I. Informatique et ses fondements

2ème partie : Algorithmique

I.2 Algorithmique

1. Tâches
2. Variables
3. Instructions élémentaires
4. Culture algorithmique

I.2 Algorithmique

1. **Tâches**
2. Variables
3. Instructions élémentaires
4. Culture algorithmique

Un problème

Étant donné un tableau de notes :

8	12	20	16,5	11	8,5	3	16	10,5

Combien d'élèves ont la moyenne ? (sachant que le tableau peut être grand)

Un problème

Étant donné un tableau de notes :

8	12	20	16,5	11	8,5	3	16	10,5

Combien d'élèves ont la moyenne ? (sachant que le tableau peut être grand)

8	12	20	16,5	11	8,5	3	16	10,5

On voit bien que ça fait 6.

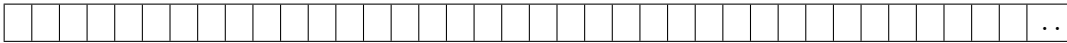
Une tâche

Que sait faire l'ordinateur ?

- faire un calcul
- éventuellement le mettre dans une variable
- exécuter séquentiellement (l'une après l'autre) des tâches
- faire un test pour choisir quelle tâche exécuter ensuite
- faire une boucle (tant que quelque chose ne se passe pas)
- ...

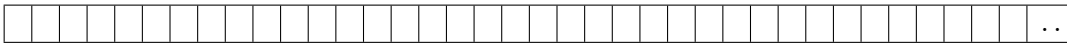
Bref, des tâches simples et unitaires.

Découpage en tâches de notre problème



Pour un million de cases, comment expliquer à l'ordinateur comment faire ?

Découpage en tâches de notre problème



Pour un million de cases, comment expliquer à l'ordinateur comment faire ?

Si je n'ai qu'une case, je la regarde. Si c'est plus grand que 10, je renvoie 1, sinon 0.

Découpage en tâches de notre problème

Pour un million de cases, comment expliquer à l'ordinateur comment faire ?

Si je n'ai qu'une case, je la regarde. Si c'est plus grand que 10, je renvoie 1, sinon 0.

Si j'ai un tableau de $n > 1$ cases, supposons que je sais que sur les $n - 1$ premières, j'ai s élèves qui ont la moyenne.
Alors je regarde la n-ième case. Si c'est plus grand que 10, alors il y a $s + 1$ élèves qui ont la moyenne. Sinon, il y en a s.

Mon algorithme (en pseudo-code)

Entrées/sorties

Entrées : entier `n` strictement positif
tableau `t` de taille `n` (les valeurs sont `t[0]` à `t[n-1]`)

Sortie : entier `s`

Algorithme : nombre de valeurs ≥ 10

```
s ← 0
pour i de 0 à n−1, faire
    si (t[i] ≥ 10)
        alors s ← s+1
    fin_si
fin_pour
```

I.2 Algorithmique

1. Tâches
2. **Variables**
3. Instructions élémentaires
4. Culture algorithmique

Variable ?

- Une variable est un espace mémoire où le programme stocke une valeur ou une structure de donnée compliquée.
- Une variable est supposée changer au cours de l'exécution du programme.

Variable ?

- Une variable est un espace mémoire où le programme stocke une valeur ou une structure de donnée compliquée.
- Une variable est supposée changer au cours de l'exécution du programme.

Exemples :

- Le nombre de followers, de vues sur Youtube, de like. . . *(un entier)*
- La moyenne d'un élève *(un réel ou plutôt un nombre à virgule flottante)*
- Toutes les notes de la classe *(un tableau, un tableau de tableaux, une base de données)*
- Une table de hachage, une liste doublement chaînée
- . . .

Une variable

- doit être créée *(dépend du langage)*
- peut être affectée : `i` $\leftarrow$ `0`
- peut être modifiée : `i` $\leftarrow$ `i+1`
- peut être l'entrée du programme.

Types de variables

- entiers
- nombres flottants
- tableaux de ... *une chaîne de caractères*
- pointeurs vers ...
- structures composées de ...et ...(et ...) un intervalle
- ...

I.2 Algorithmique

1. Tâches
2. Variables
3. **Instructions élémentaires**
4. Culture algorithmique

Instruction élémentaire : l'affectation

variable $\leftarrow$ valeur

On change la valeur d'une variable.

$\Rightarrow$ `i` $\leftarrow$ `17`

$\Rightarrow$ `i` $\leftarrow$ `i+1`

Instruction élémentaire : le test

si booléen **alors** instruction **sinon** instruction

⇒ Une valeur est-elle ≥ 10 ?
⇒ Un calcul risque-t-il de dépasser les valeurs maximales/minimales du type de la variable ?
⇒ Est-ce que je risque d'accéder hors du tableau ?

