Algorithmique des structures arborescentes

L2 Info et Math-info, 2017–18

Marc Zeitoun

15 mars 2018

Organisation

- Aujourd'hui = avant-dernier cours en amphi.
- DS dans 2 semaines (vendredi 30/03/2018)

Organisation

- Aujourd'hui = avant-dernier cours en amphi.
- DS dans 2 semaines (vendredi 30/03/2018)

- Moodle : dépôt des fichiers (plus d'édition).
- Évaluation en 2 temps.
  - Phase 1 : sans évaluation automatique $\Rightarrow$ testez !
  - Phase 2 : réouverture après notation.

Plan

Parcours en profondeur préfixe

Écriture naturelle de façon récursive.

```
DFS(t) :
si t est vide alors
    Retourner
sinon
    Traiter(t.racine)
    DFS(t.fils_gauche)
    DFS(t.fils_droit)
fin si
```

Parcours en profondeur préfixe

Écriture naturelle de façon récursive.

DFS(t) :
si t est vide **alors**
    Retourner
sinon
    Traiter(t.racine)
    DFS(t.fils_gauche)
    DFS(t.fils_droit)
fin si

Comment adapter l'algorithme pour

- générer la **liste** des nœuds dans l'ordre préfixe,
- en complexité $O(n)$, où $n =$ nombre de nœuds ?

Parcours en profondeur préfixe

Écriture naturelle de façon récursive.

DFS(t) :
si t est vide **alors**
    Retourner
sinon
    Traiter(t.racine)
    DFS(t.fils_gauche)
    DFS(t.fils_droit)
fin si

Comment adapter l'algorithme pour

- générer la **liste** des nœuds dans l'ordre préfixe,
- en complexité $O(n)$, où $n =$ nombre de nœuds ?

À éviter : Concaténations de listes $\rightsquigarrow O(n^2)$.

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop

Push(1)

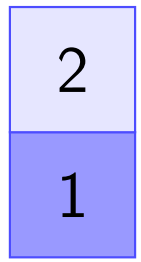
Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop



Push(1) Push(2)

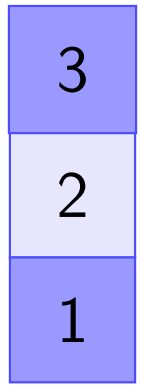
Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop



Push(1) Push(2) Push(3)

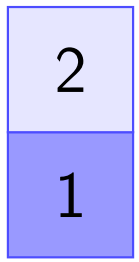
Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop



Push(1) Push(2) Push(3) Pop

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

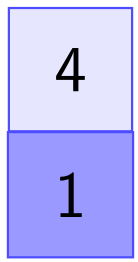
Rappel sur les piles. 2 opérations : Push(x) et Pop

Push(1) Push(2) Push(3) Pop Pop

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.
  $\rightsquigarrow$ concaténation de listes en $O(1)$.
- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop



Push(1) Push(2) Push(3) Pop Pop Push(4)

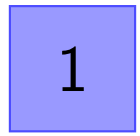
Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop



Push(1) Push(2) Push(3) Pop Pop Push(4) Pop

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.
  $\rightsquigarrow$ concaténation de listes en $O(1)$.
- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop

Push(1) Push(2) Push(3) Pop Pop Push(4) Pop Pop

Parcours en profondeur préfixe

- 1re solution : utiliser des listes doublement chaînées.

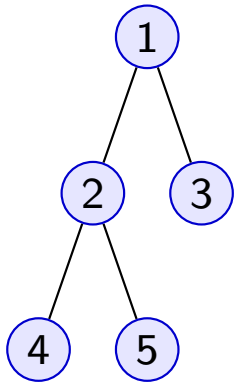
  $\rightsquigarrow$ concaténation de listes en $O(1)$.

- 2e solution : maintenir les sous-arbres non traités dans **une pile**.

Rappel sur les piles. 2 opérations : Push(x) et Pop

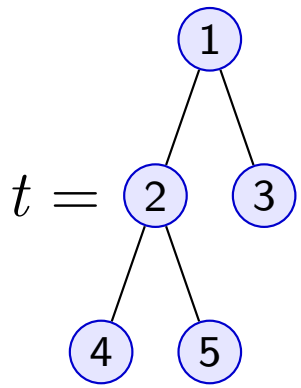
Pour le parcours, ce sont des sous-arbres qu'on mémorise

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
		t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);

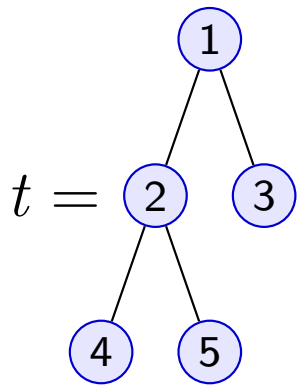
Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
		t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);



