DUT SRC – IUT de Marne-la-Vallée
15/01/2014
M2202 - Algorithmique

Cours 1
Récursivité

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

Organisation pratique

- **Contact**

- Courriel : philippe.gambette@gmail.com
  (M2202 doit apparaître dans le sujet du courriel).
- Avant ou après le cours.
- Possibilité de poser des questions, de demander des exercices supplémentaires d'entrainement.

- **Notes et devoirs**

Chez soi :

- exercices sur IRIS pour vous inciter à travailler **régulièrement** et corriger vos erreurs. **Annonce sur IRIS 6 jours avant**.
- Note de remplissage de cours à trous et/ou note globale de TP à la fin du semestre, prenant en compte votre avancée à chaque séance.

Pendant les cours :

- Devoir final le 4 juin.

+ note F. Kerdjoudj

Sources

- Cours de Jean-François Berdjugin à l'IUT de Grenoble
http://berdjugin.com/enseignements/inf/inf220/

- Cours de Xavier Heurtebise à l'IUT de Provence
http://x.heurtebise.free.fr

- *Le livre de Java premier langage*, d'A. Tasso

- http://xkcd.com, http://xkcd.free.fr

Programme des cours du semestre

- Récursivité
- Tris
- Notions de complexité
- Listes
- Arbres
- Programmation objet
- Langage PHP

Plan du cours 1 – Récursivité et tris

- Introduction à la récursivité
- Traces d'exécution de fonctions récursives
- Les tris
- Le tri par sélection
- Le tri à bulles
- Complexité des tris

Plan du cours 1 – Récursivité et tris

- Introduction à la récursivité
- Traces d'exécution de fonctions récursives
- Les tris
- Le tri par sélection
- Le tri à bulles
- Complexité des tris

Récursivité

« *La fonction qui s'appelle elle-même* »

2 façons de le voir :

- Une mise en abyme
- Une baguette magique algorithmique

Récursivité

- Une mise en abyme

La “minute xkcd” - Jeu de rôle sur table

http://xkcd.com/244
http://xkcd.free.fr?id=244

Récursivité

- Une mise en abyme

Photo Ethan Clements
http://ethanclements.blogspot.com/2010/07/mise-en-abyme.html

http://www.apprendre-en-ligne.net/blog/index.php/2008/03/29/916-mise-en-abyme

Récursivité

- Une mise en abyme

Photo Ethan Clements
http://ethanclements.blogspot.com/2010/07/mise-en-abyme.html

http://www.apprendre-en-ligne.net/blog/index.php/2008/03/29/916-mise-en-abyme

Comment dessiner ces images ?

Récursivité

- Une mise en abyme

Photo Ethan Clements
http://ethanclements.blogspot.com/2010/07/mise-en-abyme.html

http://www.apprendre-en-ligne.net/blog/index.php/2008/03/29/916-mise-en-abyme

Comment dessiner ces images ?

L'image contient une plus petite version d'elle-même.

Récursivité

- Une mise en abyme

Photo Ethan Clements
http://ethanclements.blogspot.com/2010/07/mise-en-abyme.html

http://www.apprendre-en-ligne.net/blog/index.php/2008/03/29/916-mise-en-abyme

Comment dessiner ces images ?

Pour dessiner la grande image, j'utilise le dessin de l'image en plus petit.

Récursivité

- Une mise en abyme

Qu’est-ce qu’une poupée russe ?

Photo Malix L.
http://www.flickr.com/photos/_malix_/6970987365/

Récursivité

- Une mise en abyme

Qu'est-ce qu'une poupée russe ?

Photo Malix L.
http://www.flickr.com/photos/_malix_/6970987365/

Une **poupée russe** est une poupée qui contient :
- soit rien
- soit une autre **poupée russe** plus petite

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Exemple des factorielles :

factorielle(1) = 1

factorielle(2) = 1x2 = 2

factorielle(3) = 1x2x3 = 6

factorielle(4) = 1x2x3x4 = 24

...

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Exemple des factorielles :

Initialisation :

factorielle(1) = 1

factorielle(2) = 1x2 = 2

Formule d'hérédité :

factorielle(3) = 1x2x3 = 6

factorielle(4) = 1x2x3x4 = 24

...

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Exemple des factorielles :

factorielle(1) = 1

factorielle(2) = 1x2 = 2

factorielle(3) = 1x2x3 = 6

factorielle(4) = 1x2x3x4 = 24

...

Initialisation :
factorielle(1) = 1

Formule d'hérédité :
factorielle(*n*) = ... factorielle(*n*-1) ...

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Exemple des factorielles :

factorielle(1) = 1

factorielle(2) = 1x2 = 2

factorielle(3) = 1x2x3 = 6

factorielle(4) = 1x2x3x4 = 24

...

Initialisation :
factorielle(1) = 1

Formule d'hérédité :
factorielle(*n*) = factorielle(*n*-1) x *n*

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Exemple des factorielles :

factorielle(1) = 1 —x2→

factorielle(2) = 1x2 = 2

factorielle(3) = 1x2x3 = 6

factorielle(4) = 1x2x3x4 = 24

...

