

TP05 CONCEPTION D'ALGORITHMES SIMPLES

Comme ce sera le cas durant toute l'année, votre dossier de travail est accessible¹ sur votre session dans Mes Documents/Devoirs/fleck/TP05/.

Partie I

À préparer à la maison sans ordinateur

1 Comparaison d'algorithmes

Expliquer en quoi la procédure `nb_div(n)` peut dans certains cas être bien plus rapide que `nb_div_naif(n)`.

```
1 def nb_div_naif(n):
2     '''Renvoie le nombre de diviseurs de l'entier n'''
3     c = 0
4     for i in range(1,n+1):
5         if n%i == 0:
6             c+= 1
7     return c
8
9 def nb_div(n):
10    '''Renvoie le nombre de diviseurs de l'entier n'''
11    c = 0
12    fin = n
13    k = 1
14    while k<fin:
15        if n%k == 0:
16            fin = n//k
17            if k == fin:
18                c +=1
19            else:
20                c +=2
21        k += 1
22    return c
```

2 Que fait ce programme ?

```
1 for a in range(500):
2     for b in range(a,500):
3         c2 = a**2 + b**2
4         c = c2**0.5
5         if c == int(c):
6             if a + b + c == 1000:
7                 print(a*b*c)
```

¹En cas de problème informatique, il est toujours possible de récupérer ce dossier directement sur le site internet <http://pcsi.kleber.free.fr/IPT/>

Partie II

Mise en jambe

1 Un lecteur fou...

Les pages d'un livre sont numérotées de 1 à n . Un lecteur – sans doute complètement fou – additionne les numéros des pages. La tâche est rude mais au bout du compte, il obtient 1994. Il se rend compte qu'il s'est trompé ; il a compté une page deux fois sans se rappeler laquelle !

Essayez de définir une fonction qui, connaissant la somme, renvoie la page en double et le nombre total de page². Appliquez là ensuite à votre cas particulier et stockez dans la variable `page_en_double` la page en double et dans la variable `nb_total_pages` le nombre total de pages.

2 Prison de Sikinia

Dans la prison centrale de Sikinia, il y a 30 cellules numérotées 1, 2, 3, ..., 30, toutes occupées. Les portes des cellules peuvent être dans deux états : ouvertes ou fermées. On peut passer d'un état à l'autre en faisant faire un demi-tour au bouton de la porte. Au moment où commence l'histoire, toutes les portes sont fermées.

Pour fêter le vingtième anniversaire de la république de Sikinia, le président décide d'une amnistie. Il donne au directeur de la prison les ordres suivants: « Tournez successivement d'un demi-tour les boutons :

- de toutes les portes,
- puis d'une porte sur deux, à partir de la deuxième,
- puis d'une porte sur trois, à partir de la troisième,
- puis d'une porte sur quatre, à partir de la quatrième.
- ...

Continuez ainsi jusqu'à la dernière cellule. Libérez alors les prisonniers dont la porte de cellule est ouverte. »

Pour des raisons de sécurité, le directeur de la prison aimerait connaître à l'avance quels seront les prisonniers libérés. Pouvez-vous l'aider ? Vous écrirez un programme plus général `liberation_prisonniers(n)` qui prend en argument le nombre de cellules de la prison (numérotées de 1 à n) et qui doit renvoyer la liste des numéros des cellules ouvertes. (Problème tiré de *Elements of mathematics*, 1975, St-Louis (Missouri))

3 Rentrée des classes (ou beaucoup de bruit pour rien...)

a) Énoncé

C'est le jour de la rentrée des classes et les instituteurs ont décidé d'organiser un jeu dans la cour avec tous les élèves de l'école. Ils sont tous placés en rang sur une ligne de départ d'un côté de la cour. À l'autre bout de la cour, les instituteurs ont disposé de nombreux objets. Au signal du départ tous les élèves se mettent à courir pour récolter le plus d'objets possible qu'ils rapportent ensuite à leur position de départ. Un instituteur parcourt le rang pour inscrire le score de chaque élève. Tous les objets sont remis à leur place et le signal de départ d'une seconde manche est donné. L'instituteur passe à nouveau dans le rang pour compter le nouveau score, et inscrire l'addition. Il ne sait pas alors qu'il a parcouru le rang dans l'autre sens et que ses résultats seront faux !

