing the intermediate steps performed. Give a minimum capacity cut.

Exercise 4 : Dynamic Programming — 4 point(s)

Let A be a matrix $n \times m$ of integers. We are looking for the size of a square submatrix $k \times k$ of A composed exclusively of zeros whose size is the largest.

Example : For an array

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

an algorithm must give answer 2.

Question 1 — 1.5 point(s)

Suggest a recursion formula.

Question 2 — 1.5 point(s)

Use your formula to propose a resolution algorithm by dynamic programming.

Question 3 — 0.5 point(s)

Calculate the complexity in time.

Question 4 — 0.5 point(s)

State the size of a structure in which you store intermediate results.

Exercise 5 : Dealing with Hard Problems — 4 point(s)

The problem of Maximum Clique in a graph $G = (V, E)$ is to find a subset of V' vertices of maximum size such that the subgraph induced by V' (i.e. a subgraph $G' = (V', E')$ such as $E' = E \cap V' \times V'$) is complete.

The associated decision problem, determine if there is a Clique of at least k size, is an *NP*-complete problem.

Question 1 — 1.5 point(s)

Suggest a greedy algorithm to build a subgraph that is a Clique.

Question 2 0.5 point(s)

Calculate the complexity of your algorithm.

Question 3 0.5 point(s)

Is the Clique returned by your algorithm maximum ? Justify your response.

Question 4 1.5 point(s)

The problem of Clique coverage is as follows : given a graph $G = (V, E)$ and an integer K, is there a partition of V into at most K subsets V_i such as the subgraphs induced by each V_i (i.e. $G_i = (V_i, E_i)$ such that $E_i = E \cap V_i \times V_i$) are Cliques.

A graph can be colored with, at most, K colors if each vertex of the graph can be associated with a color so that a vertex never shares a color with one of his neighbors.

Knowing that determining if a graph $G = (V, E)$ is colorable in K colors is a NP-Complete problem, show that the problem of Clique coverage is also NP-Complete.

— end —

2019-01-17