Initialisation :
factorielle(1) = 1

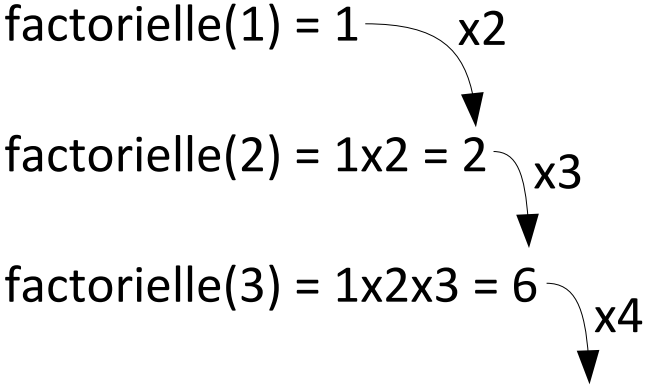
Formule d'hérédité :
factorielle(*n*) = factorielle(*n*-1) x *n*

Récursivité

- Une baguette magique algorithmique

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Exemple des factorielles :



...

Initialisation :
factorielle(1) = 1

Formule d'hérédité :
factorielle(*n*) = factorielle(*n*-1) x *n*

Récursivité

- Une baguette magique algorithmique

Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

Algorithme **factorielle**
Entrées :
Type de sortie :
Variable :
Début

Fin

Récursivité

- Une baguette magique algorithmique

Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

Récursivité

- Une baguette magique algorithmique

Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

La fonction factorielle
s'appelle elle-même !

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

entier

entier

Récursivité

- Une baguette magique algorithmique

Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

version non récursive :

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        resultat ← 1
        Pour i de 2 à n faire :
                resultat ← i x resultat
        FinPour
        renvoyer resultat
Fin
```

Plan du cours 1 – Récursivité et tris

- Introduction à la récursivité
- Traces d'exécution de fonctions récursives
- Les tris
- Le tri par sélection
- Le tri à bulles
- Complexité des tris

Récursivité

- Une baguette magique algorithmique

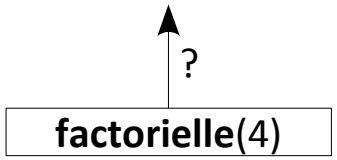
Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

Trace de **factorielle**(4) :



Récursivité

- Une baguette magique algorithmique

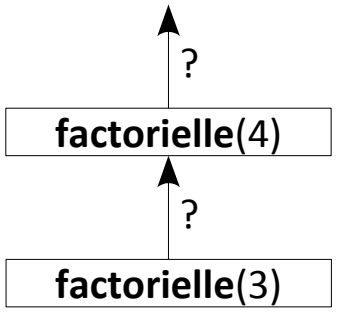
Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

Trace de **factorielle**(4) :



Récursivité

- Une baguette magique algorithmique

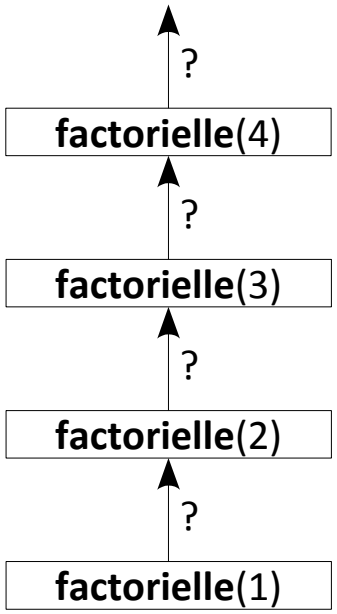
Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

Trace de **factorielle**(4) :



Récursivité

- Une baguette magique algorithmique

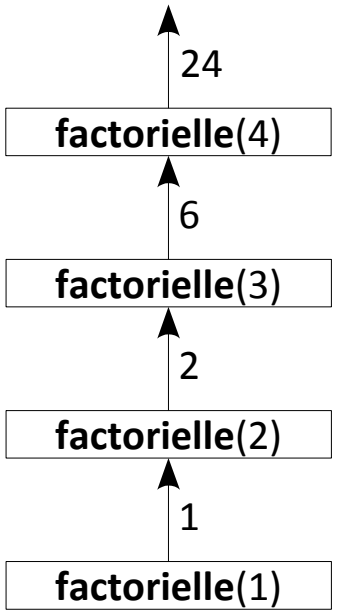
Exemple des factorielles :

- la première factorielle est 1

- pour passer de factorielle(*n*-1) à factorielle(*n*), je **multiplie par *n***

