Informatique PT. 2025 - 2026

Mathématiques & Informatique - F. Gaunard
`http://frederic.gaunard.com`
Lycée Voltaire, Paris 11e.

1 Travaux Pratiques : Révisions d'algorithmique

Pour ce premier TP de rentrée, on propose des exercices, principalement issus de l'oral **Math II** qui invitent à manipuler des listes ou des tableaux de manière assez élémentaire.
On rappelle qu'un exercice de ce type doit être, le jour de l'oral, traité en moins de 30 minutes.

Exercice 1.

Donner une fonction récursive qui détermine si un *flottant* appartient à une liste **triée**, en effectuant une recherche par dichotomie.

Exercice 2. *Codage binaire*, Oral Math II 2018

1. On considère (par exemple) les listes `S=[0,1,0,1]` et `X=[2,7,3,9]`. Donner une fonction `coder` telle que (par exemple) `coder(S,X)` renvoie `[7,9]`.
2. Écrire une fonction `decoder` telle que (par exemple) `decoder([2,7,3,9],[7,9])` renvoie `[0,1,0,1]`.
3. Écrire une fonction `incrementer` prenant en entrée le codage binaire d'un entier k (par exemple `[0,1,0,0,1,...]`) et qui renvoie le codage binaire de l'entier $k+1$.

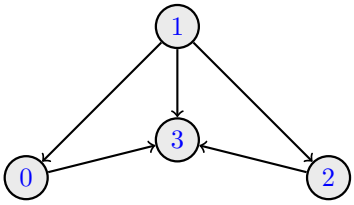
Exercice 3. *Chiffres et chaînes*, Oral Math II 2017

1. Que fait la fonction `list(str(n))` où `n` est de type `int`?
2. Écrire deux fonctions différentes prenant en entrée un entier et renvoyant respectivement le chiffre des dizaines et le chiffre des entiers de celui-ci.
3. Écrire une fonction `transforme(n)` qui parcourt les chiffres composant `n` et qui:
   - ✗ pour les termes de rang impair ne change rien;
   - ✗ pour les termes t de rang pair:
     - si le double de t est inférieur à 10, alors on remplace t par son double;
     - si le double de t est supérieur ou égal à 10, on remplace t par la somme du chiffre des dizaines et du chiffre des unités de $2t$.

   Par exemple 3 devient 6, 7 devient 5, et 32814 devient 62718.
4. Déterminer les nombres inférieurs à 100 000 invariants par la fonction `transforme`.

Exercice 4. *Graphes et sources universelles*, Oral Math II 2024

Dans un graphe orienté $G = (S, A)$ sans boucles, une *source universelle* est un sommet u tel que, pour tout $v \in S\backslash\{u\}$, on a $(u, v) \in A$ et $(v, u) \notin A$. Le but de cet exercice est la détection de sources universelles, selon la représentation du graphe.



Par exemple, le sommet  est une source universelle dans le graphe ci-dessus.

1. Peut-il exister plusieurs sources universelles?

2. Écrire une fonction `source_mat` qui prend en entrée un graphe représenté par sa matrice d'adjacence et renvoie `True` si le graphe contient une source universelle, et `False` sinon.

3. Discuter de la complexité de votre fonction.

4. Écrire une fonction `source_list` qui prend en entrée un graphe représenté par listes d'adjacence et renvoie `True` si le graphe contient une source universelle, et `False` sinon.

5. Discuter de la complexité de votre fonction.

Un graphe orienté est un *océan* s'il contient une source universelle v et que $G\backslash\{v\}$ est encore un océan. Ici $G\backslash\{v\}$ est le graphe obtenu à partir de G en supprimant le sommet v ainsi que tous les arcs qui lui sont incidents.

6. Écrire une fonction `supprime` qui prend en entrée un graphe G, représenté de la manière la plus appropriée, et un sommet v de G, et qui renvoie le graphe $G\backslash\{v\}$.

7. Écrire une fonction `ocean` qui prend en entrée un graphe, représenté de la manière la plus appropriée, et qui renvoie `True` si le graphe est un océan, et `False` sinon.

