CentraleSupélec

First year 2018–2019
ST2, campus of Saclay

Friday 25^{th} January 2018, morning

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 12 pages long.
— You can use the extra pages at the end if you need more space.
— Exercices are independent from each other.

Exercise 1 : Friends of my friends — 3 point(s)

Let us consider a social network that stores for each user its list of friends. We assume that there is a function `friends(u)` that returns the list of friends of user `u`.

We want to write an algorithm that makes contact suggestions from friends of our friends.

Question 1 — 2 point(s)

Write, in pseudo-code or Python, a function `propose(u)` taking as input a user `u` and suggesting **another** user who :

1. Is not already on `u`'s list of friends ;
2. Appears most frequently in the lists of friends of `u`'s friends.

Question 2 1 point(s)

For each user we want to know all the users who are either direct friends or for whom there is a sequence of friends who can act as intermediaries. In other words, we want to build the *transitive closure* of the graph of our social network.

Write an algorithm that does this construction.

Exercise 2 : Vertex Cover — 4 point(s)

RATP would like to install cameras in the Châtelet metro station so that it can monitor the movements of passengers within the station. The cameras will be installed at the intersections of the corridors : they make it possible to identify any person passing through one of the corridors connected to this intersection.

The objective is to allow for monitoring all the station's corridors. For cost reasons, we want to place as few cameras as possible.

Question 1 — 1.5 point(s)

Model this problem. What is the nature of the problem ? What are the entries and what is the question to ask ?

We consider the following algorithm that computes a solution to the problem at hand. It takes the list of *corridors* as a parameter whose elements are pairs of intersections $(i1, i2)$. Each corridor is identified by the two points of the station that it connects to each other.

```
def compute_solution(corridors)

R = corridors.copy()
S = []

while len(R)>0:
    (i1,i2) = R.pop()

    S.append(i1)
    S.append(i2)

    R2 = []
    for r in R:
        if r[0]!=i1 and r[1]!=i1 and r[0]!=i2 and r[1]!=i2:
            R2.append(r)

    R = R2

return S
```

Question 2 — 1.5 point(s)

What does this algorithm do ? Explain how it works in English using terms from your own modeling. Specify to which family of algorithms it belongs. Calculate its complexity in time. Justify each of your answers.

Question 3 1 point(s)

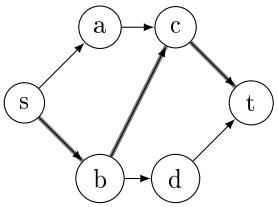
Demonstrate that this algorithm is a 2-approximation, i.e. it never yields a solution S that is larger than twice the size of optimal solutions.

Exercise 3 : Paths Search **3 point(s)**

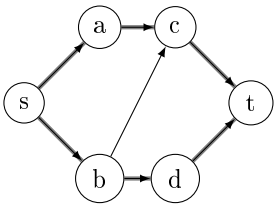
We are interested in the following problem :

> **MAX-EDGE-DISJOINT-PATHS**
> **Inputs** : A directed graph $G = (V, E)$ and two vertices s and t of G.
> **Question** : Build as many paths as possible from s to t without using the same arc twice.

Each arc can therefore **only be used in one path**. The goal is to get as many paths as possible. On the example below, it is possible to build a two-path solution (this is the maximum).



Solution of 1 path



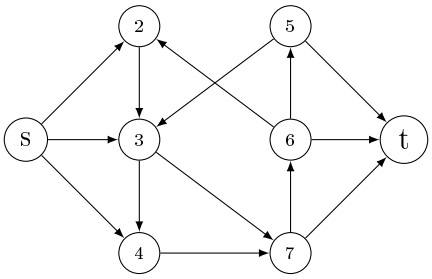
Solution of 2 paths

Question 1 2 point(s)

Propose a polynomial algorithm that computes the maximum number of possible paths. Justify your answer.

Question 2 1 point(s)

Unroll this algorithm on the next problem by detailing all the steps. What is the maximum number of paths ?



Exercise 4 : A « fun » party **5 point(s)**

A company organizes a party event for its employees. The organizers want the evening to be as entertaining as possible.

« Fun » rating. The ability of each employee to joke and be in a good mood is known and listed by the organizers. This is expressed by a « fun » rating assigned to each employee.