```
Algorithme factorielle
Entrées : entier n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=1 alors :
                resultat ← 1
        sinon :
                resultat ← n x factorielle(n-1)
        Fin si
        renvoyer resultat
Fin
```

Trace de **factorielle**(4) :



Récursivité

Même concept que la **preuve par récurrence**

La “minute mathématique”

Théorème : Je sais monter une échelle

Récursivité

Même concept que la **preuve par récurrence**

La “minute mathématique”

Théorème : Je sais monter une échelle

Démonstration :

Initialisation : je sais monter depuis le sol jusqu'au premier barreau.

Hérédité : si je sais monter jusqu'au (n-1)-ième barreau, je saurai monter jusqu'au n-ième barreau.

Un autre algorithme récursif

La **multiplication**

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Pour calculer $a \times n$:

Initialisation :

Formule d'hérédité :

Exemple :

$a \times 0 = 0$?

$a \times 1 = a$?

$a \times 2 = 2a$?

$a \times 3 = 3a$

Un autre algorithme récursif

La **multiplication**

Si je sais construire le n-ième objet à partir du (n-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Pour calculer $a \times n$:

Initialisation :

$a \times 0 =$

Formule d'hérédité :

$a \times n = \ldots a \times (n\text{-}1) \ldots$

Exemple :

$a \times 0 = 0$?

$a \times 1 = a$?

$a \times 2 = 2a$?

$a \times 3 = 3a$

Un autre algorithme récursif

La **multiplication**

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Pour calculer *a* x *n* :

Initialisation :

a x 0 =

Formule d'hérédité :

a x *n* = … *a* x (*n*-1) …

Exemple :

3 x 0 = 0 +3
3 x 1 = 3 +3
3 x 2 = 6 +3
3 x 3 = 9
3 x 4 = 12

Un autre algorithme récursif

La **multiplication**

Si je sais construire le n-ième objet à partir du (n-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Pour calculer $a \times n$:

Initialisation :

$a \times 0 = 0$

Formule d'hérédité :

$a \times n = a \times (n\text{-}1) + a$

Un autre algorithme récursif

La **multiplication**

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Pour calculer $a \times n$:

Initialisation :

$a \times 0 = 0$

Formule d'hérédité :

$a \times n = a \times (n\text{-}1) + a$

Algorithme **produit**
Entrées :
Type de sortie :
Variable :
Début

Fin