Par exemple, lors de la première manche les élèves ont obtenu dans l'ordre 3 5 2 8 3. Lors de la seconde manche ils ont obtenu 3 1 3 2 5. Le score total devrait donc être 6 6 5 10 8 mais l'instituteur a additionné 5 2 3 1 3 à 3 5 2 8 3 et a donc obtenu 8 7 5 9 6.

Les objets sont rangés pendant que l'instituteur établit le classement. Les vainqueurs sont annoncés mais d'autres enfants protestent. L'instituteur vérifie ses comptes et s'aperçoit de son erreur. En tant qu'instituteur chargé de l'enseignement de l'informatique, vous suggérez de corriger les scores à l'aide d'un programme, ce qui sera plus rapide que de le faire à la main.

²Par exemple si la somme vaut 5, c'est qu'il y a 2 pages en tout et que la page 2 a été comptée en double. Si la somme vaut 7, c'est qu'il y a 3 pages et que la première page a été comptée en double.

Écrivez un programme `corrige_score(premiere_manche,final_errone)` qui, étant données la liste des scores de la première manche et la liste erronée du score total, donne la liste correcte du score total.

b) Commentaires sur un exemple

Il y a 6 élèves qui participent au jeu. Lors de la première manche, leurs scores respectifs sont $[0,1,2,3,4,5]$, dans l'ordre du rang de gauche à droite. Après la deuxième manche, l'instituteur additionne les nouveaux scores (mais à l'envers), et obtient les scores totaux $[1,5,3,6,6,5]$. Ses additions sont les suivantes :

0	1	2	3	4	5	(scores de la première manche, fournis)
+ 1	4	1	3	2	0	(scores inversés de la 2ème manche, non fournis)
1	5	3	6	6	5	(résultats obtenus par l'instituteur, fournis)

alors qu'il aurait du calculer :

0	1	2	3	4	5	
+ 0	2	3	1	4	1	(scores dans le bon sens)
0	3	5	4	8	6	(résultats corrects, à afficher par votre programme)

Votre programme doit afficher le résultat correct des additions, soit $[0,3,5,4,8,6]$ sur cet exemple.

Partie III

Feux de forêt

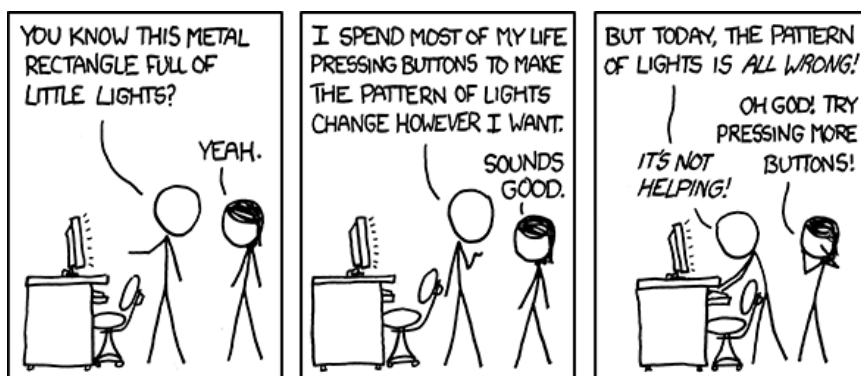
La foudre se déchaîne sur les Vosges. Les éclairs déchirent le ciel, offrant un spectacle à la fois sublime et effrayant. Néanmoins, les habitants du coin savent que la menace principale provient des feux de forêt: chaque éclair s'abattant sur la forêt engendre un départ de feu. Vous décidez d'aider les pompiers à s'organiser dans leur combat contre les flammes.

La forêt, qui s'étend tout le long de la crête de la montagne³, peut être représentée par une suite de cases, chacune à une altitude donnée. Lorsque la foudre s'abat sur une case, celle-ci s'enflamme, puis se propage ensuite aux cases voisines, toujours en montant. Autrement dit, le feu se propage d'une case aux cases voisines de même altitude ou d'altitude supérieure.

Écrivez un programme `feux_de_foret(altitudes,impacts)` qui détermine le nombre de cases de forêt qui risquent de brûler connaissant la liste `altitudes` des altitudes des cases successives le long de la crête et la liste `impacts` des positions (variant de 0 à `len(altitudes)-1`) où frappent les éclairs (ces positions étant données dans un ordre a priori quelconque).