Instruction élémentaire : la séquence

instruction ; instruction

Par exemple :

- affectation, puis test
- affectation, puis boucle, puis test
- affectation, puis affectation, puis affectation, puis affectation

Instruction élémentaire : la boucle

La boucle pour :

pour variable **de** valeur **à** valeur
faire instruction **fin_pour**

⇒ parcourir un tableau (chercher un max, calculer une moyenne, remplir un tableau...)

La boucle tant que (plus générale) :

tantque booléen **faire** instruction **fin_tantque**

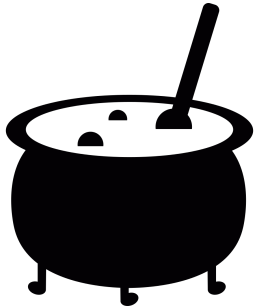
⇒ suite convergente
⇒ itération non bornée (problème de terminaison)

Instructions $\hookrightarrow$ algorithmes

Prenez à la quantité souhaitée :

- la séquence
- le test
- la boucle
- l'affectation de variable

Mélangez bien.



Witch cauldron : Icon made by Freepik from www.flaticon.com http://www.freepik.com/free-icon/witch-cauldron_705580.htm

Vous aurez tous les algorithmes du monde !

Exemple : recherche dichotomique

Entrées/sorties

Entrées : tableau **trié** `t` d'entiers, de taille `n` > 0 et une valeur `v`
Sortie : booléen qui dit si `v` apparaît dans `t`

```
inf ← 0; sup ← n−1;
tantque (inf ≤ sup) faire
  i ← (inf+sup)/2;
  si (t[i] = v) alors renvoie VRAI;
  sinon
    si (t[i] > v) alors sup ← i−1;
                  sinon inf ← i+1;
    fin_si
  fin_si
fin_tantque
renvoie FAUX;
```

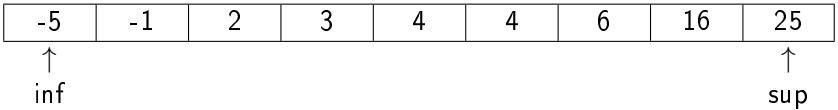
Exemple : recherche dichotomique

Je cherche 5.

-5	-1	2	3	4	4	6	16	25

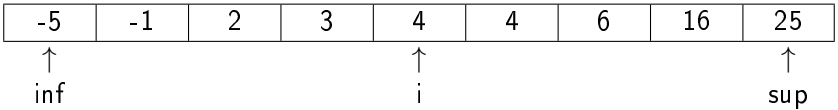
Exemple : recherche dichotomique

Je cherche 5.



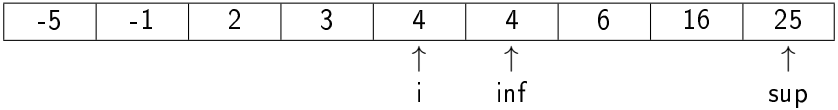
Exemple : recherche dichotomique

Je cherche 5.



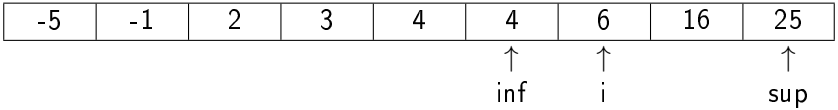
Exemple : recherche dichotomique

Je cherche 5.



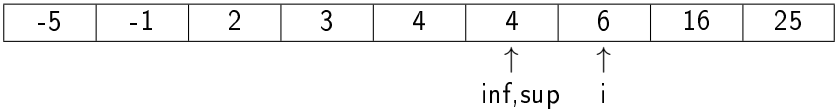
Exemple : recherche dichotomique

Je cherche 5.



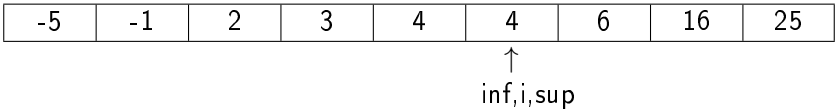
Exemple : recherche dichotomique

Je cherche 5.



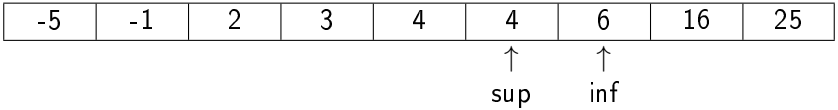
Exemple : recherche dichotomique

Je cherche 5.



