Épreuve d'Informatique – Session 2020 – Filière PSI

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- ✓ Cette épreuve est composée d'un exercice et de trois parties tous indépendants ;
- ✓ Toutes les instructions et les fonctions demandées seront écrites en Python ;
- ✓ Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- ✓ Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈

Exercice : (4 points)

Les coefficients binomiaux

1 pt **Q1-** Écrire la fonction `fact(p)` qui reçoit en paramètre un entier positif **p**, et qui retourne la valeur de **factorielle p** : **p! = 1 * 2 * 3 * … * (p-1) * p**.

NB : La fonction **fact(0)** retourne 1

0.5 pt **Q2-** Déterminer la complexité de la fonction `fact(p)`

Un **coefficient binomial** est défini pour deux entiers positifs **n** et **k** tels que **k≤n**. C'est le nombre de parties de **k** éléments dans un ensemble de **n** éléments. On le note : $\binom{n}{k}$, et sa valeur est calculée par la formule suivante :

$$\binom{n}{k} = \frac{n!}{k! * (n-k)!}$$

1 pt **Q3-** Écrire la fonction `binomial(n,k)` qui reçoit en paramètres deux entiers positifs **n** et **k** tels que **k≤n**, et qui retourne la valeur du coefficient binomial $\binom{n}{k}$.

1.5 pt **Q4-** Écrire la fonction `binomiaux(n)`, qui reçoit en paramètre un entier positif **n**, et qui affiche tous les coefficients binomiaux $\binom{n}{k}$ tel que : **k = 0, 1, 2, … , n**.

Exemple :

La fonction **binomiaux (6)** affiche les nombres : **1** , **6** , **15** , **20** , **15** , **6** et **1**

Partie I : Calcul numérique

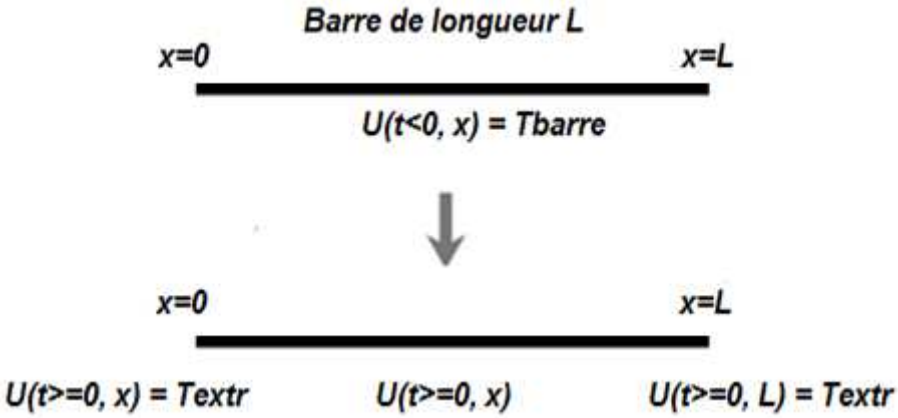
Équation de la diffusion thermique

Solution numérique

On considère une barre solide de longueur L, de coefficient de diffusion thermique D.

La barre est initialement "préparée" dans un état de température $Tbarre$.

Les deux extrémités de la barre sont maintenues à une température extérieure constante $Textr$.



L'équation de la diffusion thermique à une dimension, est la suivante :

$$(E) \qquad \frac{dU(t,x)}{dt} = D * \frac{d^2U(t,x)}{dx^2}$$

Avec : $U(t, x)$ est la température de la position x dans la barre de la barre, à l'instant t.

On recherche une solution numérique à ce problème par la méthode classique **des différences finies**.

Supposons que nous cherchions l'évolution de $U(t, x)$ sur une durée totale T.