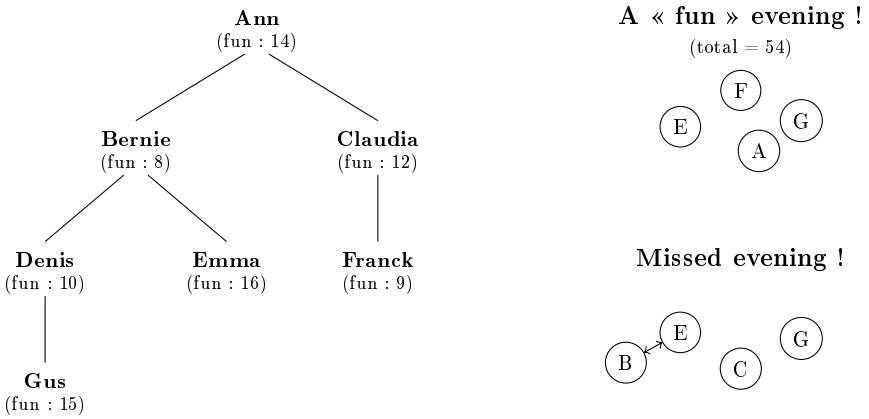


FIGURE 1 – An example of parties in a company of 7 people

Never with my « n+1 ». Each employee has a precise hierarchical position in the company. It is then impossible for an employee and his « n+1 » (i.e. his direct superior) both to be present at the party, because it would not be « fun » at all...

The goal is to organize an evening where the sum of the « fun » ratings of all the participants in the evening is at a maximum, while respecting the constraint « never with my n+1 ». To do this, we will write an algorithm using the principle of **dynamic programming**.

Useful functions

For any given employee i, we note :
- fun(i) the value of its « fun » rating ;
- sub(i) the list of his direct subordinates (those for whom he is the « n+1 »).

We will compute two values :
- $\text{total}_{\text{yes}}(i)$ the maximum « fun » score of an evening limited to employee i and all his direct and indirect subordinates, **when i is present in the evening event** ;
- $\text{total}_{\text{no}}(i)$ the maximum « fun » score of an evening limited to employee i and all his direct and indirect subordinates, when i **is not present** in the evening event.

Question 1

2 point(s)

Write recursion formulas for $\text{total}_{\text{yes}}(i)$ and $\text{total}_{\text{no}}(i)$ based on the $\text{total}_{\text{yes}}$ and total_{no} values of the subordinates.

Question 2 3 point(s)

Write a dynamic programming algorithm that maximizes the sum of the « fun » ratings of the participants of the evening event. You can write one or more functions according to what you think is most appropriate.

Exercise 5 : Problem complexity — 3 point(s)

The following two problems are considered :

BI-PARTITION
Inputs : A set SE of numbers.
Question : Is there a subset $F \subseteq E$ such that $\sum_{x \in F} x = \frac{1}{2} \sum_{x \in E} x$?

SUBSETSUM
Inputs : A set S of numbers and a number t
Question : Is there a subset $T \subseteq S$ such that $\sum_{x \in T} x = t$?

Question 1 — 1 point(s)

Show that SUBSETSUM is in NP.

Question 2 — 2 point(s)

Knowing that BI-PARTITION is NP-complete, show that SUBSETSUM is also NP-complete.

Exercise 6 : Back to hospitals **2 point(s)**

Let's get back to the problem of hospitals coverage that we worked on in our two lab sessions (TP).

Reminders : We have defined a `combination_k` function that can be used as an iterator in a loop to write an exhaustive search algorithm. For example, the code below shows all combinations of 10 cities among potential candidates.

```
def test_combination(candidates):
    for c in combination_k(candidates,10):
        print(c) # prints a list of 10 cities
```

We also wrote an `all_distances` function that computes a dictionary containing all the distances between two cities in the territory. The keys of the dictionary are pairs of cities and the values are the distances between cities :

```
def test_distances(cities):
    dict_dist = all_distances(cities)
    print(dict_dist[('A','B')])
```

The code above prints the distance between the cities with names A and B in the territory (if they exist).

Definition : The *diameter* of a list of cities is the largest distance between two cities in the list.

Question 1 1.5 point(s)

Write a function that takes as parameters : a territory, a list of candidate cities and a number k and returns the combination of k candidate cities having the smallest diameter (or one of these combinations if there exists several).

Question 2 0.5 point(s)

What is the complexity of this algorithm? Justify your answer.

Exercise 7 : Bonus **1 point(s)**

We consider the following problem :

SATISFIABILITY
Inputs :
— a set l of boolean variables ;
— a logic formula $f(l)$ over l.

Question : Is there an assignment of values to l such that $f(l)$ is true ?

Question 1 1 point(s)

Write a polynomial algorithm to solve this problem.

— end —

2019-01-30

Extra page n. 1