Un autre algorithme récursif

La **multiplication**

Si je sais construire le *n*-ième objet à partir du (*n*-1)-ième objet, et que je sais construire le premier objet, alors je sais construire tous les objets.

Pour calculer *a* x *n* :

Initialisation :

a x 0 = 0

Formule d'hérédité :

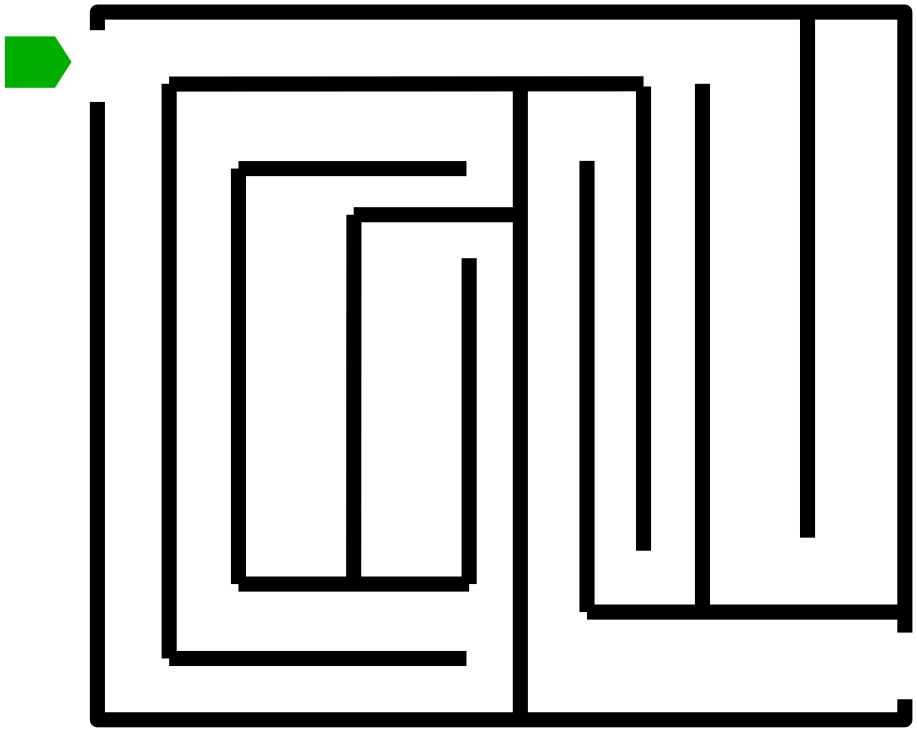
a x *n* = *a* x (*n*-1) + *a*

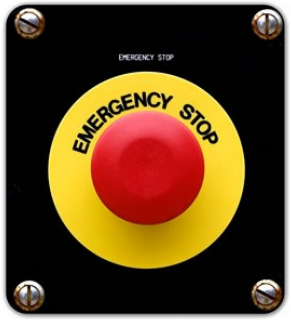
```
Algorithme produit
Entrées : entiers a et n
Type de sortie : entier
Variable : entier resultat
Début
        Si n=0 alors :
                resultat ← 0
        sinon :
                resultat ← produit(a,n-1) + a
        Fin si
        renvoyer resultat
Fin
```

Récursivité

Le labyrinthe et les robots tueurs :

10 personnes dans un labyrinthe poursuivies par des robots tueurs. Si on appuie sur le bouton stop dans les 5 minutes, les robots sont désactivés.

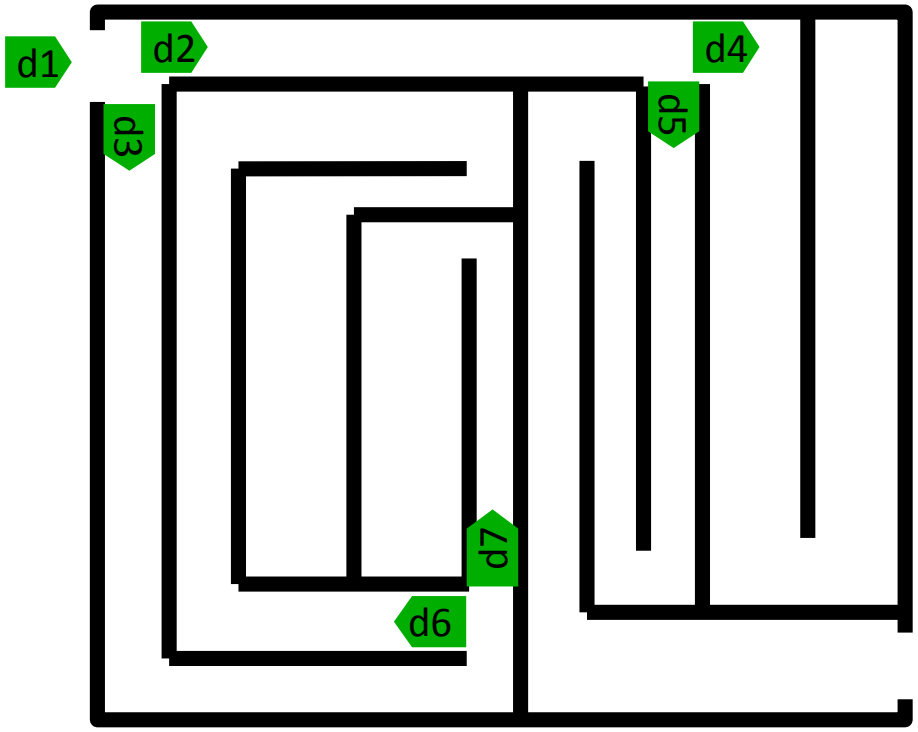




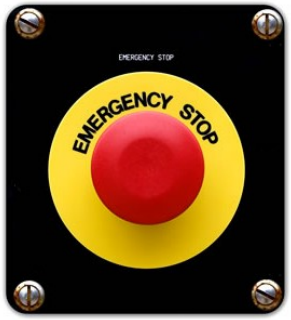
Récursivité

Le labyrinthe et les robots tueurs :

10 personnes dans un labyrinthe poursuivies par des robots tueurs. Si on appuie sur le bouton stop dans les 5 minutes, les robots sont désactivés.



