- Parmi n personnes, une *célébrité* est définie comme étant une personne qui est connue de tous mais qui ne connaît personne d'autre qu'elle-même. Le but de ce devoir est de concevoir un algorithme efficace de recherche d'une célébrité à partir de questions de la forme : "Excusez-moi, connaissez-vous la personne qui est là-bas ?"
- À un ensemble E de n personnes désignées par $1, 2, \ldots, n$ on associe un graphe $G(E)$ de la façon suivante :
  - chaque personne i est représentée par un nœud du graphe noté i également (il n'y a pas d'autres nœuds) ;
  - (i, j) est un arc du graphe ssi la personne i connaît la personne j.
- La *matrice d'adjacence* du graphe $G(E)$ est la matrice booléenne A, carrée d'ordre n, définie par $A_{i,j} = 1$ si (i, j) est un arc du graphe, et $A_{i,j} = 0$ sinon.
- Par *algorithme*, il faut comprendre (recommandation superflue, mais bon...) : une fonction écrite en langage Caml (en évitant les références) accompagnée d'un commentaire, d'une justification etc.

1° Combien peut-il y avoir de célébrités dans E ?

2° Donner une matrice d'adjacence correspondant à un ensemble de quatre personnes possèdant une célébrité.

3° Donner une matrice d'adjacence correspondant à un ensemble de quatre personnes ne possèdant pas de célébrité.

4° Caractériser une célébrité de E au vu de la matrice d'adjacence A du graphe $G(E)$.

- Pour les calculs de complexité demandés dans la suite, l'unité de coût est la consultation d'un coefficient de la matrice d'adjacence.

5° Donner un algorithme qui permet, au vu de la matrice d'adjacence A de $G(E)$, de tester si une personne i donnée est une célébrité de E. Exprimer son coût en fonction de n.

6° Donner un algorithme qui détermine si E possède une célébrité à partir de la matrice d'adjacence A de $G(E)$. Exprimer son coût en fonction de n.

- On essaie à présent une autre méthode.

7° Indiquer comment déterminer si $E = [\![1, n]\!]$ $(n > 2)$ possède une célébrité connaissant le résultat pour l'ensemble $[\![1, n-1]\!]$. On indiquera précisément le temps de calcul du passage de $n-1$ à n.

8° Quel est le coût dans le pire des cas d'un algorithme qui applique la récurrence de la question **7°** ?

- Pour obtenir un algorithme efficace de recherche de célébrité on peut exploiter les réponses aux questions posées aux personnes comme suit. Supposons qu'on demande à Roméo s'il connaît Juliette. Si sa réponse est "oui", Roméo ne peut être une célébrité. Et dans le cas contraire c'est Juliette qui ne peut être une célébrité. Ainsi, au moyen d'une seule question on réduit le problème à l'examen d'un ensemble possédant une personne de moins. C'est le principe du *duel*.

9° Écrire un algorithme qui permet d'éliminer toutes les personnes de E sauf une en utilisant le principe du duel, à partir de la matrice d'adjacence A de $G(E)$. Quel est son temps de calcul ? Que dire de la personne restante ? Comment conclure ?

10° Citez une célébrité (que vous connaissez) !

Solution proposée

1° Il peut y en avoir zéro (si tout le monde connaît tout le monde, par exemple), une (si $n = 1$, par exemple) mais il ne peut y en avoir deux ou plus : s'il y une célébrité, elle ne connaît personne et il ne peut donc y avoir une deuxième célébrité.

2° On suppose que la célébrité est la personne 1 :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

3° La matrice identité convient, la matrice dont tous les coeffients sont égaux à 1 aussi.

4° i est une célébrité ssi $A_{i,j} = \delta_{ij}$ pour tout $j \in [\![1, n]\!]$ (elle ne connaît personne hormis elle-même) et $A_{j,i} = 1$ pour tout $i \in [\![1, n]\!]$ (tous les autres la connaissent).

5° On vérifie que la ligne i ne contient que des 0 (sauf en position i), et que la colonne i ne contient que des 1.

```
let est_célèbre i a = let n = vect_length a in try
 for j = 1 to n do
  if (i<>j) && (a.(i-1).(j-1) = 1 || (a.(j-1).(i-1) = 0))
  then failwith "pas célèbre"
 done; true
 with _ -> false;;
```

Le coût est au plus de $2n - 2$ (on ne teste pas la diagonale), soit un $O(n)$.

6° On regarde successivement si $1, \ldots, n$ sont des célébrités.

```
let il_y_a_une_célébrité a = let n = vect_length a in try
 for i = 1 to n do
  if est_célèbre i a then failwith "célèbre"
 done; false
 with _ -> true;;
```

Il y a au plus $n(2n - 2)$ test ainsi le coût est $O(n^2)$.

7° Si $[\![1, n-1]\!]$ a une célébrité i, on fait deux tests supplémentaires pour vérifier si i reste célèbre dans $[\![1, n]\!]$.
Si c'est le cas, on a fini.
Sinon , on doit encore tester la célébrité de n : ce qui nécessite $2n - 4$ tests (2 ont déjà été effectués avec i).
Enfin, si $[\![1, n-1]\!]$ n'a pas de célébrité, il ne reste qu'à tester en $2n - 2$ coûts la célébrité de n dans $[\![1, n]\!]$.

8° On dispose ainsi d'une méthode en $O(n)$ pour passer de $[\![1, n-1]\!]$ à $[\![1, n]\!]$. Un algorithme qui itère cette méthode est à nouveau en $O(n^2)$.

9° On itère sur $[\![1, n]\!]$, le duel entre deux personnes :

```
let duel a = let n=vect_length a in
   let rec duel_aux a  i j =
     match (a.(i).(j), j) with
       | (1, j) when j=n-1 -> n
       | (1, _)            -> duel_aux a j (j+1)
       | (0, j) when j=n-1 -> i+1
       | _                 -> duel_aux a i (j+1)
   in duel_aux a 0 1;;
```

Chaque duel élimine une personne. Le gagnant final est ainsi obtenu en $n - 1$ tests. Il reste encore à vérifier s'il s'agit bien d'une célébrité : ce qui coûte $O(n)$. Finalement, on dispose d'un algorithme linéaire pour résoudre notre problème.

10° Vous ! Dans un groupe à une seule personne.
Une autre solution (plus fun !) : soit un groupe de personnes dans lequel on rajoute Brigitte Bardot sur sa moto. C'est une célébrité car tout le monde la connait et "elle ne connait plus personne en Harley Davidson" !