La durée totale T d'évolution est subdivisée en n sous-intervalles de durée : $\mathbf{dt} = \frac{\mathbf{T}}{\mathbf{n}}$

Ainsi, l'instant "discret" t_i est : $t_i = i * dt$ avec $i \in [\![0, n]\!]$

De même, la barre de longueur L est spatialement subdivisée en p tronçons de longueur : $\mathbf{dx} = \frac{\mathbf{L}}{\mathbf{p}}$

Ainsi, l'abscisse "discrète" x_i est : $x_i = i * dx$ avec $i \in [\![0, p]\!]$

On suppose que les variables globales suivantes, sont déclarées et initialisées :

- ✓ $L = 1.0$ La longueur de la barre
- ✓ $D = 0.3$ Le coefficient de diffusion thermique
- ✓ $T = 1.0$ La durée totale de l'évolution
- ✓ $Tbarre = 100.0$ La température de la barre
- ✓ $Textr = 20.0$ La température extérieure
- ✓ $n = 10000$ Le nombre de subdivision de T
- ✓ $p = 100$ Le nombre de subdivision de L

On suppose que les modules **numpy** et **matplotlib.pyplot** sont importés :

```
import numpy as np
import matplotlib.pyplot as pl
```

Q.1- Écrire la fonction **`vecteur_init()`** qui retourne un vecteur U de longueur $p+1$, tel que :

- $U[i] = Textr$ si $i = 0$ ou $i = p$
- $U[i] = Tbarre$ si $1 \leq i \leq p-1$

La fonction **vecteur_init ()** retourne le vecteur U suivant :

[20. 100. 100. 100. … 100. 100. 100. 20.]

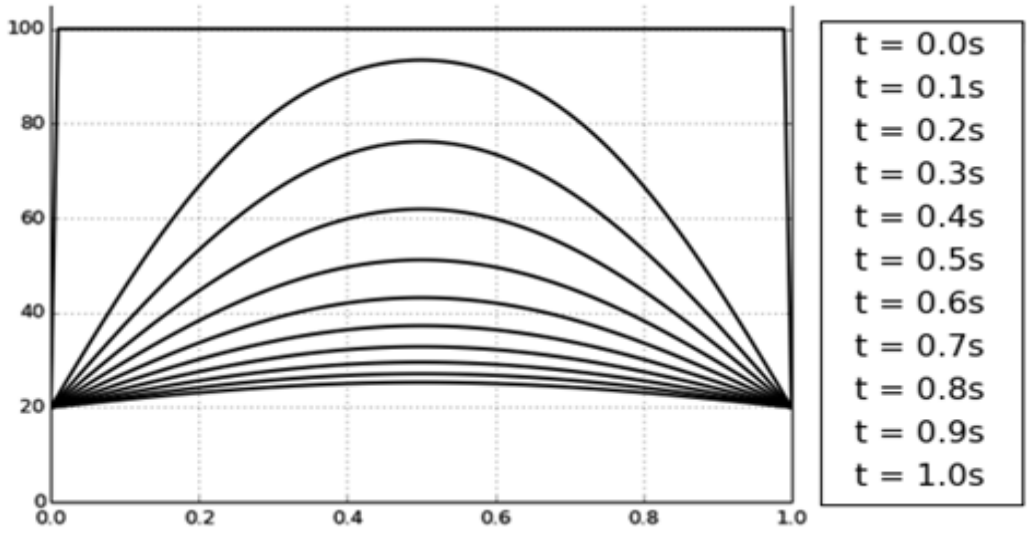
Chaque élément $U[i]$ représente la température du point $i*dx$ dans la barre, à l'instant $t_0 = 0\ s$.

L'équation de la diffusion thermique (E) discrétisée s'écrit sous forme d'une relation de récurrence qui permet d'obtenir la température de la barre à l'instant t_{i+1} en fonction de la température de la barre à l'instant t_i : Si le vecteur U représente la température de la barre à l'instant t_i, alors la température de la barre à l'instant t_{i+1} est représentée par le vecteur V, tel que :

- Si $i = 0$ ou $i = p$ alors $V[i] = U[i]$
- Si $1 \leq i \leq p-1$ alors $V[i] = U[i] + (D * \frac{dt}{dx^2}) * (U[i-1] - 2 * U[i] + U[i+1])$

Q.2- Écrire la fonction **`differences_finies(U)`** qui reçoit en paramètre un vecteur U qui représente la température de la barre à un instant t_i. La fonction retourne le vecteur V qui représente la température de la barre à l'instant t_{i+1}.

