Collège de Candolle

ÉPREUVE REGROUPÉE

Discipline: INFORMATIQUE **Date:** 22/12/2023
Cours: 2 IN DP 02 et 08 **Durée:** 105 minutes

Maître: JC Barros
Matériel autorisé: aucun (aide-mémoire inclus à la fin de l'épreuve)

Consignes de travail:
- **Ecriture lisible et présentation soignée**
- **Mise en évidence des résultats**
- **Il y a 8 pages + l'aide-mémoire**

Nom et prénom: **Total: 44 points**

Ex 1: On souhaite trier la suite de nombres suivante: [3, 8, 5, 1, 4, 9] **(5 pts)**

Observer les étapes de l'algorithme de tri ci-dessous. (1 pt)

```
[3, 8, 5, 1, 4, 9] # liste de départ
[1, 8, 5, 3, 4, 9] # après 1 échange
[1, 3, 5, 8, 4, 9] # après 2 échanges
[1, 3, 4, 8, 5, 9] # après 3 échanges
[1, 3, 4, 5, 8, 9] # après 4 échanges
```

a. Quel est le nom de cet algorithme?

Réponse:

b. Est-ce que l'algorithme de tri est arrivé à son terme? (2 pts)

Réponse explicitée:

c. Combien de **comparaisons** ont dû être réalisées avant le premier échange? (2 pt)

Réponse explicitée:

Ex 2: Stratégie algorithmique... **(3 pts)**

a. Comment s'appelle un algorithme qui résout un problème en le découpant d'abord en sous-problèmes, puis résout les sous-problèmes pour enfin arriver à trouver la solution du problème initial ? (1 pt)

Réponse :

b. Donner deux exemples d'algorithmes qui appartiennent à cette stratégie: (2 pt)

Réponse :

Ex 3: Voici un algorithme écrit en pseudo-code. **(11 pts)**

Remarques: Les positions des listes sont numérotées à partir de 1 (ex: `Liste[2]` vaut 4)

`POUR k de x à y` prend `k = x` pour commencer puis va jusqu'à `y` compris.

```
Entrée: Liste = [3, 4, 1, 2, 6, 5]

n = taille (Liste)

POUR i de n-1 à 1 {
  POUR j de 1 à i {
    SI Liste[j] > Liste[j+1]
    ALORS Liste[j] ↔ Liste[j+1]}}

Sortie: Liste = [1, 2, 3, 4, 5, 6]
```

a. A quoi sert l'algorithme représenté par ce pseudo-code ? (1 pt)

Réponse :

b. Quel est le nom de cet algorithme? (1 pt)

Réponse :

c. Quelle doit être la taille de la liste en entrée pour que cet algorithme fonctionne correctement? (2 pt)

Réponse explicitée :

d. Combien d'échanges (↔) auront lieu dans l'algorithme pendant que `i` vaut `n-1`? (2 pts)

Réponse explicitée :

e. Traduire ce pseudo-code en programme Python. (5 pts)

Les lignes verticales servent à aligner le code en fonction des indentations nécessaires.

Exemple de code aligné sur la grille:

```
for i in range(0, 5):
    print(i)
```

Programme:

Ex 4: Voici le code d'un algorithme nommé "*binary search*". **(11 pts)**

Il reçoit comme entrée une liste ordonnée (`liste`) et une valeur (`valeur`).
Il retourne la position de `valeur` dans la liste si elle s'y trouve.

```python
def binarySearch(liste, valeur):
    debut = 0
    fin = len(liste) - 1
    while fin >= debut:
        milieu = (debut+fin) // 2
        if liste[milieu] == valeur:
            return milieu
        elif liste[milieu] > valeur:
            fin = milieu - 1
        else:
            debut = milieu + 1
    return -1

# entrées
une_liste = [1, 3, 5, 7, 9]
une_valeur = 7

# Appel de la fonction
res = binarySearch(une_liste, une_valeur)

# sortie
print(res)
```

a. Quelle est la stratégie employée dans cet algorithme? (1 pt)
Réponse:

b. Est-ce que l'algorithme fonctionne également pour chercher une valeur de type `float` dans une liste contenant des éléments de type `float`? (1 pt)
Réponse:

c. Combien de fois la boucle while va-t-elle se répéter dans ce programme? (1 pt)
Il peut être utile de faire un tableau avec une colonne par variable et une ligne par étape
Réponse:

d. Quel est le résultat affiché dans la console par ce programme ? (1 pt)
Réponse (précise):

e. Quel serait le résultat affiché par: (1 pt)