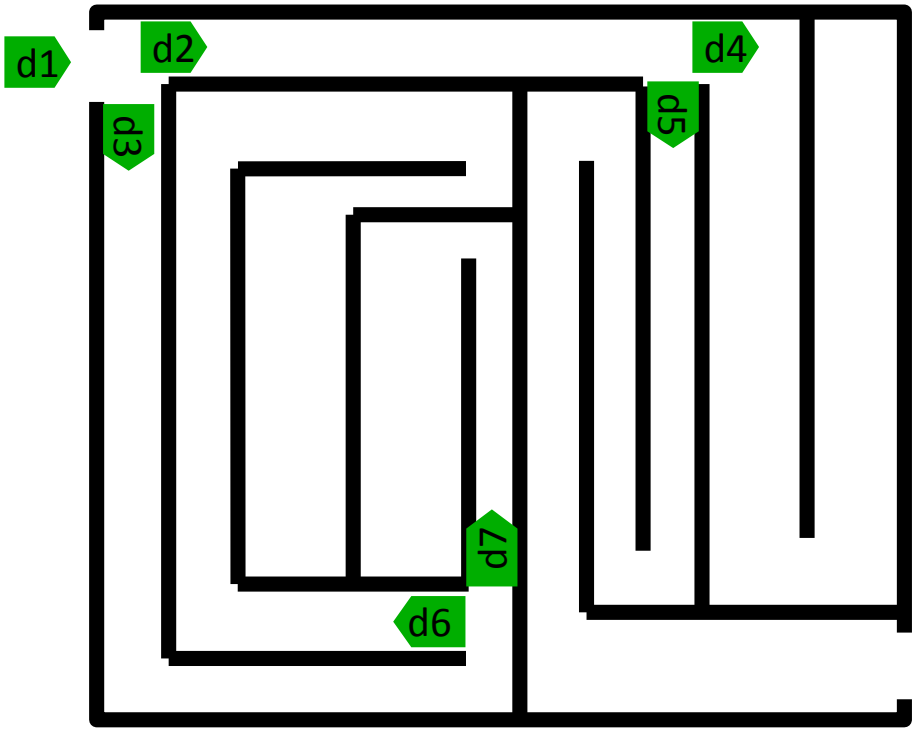
Algorithme **TrouveBouton**
Entrées : entier *nombrePersonnes*,
flottant *tempsRestant*,
chaîne de caractères *direction*
Type de sortie : booléen



Récursivité

Le labyrinthe et les robots tueurs :

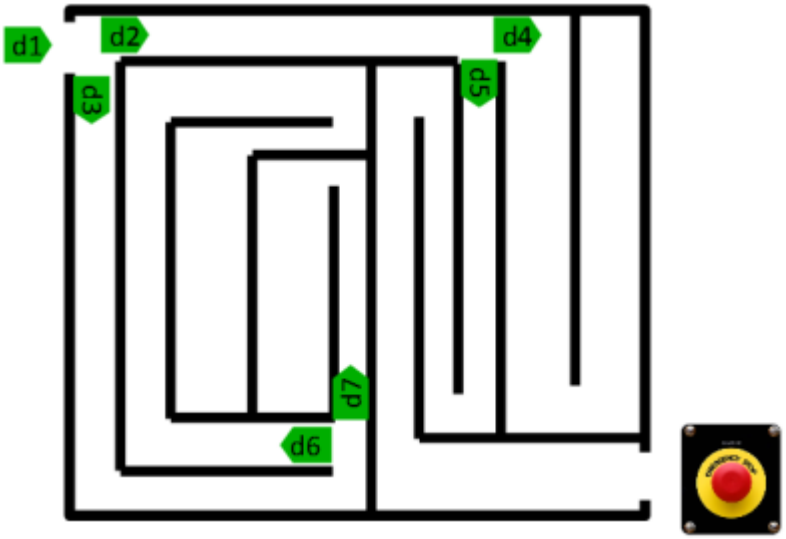
10 personnes dans un labyrinthe poursuivies par des robots tueurs. Si on appuie sur le bouton stop dans les 5 minutes, les robots sont désactivés.



Algorithme **trouveBouton**
Entrées : entier *nombrePersonnes*, flottant *tempsRestant*, chaîne de caractères *direction*
Type de sortie : booléen

Si on trouve le bouton à temps, on appuie dessus, et l'algorithme renvoie VRAI. Si on trouve une intersection, le groupe se sépare en deux. Si on ne trouve pas le bouton à temps, on renvoie FAUX.

Récursivité



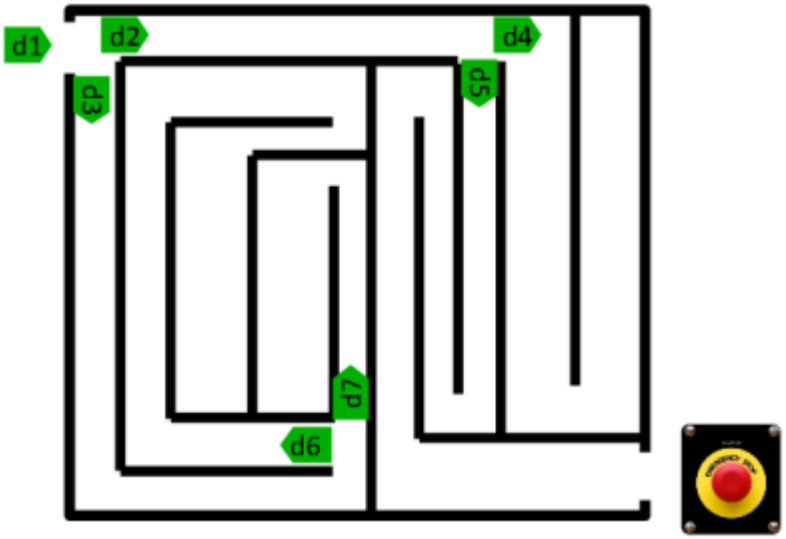
Algorithme **trouveBouton**
Entrées : entier *nombrePersonnes*, flottant *tempsRestant*, chaîne de caractères *direction*
Type de sortie : booléen

Si on trouve le bouton à temps, on appuie dessus, et l'algorithme renvoie VRAI. Si on trouve une intersection, le groupe se sépare en deux. Si on ne trouve pas le bouton à temps, on renvoie FAUX.