Votre programme doit afficher un unique entier : le nombre de cases de forêt menacées par les flammes.

Pour passer tous les tests, votre programme doit être aussi efficace que possible (limitations en temps à 1 s sachant que le nombre de cases et le nombres d'éclairs peuvent varier dans $[1; 40\,000]$).



This is how I explain computer problems to my cat. My cat usually seems happier than me.

³Enfin des collines, quoi, c'est les Vosges...

Partie IV

Des histoires de petits pois

Un marchand de légumes très maniaque souhaite ranger ses petits pois en les regroupant en boîtes de telle sorte que chaque boîte contienne un nombre factoriel de petits pois. On rappelle qu'un nombre est factoriel s'il est de la forme $1, 1 \times 2, 1 \times 2 \times 3, 1 \times 2 \times 3 \times 4 \dots$ et qu'on les note sous la forme suivante : $n! = 1 \times 2 \times 3 \times \dots \times (n-1) \times n$

Il souhaite également utiliser le plus petit nombre de boîtes possible. Ainsi, s'il a 17 petits pois, il utilisera :

- 2 boîtes de $3! = 6$ petits pois, soit 12 petits pois rangés;
- 2 boîtes de $2! = 2$ petits pois, soit 4 petits pois rangés;
- 1 boîte de $1! = 1$ petit pois, soit 1 petit pois rangé.

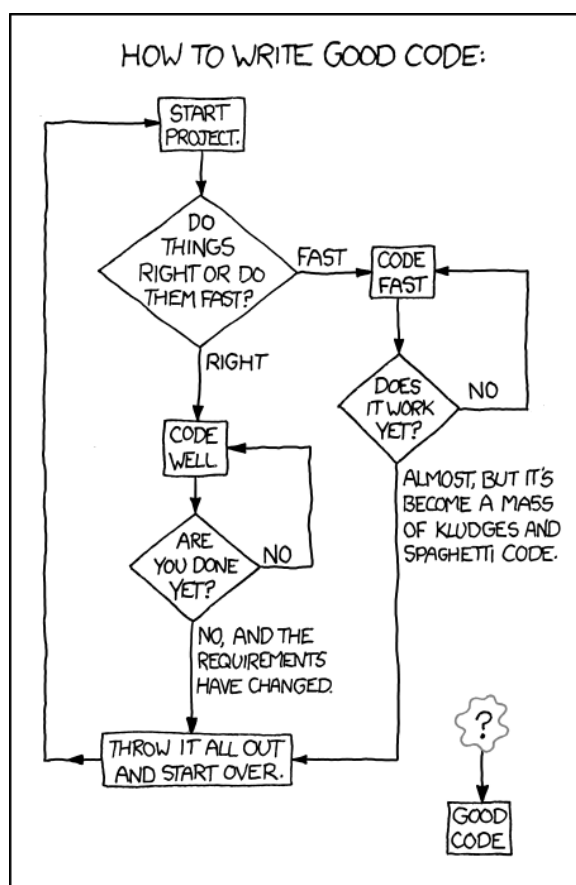
ce qui donne bien $2 \times 3! + 2 \times 2! + 1 \times 1! = 12 + 4 + 1 = 17$.

D'une manière générale, s'il a `nb_petits_pois`, il doit trouver une suite $a_1, a_2, \dots, a_p$ d'entiers positifs ou nuls avec $a_p > 0$ et telle que

$$\text{nb_petits_pois} = a_1 \times 1! + a_2 \times 2! + \dots + a_p \times p!$$

avec $a_1 + \dots + a_p$ minimal.

Votre programme `range_petits_pois(nb_petits_pois)` doit renvoyer la liste $[a_1, a_2, \dots, a_p]$ (unique) qui permettra à l'assistant du marchand de préparer les boîtes avant d'y ranger ses petits pois. Bien entendu, tout comme les factoriels, le nombre de petits pois à ranger peut vite devenir astronomique⁴ et il ne faut pas que le programme prenne plus de 1 s pour donner sa réponse.



You can either hang out in the Android Loop or the HURD loop.

⁴Le marchand a tendance à faire ses achats en gros, mais on assure que ce nombre n'excèdera pas 10^{1700}