

Aide-mémoire informatique

2^{ème} année, 1^{er} 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

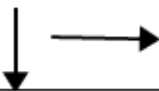
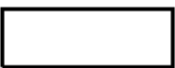
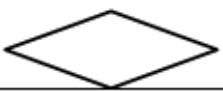
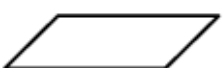
On choisit, à chaque étape de l'exécution de l'algorithme, une possibilité qu'on estime optimale (localement). Il faut donc un critère pour faire ce choix. On laisse tomber toutes les autres et on gagne ainsi en efficacité.

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

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 :

1^{ère} 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.