Que renvoie **trouveBouton**(10,5,”d1”) ?

Il faut faire la trace de **trouveBouton**(10,5,”d1”).

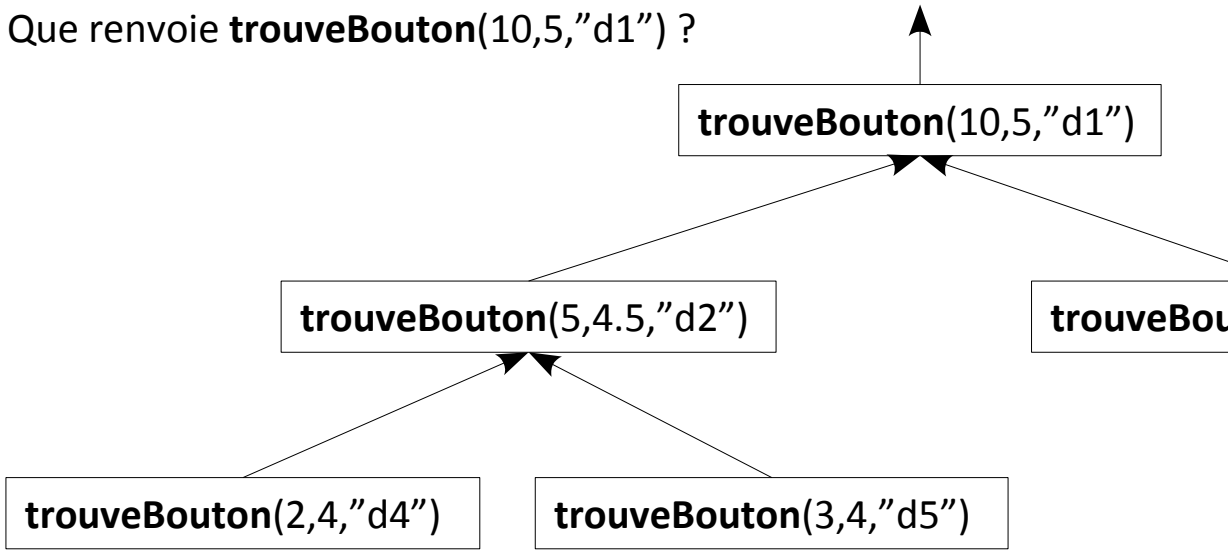
Récursivité



Algorithme **trouveBouton**
Entrées : entier *nombrePersonnes*, flottant *tempsRestant*, chaîne de caractères *direction*
Type de sortie : booléen

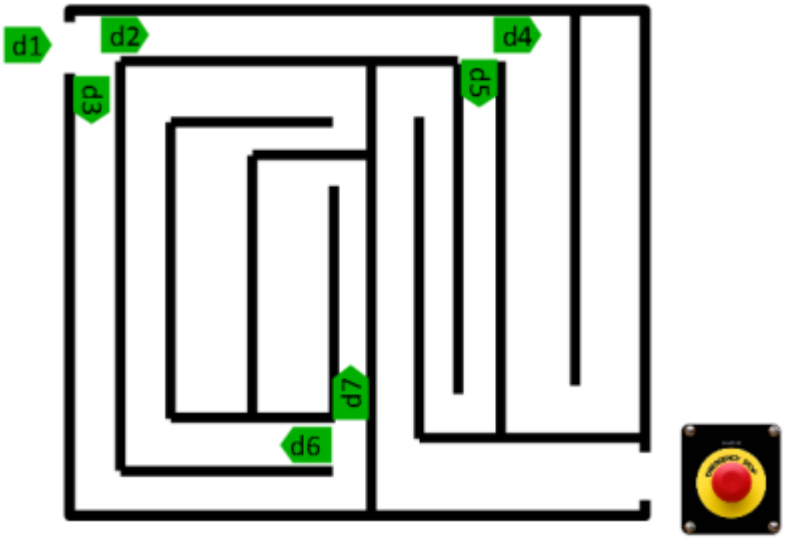
Si on trouve le bouton à temps, on appuie dessus, et l'algorithme renvoie VRAI. Si on trouve une intersection, le groupe se sépare en deux. Si on ne trouve pas le bouton à temps, on renvoie FAUX.

Que renvoie **trouveBouton**(10,5,"d1") ?



Arbre d'exécution de **trouveBouton**(10,5,"d1").

Récursivité

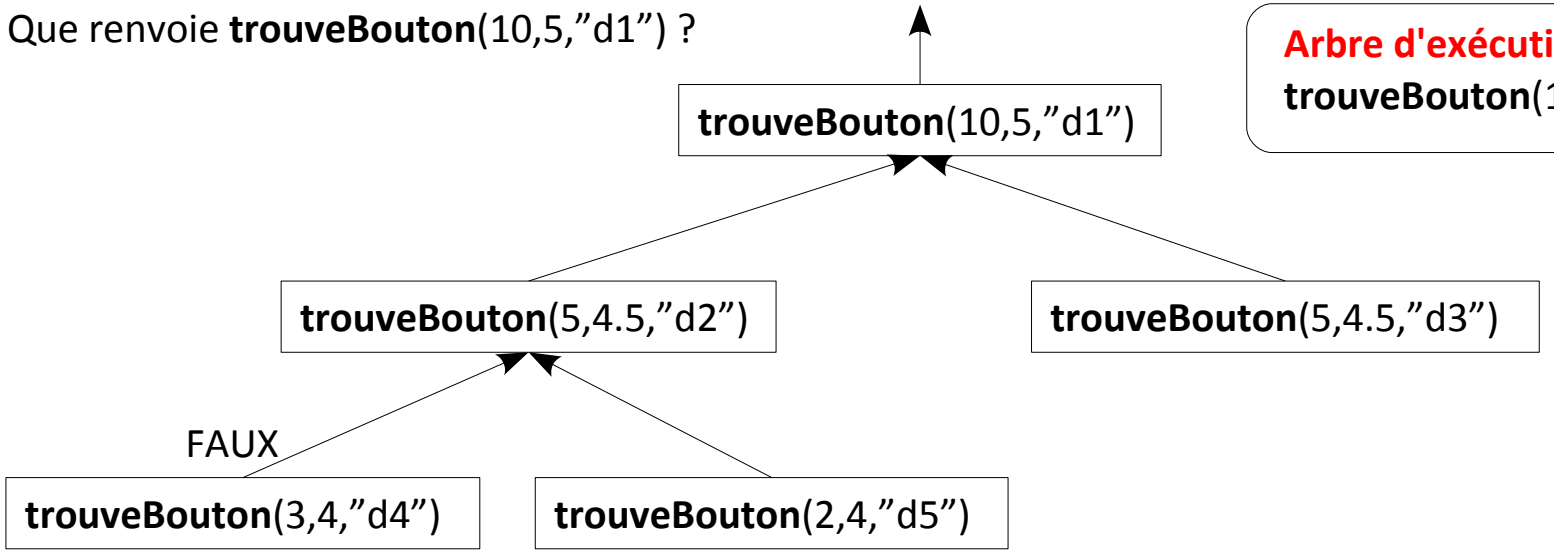


Algorithme **trouveBouton**
Entrées : entier *nombrePersonnes*, flottant *tempsRestant*, chaîne de caractères *direction*
Type de sortie : booléen

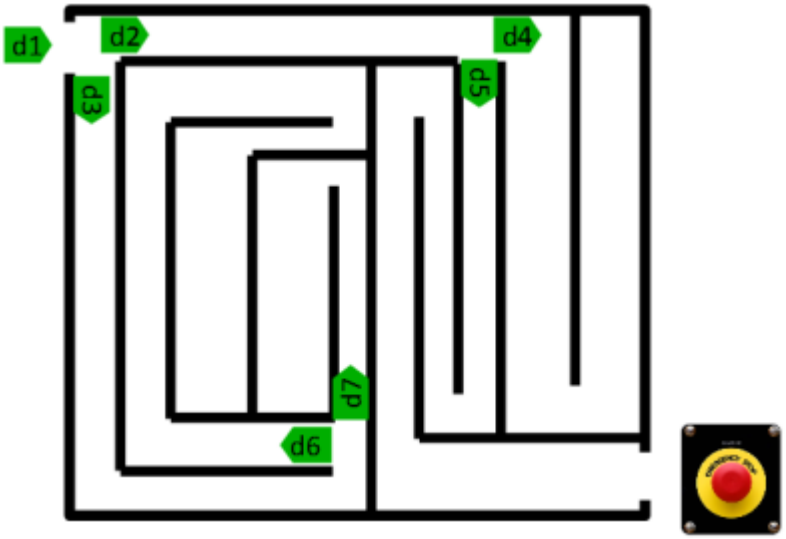
Si on trouve le bouton à temps, on appuie dessus, et l'algorithme renvoie VRAI. Si on trouve une intersection, le groupe se sépare en deux. Si on ne trouve pas le bouton à temps, on renvoie FAUX.