Q.3- Écrire le programme qui trace la représentation graphique de l'évolution de la température de la barre, à intervalle de temps égal à $0.1s$



NB : Chaque courbe représente la température de la barre à un instant t

Partie II : Base de donnés et langage SQL

Classement des candidats

Une grande école d'ingénieurs organise un concours au profit des étudiants (candidats), qui veulent y accéder pour suivre leurs études supérieures. Le concours est composé de plusieurs épreuves : une seule épreuve par matière.

Après le passage des épreuves, une note totale est calculée pour chaque candidat. Ensuite les candidats sont classés dans l'ordre décroissant de la note totale.

*NB : Une note inférieure à **5.0** est une note éliminatoire du classement final. Si un candidat possède au moins une note éliminatoire, alors ce candidat sera exclu du classement final.*

L'école d'ingénieurs utilise une base de données relationnelle composée de trois tables :

Candidats

numéro	*(entier)*
nom	*(texte)*
prénom	*(texte)*

Notes

numC	*(entier)*
codM	*(texte)*
note	*(réel)*

Matières

code	*(texte)*
nom	*(texte)*
coeff	*(entier)*

La table **'Candidats'** contient les numéros, les noms et les prénoms des candidats. Le champ **numéro** est la clé primaire dans cette table.

Exemples :

numéro	nom	prénom
416	Jarfaoui	Hicham
70	Hilal	Samira
162	Senhaji	Amal
23	Bakouri	Ahmed
…	**…**	**…**

La table **'Matières'** contient les codes, les noms et les coefficients des matières. Le champ **code** est la clé primaire dans cette table.

Exemples :

code	nom	coeff
M	Mathématiques	14
P	Physique	10
SI	Sciences de l'ingénieur	6
Ch	Chimie	3
…	**…**	**…**

La table **'Notes'** contient, pour chaque candidat, la note correspondante à chaque matière. Les champs **numC** et **codM** sont deux clés étrangères, qui font respectivement référence, aux champs **numéro** et **code** des tables 'Candidats' et 'Matières'.

Exemples :

numC	codM	note
70	**SI**	14,50
162	**SI**	17,00
416	**SI**	12,25
70	**M**	16,00
162	**M**	13,50
416	**M**	08,35
70	**P**	04,75
162	**P**	11,05
416	**P**	10,10
…	**…**	**…**

Q.1 – Déterminer la clé primaire de la table 'Notes', et justifier votre réponse.

Q.2 – Écrire, en algèbre relationnelle, une requête qui donne les numéros, les noms et prénoms de tous les candidats, qui ne possèdent pas de note éliminatoire, en matière de code **'M'**.

Q.3 – Écrire la requête précédente (**Q.2**) en langage SQL.

Q.4 – Écrire, en langage SQL, une requête qui donne les noms des matières, la note maximale et la note minimale de chaque matière, triés dans l'ordre décroissant de la moyenne des notes de chaque matière.

Q.5 – Écrire, en langage SQL, une requête qui donne le compte des candidats qui sont exclus du classement final.

Q.6– Pour les candidats non exclus du classement final, la **note totale** de chaque candidat est calculée par la formule suivante :

$$\boldsymbol{note\ totale} = \sum (\boldsymbol{note} * \boldsymbol{coefficient})$$

La note d'une matière est multipliée par le coefficient correspondant à cette matière.

Écrire, en langage SQL, une requête qui donne le numéro, le nom, le prénom et la note totale de chaque candidat non exclu, ayant la note totale supérieure strictement à ***1000.0***, triés dans l'ordre décroissant de la note totale.

Partie III : Problème

Transformée de 'Burrows-Wheeler'

La **transformée de Burrows-Wheeler**, couramment désignée par l'acronyme **BWT** (pour *Burrows-Wheeler Transform*) est une technique inventée par Michael Burrows et David Wheeler, et elle a été publiée en 1994. Cette technique permet de réorganiser les données : c'est une transformation qui, à partir d'un texte donné, produit un autre texte contenant exactement les mêmes caractères, mais dans un autre ordre, pour lequel les répétitions de caractères ont tendance à être contiguës.

Le but de cette partie est de réaliser l'algorithme de la transformée de Burrows-Wheeler.