8. Combien de graphes orientés sont des océans?
Proposer une autre méthode pour tester si un graphe est un océan.

Exercice 5. *Tri*, Oral Math II 2024

1. Écrire une fonction `parcours` d'argument une liste $L = [a_0, a_1, \ldots, a_{n-1}]$ qui modifie cette liste en permutant a_k et a_{k+1} si $a_{k+1} < a_k$, pour k variant de 0 à $n-2$, et qui renvoie le nombre de permutations effectuées.
Par exemple, si `L = [5, 4, 1, 6]`, alors `parcours(L)` renvoie le nombre 2 et la liste `L` devient `[4, 1, 5, 6]`.

2. Que peut-on dire du dernier élément d'une liste à laquelle on a déjà appliqué une fois la fonction `parcours` ?

3. On veut utiliser cette fonction pour trier par ordre croissant une liste selon le principe suivant : pour une liste L donnée en entrée on répète l'application de la fonction parcours jusqu'à ce que `L` devienne invariante par cette fonction.
Écrire une fonction `tri` qui ordonne une liste donnée en entrée selon ce principe et qui renvoie le nombre d'itérations effectuées.

4. Expliquer pourquoi la fonction `tri` donne bien le résultat voulu en un temps fini.
Évaluer son coût de calcul dans le meilleur des cas et dans le pire des cas.

5. Écrire une fonction `tri1` comme une amélioration de la fonction `tri` en évitant les comparaisons inutiles.

Exercice 6. *Nombres de type $\mathcal{T}$*, Math II 2018

Pour $n \in \mathbb{N}$, on note $t_n = 1 + 2 + 3 + \cdots + n = \sum_{k=1}^{n} k$. Un entier j est dit de type $\mathcal{T}$ si $j \in \{t_n : n \in \mathbb{N}\}$.

1. Exprimer t_n en fonction de n.
2. Écrire un code donnant la liste de tous les entiers de type $\mathcal{T}$ inférieurs à 100.
3. Écrire une fonction somme avec comme arguments deux listes L_1,L_2 et un entier n, qui renvoie la liste des entiers, inférieurs à n, pouvant s'écrire comme la somme d'un élément de L_1 et d'un élément de L_2, liste rangée dans l'ordre croissant.
4. Donner la commande qui en utilisant la fonction précédente, permet de donner la liste des entiers inférieurs à 100 pouvant s'écrire comme la somme de trois entiers de type $\mathcal{T}$ inférieurs à 100. Que peut-on conjecturer?
5. Écrire une fonction `nbdecomb` qui prend en entrée un entier n et qui renvoie le nombre de combinaisons possibles de 3 entiers de type $\mathcal{T}$ dont la somme donne n.
6. En déduire le premier entier N qui est la somme de 10 combinaisons de 3 entiers de type $\mathcal{T}$.
7. Estimer le temps de calcul de la fonction `nbdecomb`.

Exercice 7. *Mots cycliques*, Oral Math II 2022

On considère ici des mots écrits uniquement avec les symboles 0 et 1, qui ne contiennent pas le sous-mot 101. On note E l'ensemble des mots ayant cette propriété, et E_n l'ensemble des mots de longueur n ayant cette propriété.
On modélise en Python un mot écrit avec les symboles 0 et 1 par une chaîne de caractères.

1. Écrire une fonction `suivant` prenant en entrée un mot `m` de E_n, et renvoyant la liste des mots de E_{n+1} commençant par `m`.
   Par exemple, `suivant('01')` renverra `['010','011']` et `suivant('10')` renverra `['100']`.

2. Écrire une fonction `mots` prenant en entrée un entier `n`, et renvoyant la liste des mots de E_n.

On dit qu'un mot est cyclique s'il fini par `'10'` et commence par `'1'`, ou s'il fini par `'1'` et commence par `'01'`.

3. Écrire une fonction `nbc` prenant en entrée un entier `n`, et renvoie le nombre de mots cycliques dans E_n.
   Par exemple, `nbc(3)` renverra 2.