Que renvoie **trouveBouton**(10,5,"d1") ?

Arbre d'exécution de **trouveBouton**(10,5,"d1").



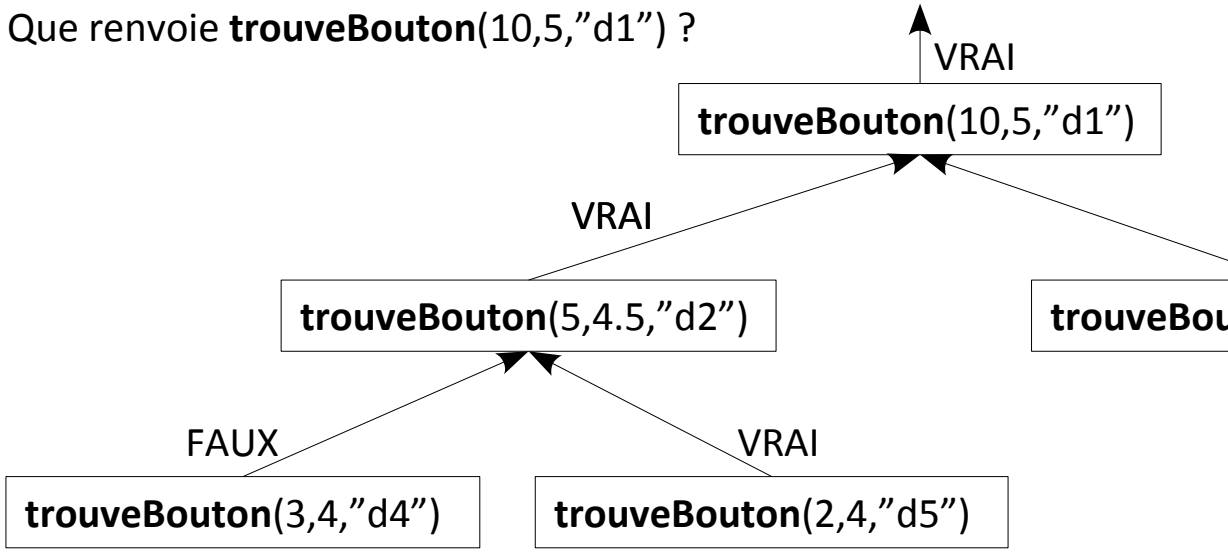
Récursivité



Algorithme **trouveBouton**
Entrées : entier *nombrePersonnes*, flottant *tempsRestant*, chaîne de caractères *direction*
Type de sortie : booléen

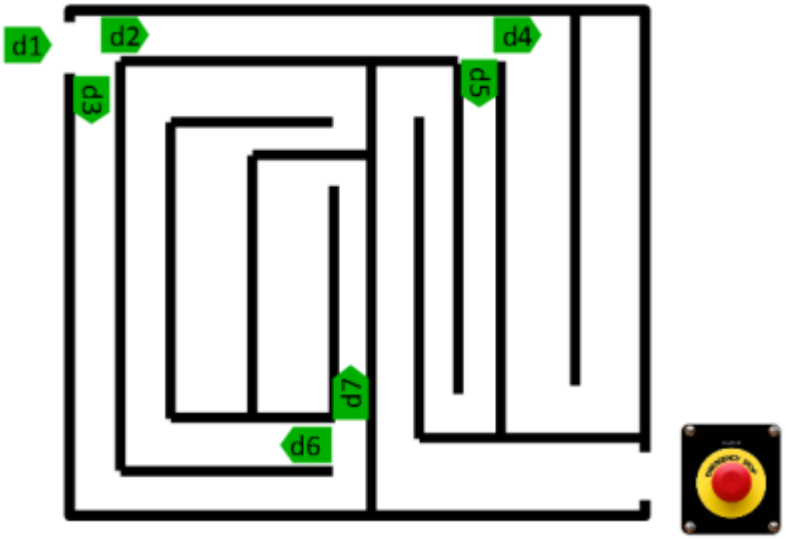
Si on trouve le bouton à temps, on appuie dessus, et l'algorithme renvoie VRAI. Si on trouve une intersection, le groupe se sépare en deux. Si on ne trouve pas le bouton à temps, on renvoie FAUX.

Que renvoie **trouveBouton**(10,5,"d1") ?



Arbre d'exécution de **trouveBouton**(10,5,"d1").

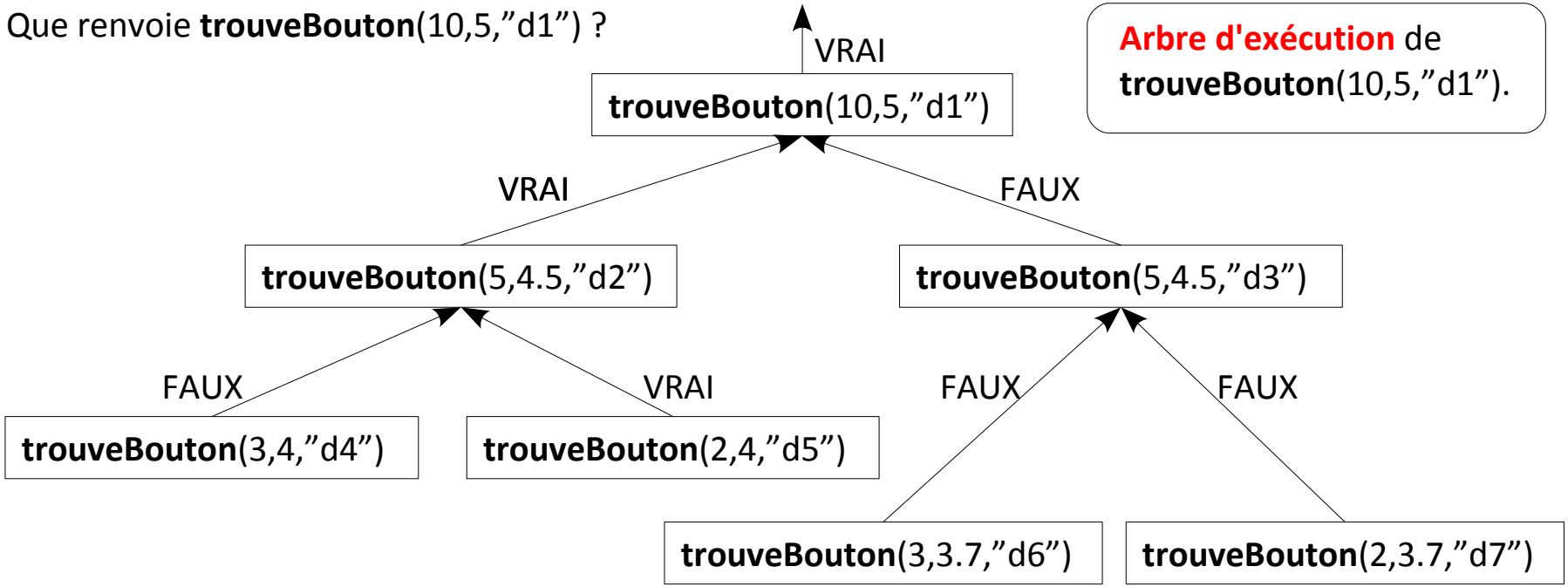
Récursivité



Algorithme **trouveBouton**
Entrées : entier *nombrePersonnes*, flottant *tempsRestant*, chaîne de caractères *direction*
Type de sortie : booléen

Si on trouve le bouton à temps, on appuie dessus, et l'algorithme renvoie VRAI. Si on trouve une intersection, le groupe se sépare en deux. Si on ne trouve pas le bouton à temps, on renvoie FAUX.

Que renvoie **trouveBouton**(10,5,"d1") ?



Arbre d'exécution de **trouveBouton**(10,5,"d1").