1- Rotation d'une chaîne de caractères

*On considère une chaîne de caractères **T** non vide, et un entier **p** tel que : **0 ≤ p < taille de T**.*

*La **rotation de T d'indice p** est la chaîne de caractères obtenue par la concaténation des deux sous-chaînes suivantes :*

- *La sous-chaîne de **T**, composée de la suite des caractères en partant de la position **p**, jusqu'à la dernière position dans **T** ;*
- *et de la sous-chaîne de **T**, composée de la suite des caractères en partant de la première position de **T**, jusqu'à la position **p-1** dans **T**.*

Exemples :

T = 'daacdacdab'

- La rotation de **T** d'indice **0** est la chaîne : **'daacdacdab'** + '' = **'daacdacdab'**
- La rotation de **T** d'indice **1** est la chaîne : '**aacdacdab**' + 'd' = **'aacdacdab**d**'**
- La rotation de **T** d'indice **2** est la chaîne : '**acdacdab**' + 'da' = **'acdacdab**da**'**
- La rotation de **T** d'indice **9** est la chaîne : '**b**' + 'daacdacda' = **'b**daacdacda**'**

Q.1- Écrire la fonction `rotation(T,p)`, qui reçoit en paramètres une chaîne de caractères **T** non vide, et un entier **p** tel **0 ≤ p < taille de T**. La fonction retourne la rotation de **T** d'indice **p**.

2- Liste des rotations d'une chaîne de caractères

Q.2- Écrire la fonction `liste_rotations(T)`, qui reçoit en paramètre une chaîne de caractères **T** non vide, et qui retourne une liste **R** qui contient toutes les rotations possibles de **T**.

Exemple :

T = 'daacdacdab' (la taille de la chaîne **T** est **10**)

La fonction **liste_rotations (T)** retourne la liste des **10** rotations possibles :

R = ['daacdacdab', 'aacdacdabd', 'acdacdabda', 'cdacdabdaa', 'dacdabdaac', 'acdabdaacd', 'cdabdaacda', 'dabdaacdac', 'abdaacdacd', 'bdaacdacda']

3- Tri de la liste des rotations

Q.3- Écrire la fonction `tri_rotations(R)`, qui reçoit en paramètre la liste **R** qui contient toutes les rotations d'une chaine de caractères. La fonction trie les éléments de **R** dans l'ordre alphabétique.

Exemple :

R= ['daacdacdab', 'aacdacdabd', 'acdacdabda', 'cdacdabdaa', 'dacdabdaac', 'acdabdaacd', 'cdabdaacda', 'dabdaacdac', 'abdaacdacd', 'bdaacdacda']

Après l'appel de la fonction **tri_rotrations (R)**, on obtient la liste triée :

['aacdacdabd', 'abdaacdacd', 'acdabdaacd', 'acdacdabda', 'bdaacdacda', 'cdabdaacda', 'cdacdabdaa', 'daacdacdab', 'dabdaacdac', 'dacdabdaac']

4- Position de la chaîne initiale dans la liste des rotations triée

La liste **R** contient toutes les rotations de la chaine de caractères **T**. On suppose que la liste **R** est triée dans l'ordre alphabétique. La chaîne **T** est un élément de la liste **R** (**T** est la rotation d'indice **0**). Et on veut chercher la position de **T** dans la liste **R** triée. Pour cela, on propose d'utiliser le principe de la **recherche dichotomique**.

La **recherche dichotomique** est un algorithme de complexité logarithmique, qui permet de rechercher la position d'un élément **x** dans une liste **L** triée. Le principe de cet algorithme est le suivant :

1. *Calculer la position **m** du milieu de la liste **L** ;*
2. *Si **L[m] = x**, alors retourner **m** (**m** est la position de **x** dans **L**) ;*
3. *Si **x<L[m]**, alors reprendre l'étape **(1)** en considérant la* première *moitié de la liste **L** ;*
4. *Si **x>L[m]**, alors reprendre l'étape **(1)** en considérant la* deuxième *moitié de la liste **L**.*

Q.4.a- Écrire la fonction `position(T,R)`, qui reçoit en paramètres une chaîne de caractères **T**, non vide, et la liste **R** de toutes les rotations de **T**, triée dans l'ordre alphabétique. La fonction retourne la position de la chaîne **T** dans la liste triée **R**, en utilisant le principe de la recherche dichotomique.