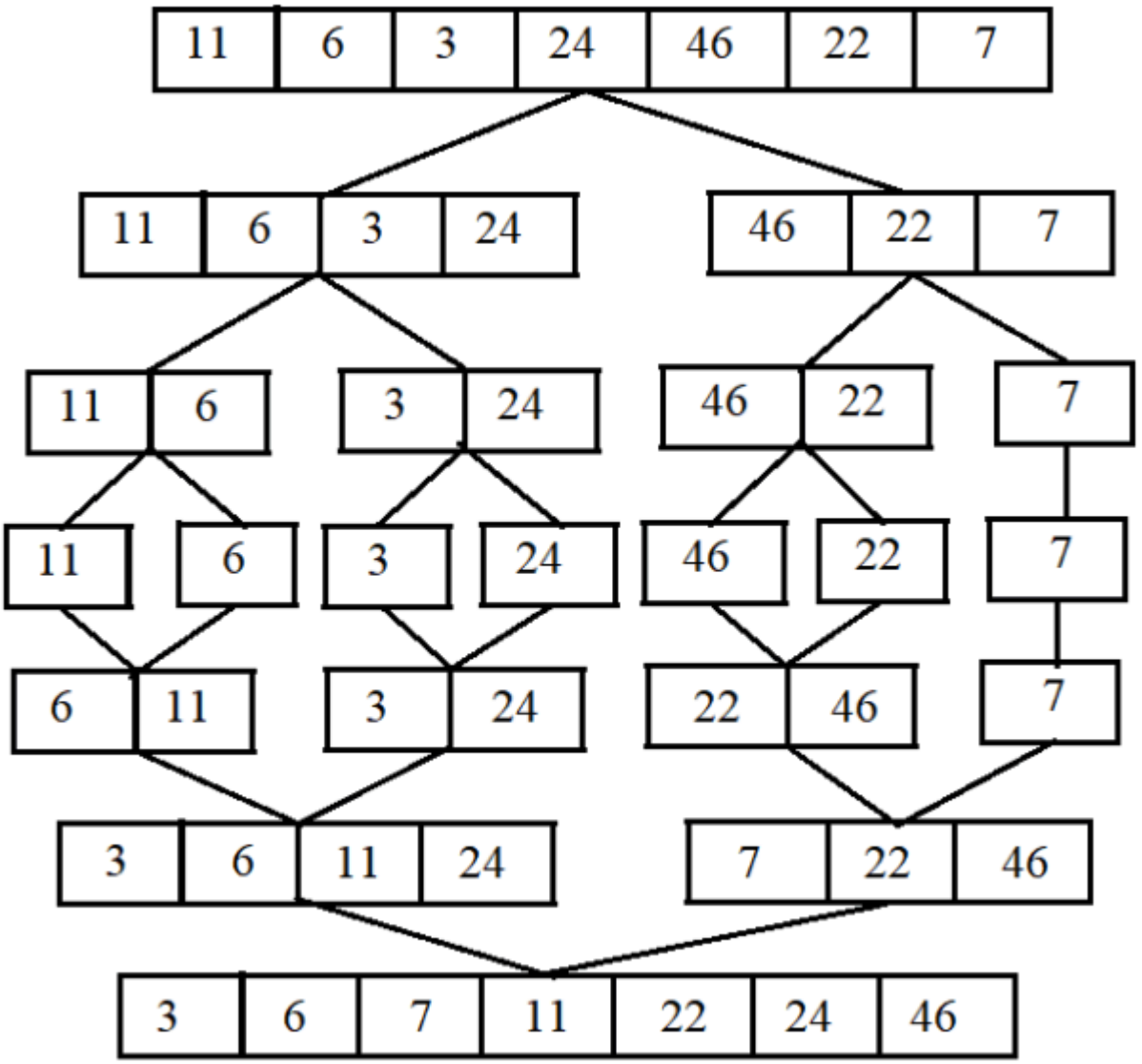
`print(binarySearch([1, 2], 3))` ?