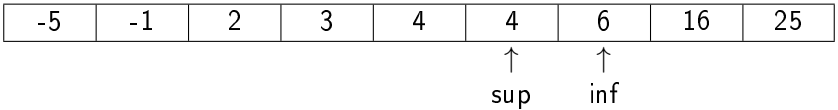
Exemple : recherche dichotomique

Je cherche 5.



Exemple : recherche dichotomique

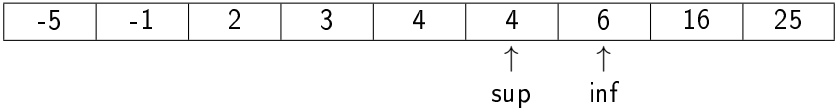
Je cherche 5.



donc 5 n'est pas dans le tableau.

Exemple : recherche dichotomique

Je cherche 5.



donc 5 n'est pas dans le tableau.

En fait `(inf+sup)/2` devrait être `inf+(sup-inf)/2`.

I.2 Algorithmique

1. Tâches
2. Variables
3. Instructions élémentaires
4. **Culture algorithmique**

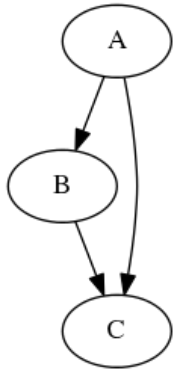
Structures de données complexes

- liste
- liste doublement chaînée
- pile, file
- table de hachage
- arbre, tas
- graphe (orientés ou pas)
- base de données
- ...

Graphes

Graphe = ensemble de nœuds et d'arêtes

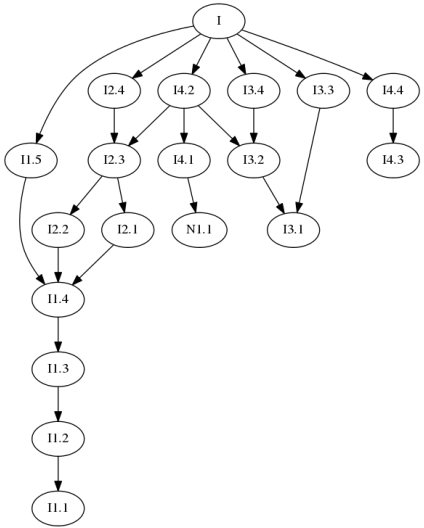
Pour représenter un réseau, des dépendances, des relations...



Structures de données : matrice d'adjacence (nœud $\rightarrow$ nœud $\rightarrow$ booléen) ou une liste de voisins (nœud $\rightarrow$ liste de nœuds).

Parcours de graphe (ancêtres d'un nœud)

Dans ce MOOC, quelles séquences sont nécessaires pour suivre une séquence donnée ?



Parcours de graphe

Entrées/sorties

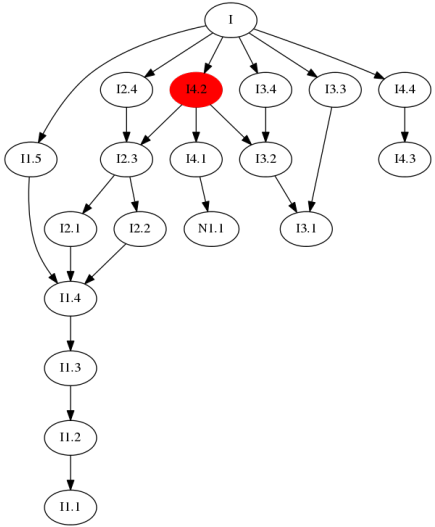
Entrées : graphe de **dépendance** g sous forme d'une liste de voisins
un nœud initial `n`.

Sortie : une liste des descendants de `n` (inclus)

```
res ← vide;
a_traiter ← vide;
a_traiter ← ajouter(n,a_traiter)
tantque a_traiter n'est pas vide
    x ← tête(a_traiter);
    a_traiter ← queue(a_traiter);
    res ← ajouter(x,res);
    l ← g[x];        # la liste des fils de x
    a_traiter ← ajouter(l,a_traiter);
fin_tantque;
```