Exemple :

T = 'daacdacdab'

La liste des rotations de T, triée dans l'ordre alphabétique est :

R = ['aacdacdabd', 'abdaacdacd', 'acdabdaacd', 'acdacdabda', 'bdaacdacda', 'cdabdaacda', 'cdacdabdaa', **'daacdacdab'**, 'dabdaacdac', 'dacdabdaac']

La fonction **position (T, R)** retourne le nombre **7**, qui représente l'indice de la chaîne **T** dans la liste des rotations triée **R**.

Q.4.b- Déterminer la complexité de la fonction **position(T, R)**.

5- Transformée de Burrows-Wheeler

La transformée de Burrows-Weeler se contente de réorganiser les caractères d'une chaîne de caractères **T**, de manière à obtenir un autre texte contenant exactement les mêmes caractères de **T**, mais dans un autre ordre, pour lequel les répétitions de caractères ont tendance à être contiguës.

Le principe de la transformée de Burrows-Weeler est le suivant :

1. On construit la liste **R** de toutes les rotations possibles de la chaîne **T** ;
2. On trie les éléments de la liste **R** dans l'ordre alphabétique ;
3. On recherche la position **p** de la chaîne **T** dans la liste **R** triée ;
4. On construit la chaîne **B** composée des derniers caractères des chaines de la liste **R** triée ;
5. Le résultat de la transformée de Burrows-Wheeler est le tuple **(B, p)**.

Exemple :

T = 'daacdacdab'

La liste des rotations de **T**, triée dans l'ordre alphabétique est :

R = ['aacdacdab**d**', 'abdaacdac**d**', 'acdabdaac**d**', 'acdacdabd**a**', 'bdaacdacd**a**', 'cdabdaacd**a**', 'cdacdabda**a**', 'daacdacda**b**', 'dabdaacda**c**', 'dacdabdaa**c**']

La position de **T** dans **R** est : **p=7**.

La chaîne composée des derniers caractères des chaînes de la liste **R** triée est : **B = 'dddaaaabcc'**.

Le résultat de la transformée de Burrows-Wheeler est le tuple : **('dddaaaabcc'** , **7)**

Q.5- Écrire la fonction **`BWT(T)`**, qui reçoit en paramètre une chaîne de caractères **T** non vide, et qui retourne le tuple **(B, p)**.

Exemple :

T = 'daacdacdab'

La fonction **BWT (T)** retourne le tuple : **('dddaaaabcc'** , **7)**

6- La compression RLE

La transformée de Burrows-Wheeler est à la base de certains algorithmes de compression. Elle est utilisée avant la compression de données.

La **compression RLE (R**un **L**ength **E**ncoding) est un algorithme qui permet de compresser un texte contenant des suites de caractères identiques. Il consiste à repérer la répétition des caractères dans le texte, et de les remplacer par un chiffre qui représente le nombre de répétition d'un caractère, suivi du caractère répété.

Par exemple, on considère la chaîne **S = 'BBBBZZZZZZZZZZZZZZZZZZZZZZZZBBBX'**

Après la compression de la chaîne **S**, on obtient la chaîne **'4B9Z9Z5Z3B1X'**

Q.6.a- Écrire la fonction **`decompression(ch)`**, qui reçoit en paramètre une chaîne de caractères **ch** qui représente une chaîne compressée par l'algorithme de compression RLE. La fonction retourne la chaîne décompressée.

Exemple :

La fonction **decompression ('4B9Z9Z5Z3B1X')** retourne la chaîne de caractères décompressée :

'BBBBZZZZZZZZZZZZZZZZZZZZZZZZBBBX'

Q.6.b- Écrire la fonction **`RLE(S)`**, qui reçoit en paramètre une chaîne de caractères **S**. La fonction retourne la chaîne compressée de **S**, en utilisant le principe de la compression **RLE**.

Exemple :

S = 'BBBBZZZZZZZZZZZZZZZZZZZZZZZZZBBBX'

La fonction **RLE (S)** retourne la chaîne de caractères **'4B9Z9Z5Z3B1X'**

NB : Dans cet exemple, le caractère '**Z'** est répété **23** fois. La compression de cette suite de caractères donne pour résultat : **'9Z9Z5Z',** et non pas **'23Z'**.

≈ ≈ ≈ ≈ ≈ ≈ FIN DE L'ÉPREUVE ≈ ≈ ≈ ≈ ≈ ≈