Réponse (précise):

f. Traduire la fonction **`binarySearch`** en logigramme (6 pt)

Réponse:

Ex 5: Voici le déroulement d'un algorithme de tri **(3 pts)**

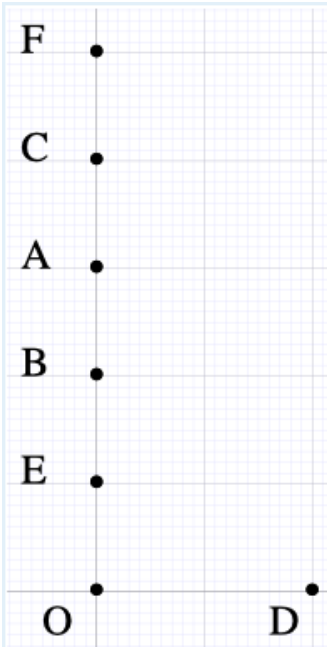


a. Quel est le nom de cet algorithme?

Réponse : (avec deux caractéristiques qui distinguent cet algorithme de tri des autres vus en classe)

Ex 6: Plus court chemin... **(6 pts)**

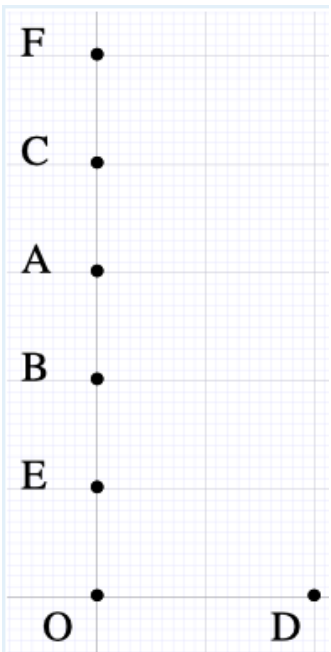
On part du point O et on veut atteindre tous les points(A, B, C, D, E et F) en parcourant la distance totale la plus petite possible.
Le problème consiste donc à trouver l'ordre dans lequel il faut rejoindre les points pour minimiser la distance totale.



a. On veut implémenter une stratégie gloutonne; proposer un **critère** pour choisir le prochain point à atteindre à partir du point où on en est. (2 pts)
Réponse:

b. Tracer le chemin à suivre (sur le graphique en-haut) en adoptant cette stratégie gloutonne (relier les points par des droites). (2 pts)

c. La solution trouvée en b est-elle optimale ? **Si oui**, expliquer pourquoi. (2 pts)
Sinon, tracer le chemin optimal sur le graphique ci-dessous.



Ex 7: Une aventurière doit consommer un maximum d'énergie [cal] **(5 pts)**
dans son alimentation sans dépasser la masse de 100 [g] de nourriture.
Voici la liste des aliments avec chaque masse et énergie:

label	A	B	C	D	E	F	G	H
aliment	poulet	brocoli	riz	saumon	banane	avocat	patate	oeufs
masse [g]	28	70	90	10	35	35	40	50
énergie [cal]	45	35	60	20	30	55	40	30

Utiliser un algorithme glouton pour composer le repas de cette aventurière.

a. Proposer un critère pour choisir "le meilleur" aliment à chaque étape? (1 pts)

b. Compléter le tableau suivant avec les aliments pris par l'algorithme, dans l'ordre.
Attention, il se pourrait qu'il y ait plus de colonnes que nécessaire. (2 pts)

label							**totaux**
masse							
énergie							

c. La solution proposée par cet algorithme est-elle optimale ? (2 pts)

Choisir ci-dessous et compléter.

- **oui**, car...