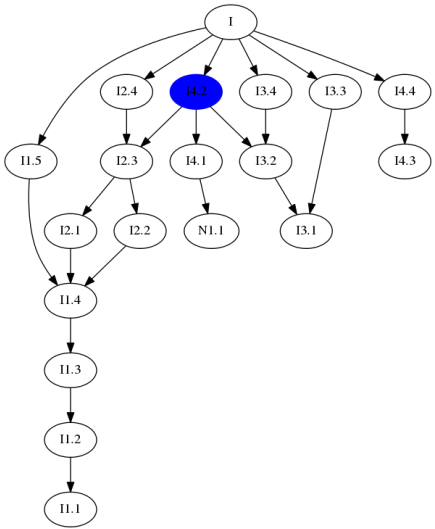
Parcours de graphe : déroulé sur I4.2

`a_traiter` en rouge, `res` en vert et `x` en bleu



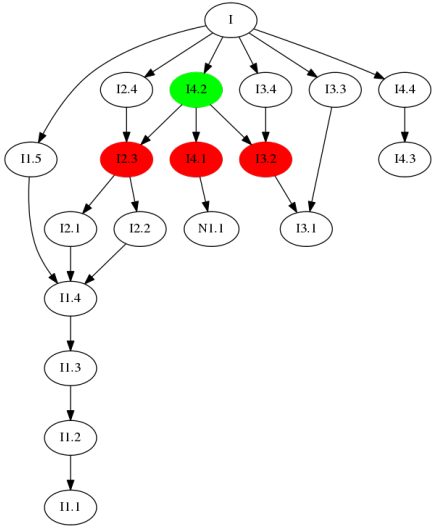
Parcours de graphe : déroulé sur I4.2

`a_traiter` en rouge, `res` en vert et `x` en bleu



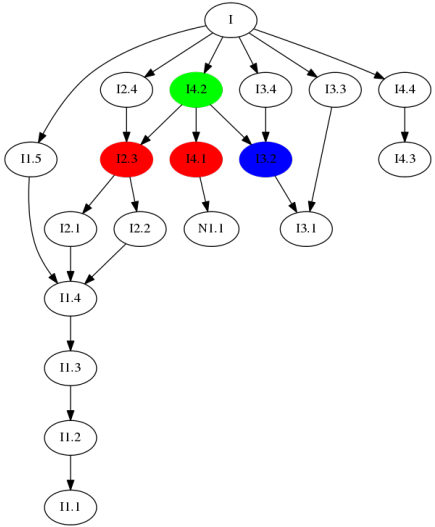
Parcours de graphe : déroulé sur I4.2

`a_traiter` en rouge, `res` en vert et `x` en bleu



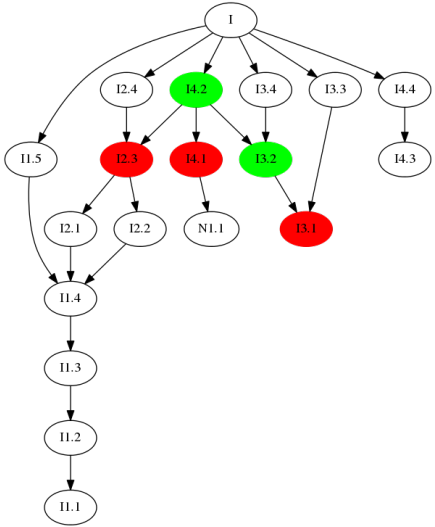
Parcours de graphe : déroulé sur I4.2

`a_traiter` en rouge, `res` en vert et `x` en bleu



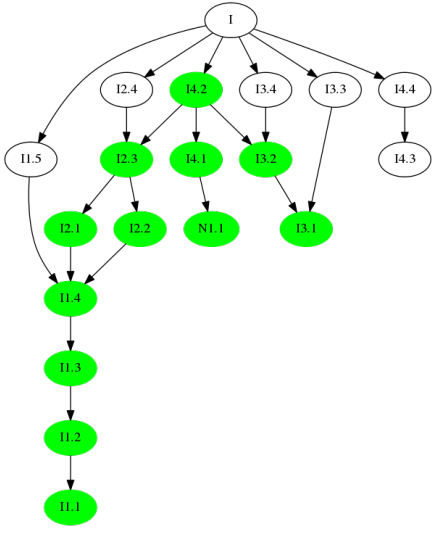
Parcours de graphe : déroulé sur I4.2

`a_traiter` en rouge, `res` en vert et `x` en bleu



Parcours de graphe : déroulé sur I4.2

`a_traiter` en rouge, `res` en vert et `x` en bleu



L'algorithmique ? une science !

- Algorithms (R. Sedgewick et K. Wayne)
- The Art of Computer Programming (D. Knuth)
- Apprendre à programmer avec OCaml (S. Conchon et J.-C. Filliâtre)
- Programmation Efficace (en Python) (C. Dürr et J.-J. Vie)
- Poly de cours de J.-C. Filliâtre (en Java) `http://www.enseignement.polytechnique.fr/informatique/INF411/`
- ...