- **non**, voici une meilleure solution que je propose dans le tableau ci-dessous:

label							totaux
masse							
énergie							

FIN DE L'ÉPREUVE (aide-mémoire suit)

Aide-mémoire informatique

2ème année, 1er semestre

Algorithmes

Stratégies:

Force brute

Consiste à calculer toutes les possibilités d'un problème puis à trouver la (les) solution(s) optimale(s). On trouve toujours la meilleure solution mais selon l'algorithme et la taille de l'entrée, il est possible que le temps de calcul soit rédhibitoire.

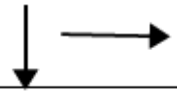
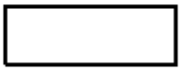
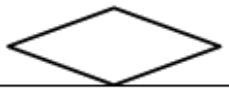
Glouton

On choisit, à chaque étape de l'exécution de l'algorithme, une possibilité qu'on estime optimale (localement). Il faut donc un critère pour faire ce choix. On laisse tomber toutes les autres et on gagne ainsi en efficacité.
Par contre, on n'est pas sûr de trouver la solution optimale (globalement).

Diviser pour régner

On divise un problème complexe en plusieurs sous-problèmes moins complexes. Une fois que les sous-problèmes sont suffisamment petits, on les résout. On finit par combiner les sous-solutions afin d'obtenir la solution globale au problème.

Rappel pour les logigrammes:

Symbole	Nom	Rôle
	Flèche	Indique la séquence du programme
	Terminaison	Débute ou termine un programme
	Processus	Exécute une tâche
	Décision	Génère une décision (oui, non, etc.)
	Entrée/Sortie	Accepte une entrée ou produit une sortie

Programmation (Python)

Calculs :
opérations de base: + - * / **
reste et quotient de division euclidienne: % //
var % div vaut 0 si var est divisible par div
ex: 5 ** 2 → 25 5 // 2 → 2 5 % 2 → 1

Variables et affectations :
types: int, float, str, bool, list ...
int(), float() pour **convertir** en int ou en float
affectation simple : `ma_variable = -5`
permutation de variables: `a, b = b, a`
affectations combinées: +=, -=, *=, etc.
ex: x += 1 est comme x = x + 1

Listes :
liste: `L = [-1, 3.2, “abc”]`
liste vide : `[]`
multiplication: `[0]*3 -> [0, 0, 0]`
élément d’une liste: `L[1] = 5`
doit déjà exister, l’indice commence à 0.
ajouter un élément à une liste :
`L.append(-2.5)`
supprimer élément à la position i :
`del L[i]` ou `L.pop(i)`
supprimer élément de valeur v dans L :
`L.remove(v)`
longueur d’une liste L :
`len(L)`
somme des valeurs d’une liste:
`sum(liste)`
slice: `L[i:j]` (i inclus ! j exclu)

Chaînes de caractères :
chaîne: `txt="bonjour"` ou `txt='bonjour'`
concaténation: txt + " est un texte"
élément, slice d’une chaîne: texte[i]
comme listes mais non modifiable

Entrées/sorties :
sortie: print(valeur,... , end='\n')
valeur de type quelconque, end optionnel
entrée: texte = input("message")
Attention: retourne toujours un str

Commentaires :
sur une ligne : # texte
plusieurs lignes : ’’’texte’’’ ou """texte"""

Structures de contrôle :
1ère ligne finit par ':' puis indentation des lignes du bloc
tests simples: ==, !=, <, <=, >, >=, in
tests combinés: and, or, not, a < b < c

conditionnelle
```
if condition1:
    instructions
elif condition2:
    # elif =contraction de else if
    instructions
else:
    instructions
```

boucle while (boucle non bornée) :
```
while condition:
    instructions
```

boucle for (boucle bornée) :
```
for i in liste: # boucle parcourt
la liste
    instructions
for i in range(5):#pour i de 0 à 4
    instructions
```

range(i,j,k): de i (inclus) à j (exclu) par pas de k
sauter, quitter une boucle : continue, break

Fonctions:
définition de la fonction f à deux paramètres:
```
def f(x, y):
    reponse = x**2 + y**2
    return reponse
```

appel de la fonction: f()
variable = f(3, 5)
print(f(2, 6))
etc.