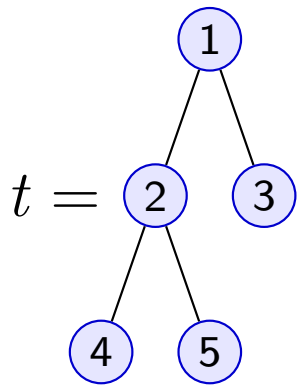
Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	1	Accumuler(t.racine); Push(t.right); Push(t.left);

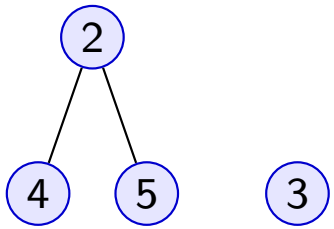


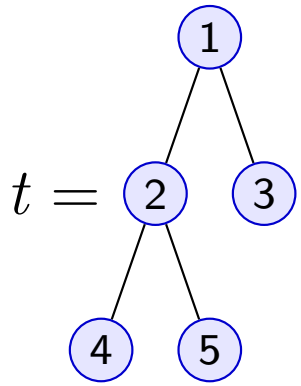
Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	1	
3		Push(t.right); Push(t.left);

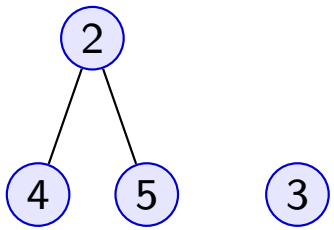


Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	1	
		Push(t.left);

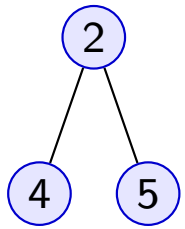


Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	1	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);

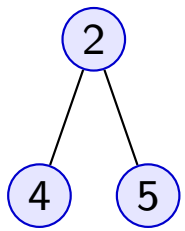
Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	1	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);
3		

$t =$ 

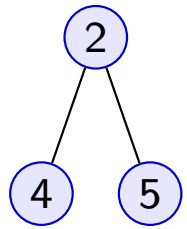
Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	2, **1**	Accumuler(t.racine); Push(t.right); Push(t.left);
3		

$t =$ 

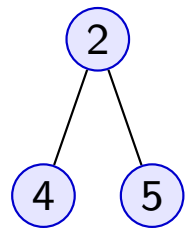
Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	2, **1**	
5 3		Push(t.right); Push(t.left);

$t =$ 

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	2, **1**	
4 5 3		Push(t.left);

$t =$ 

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	2, 1	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);
4 5 3		

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	2, **1**	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);
5 3		

$t =$ 4

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	4, **2**, **1**	Accumuler(t.racine); Push(t.right); Push(t.left);
5 3		

$t =$ 4

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	4, **2**, **1**	
⑤ ③		Push(t.right); Push(t.left);

$t =$ ④

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	4, **2**, **1**	
5 3		Push(t.left);

$t =$ 4

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	4, 2, 1	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);
5 3		

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	4, **2**, **1**	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);
3		

$t =$ 5

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	5, **4**, **2**, **1**	Accumuler(t.racine); Push(t.right); Push(t.left);
③		

$t =$ ⑤

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	5, **4**, **2**, **1**	
③		Push(t.right); Push(t.left);

$t =$ ⑤

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	5, **4**, **2**, **1**	
③		Push(t.left);

$t =$ ⑤

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	5, **4**, **2**, **1**	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);
3		

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	5, **4**, **2**, **1**	t = Pop(); Accumuler(t.racine); Push(t.right); Push(t.left);

$t =$ (3)

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	3, **5**, **4**, **2**, **1**	Accumuler(t.racine); Push(t.right); Push(t.left);

$t =$ 

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	3, **5**, **4**, **2**, **1**	Push(t.right); Push(t.left);

$t =$ 

Parcours en profondeur préfixe

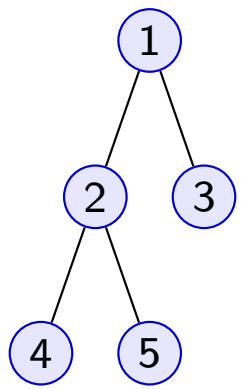
Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	3, **5**, **4**, **2**, **1**	
		Push(t.left);

$t =$ 

Parcours en profondeur préfixe

Pile d'arbres (haut de pile à gauche)	Accumulateur	Actions
	3, **5**, **4**, **2**, **1**	

Parcours en profondeur préfixe : 1, 2, 4, 5, 3



Parcours en profondeur : programmation

- On peut utiliser une liste pour simuler la pile.
- Les opérations Push et Pop s'implémentent en $O(1)$.

Parcours en profondeur : programmation

- On peut utiliser une liste pour simuler la pile.
- Les opérations Push et Pop s'implémentent en $O(1)$.
- Si la pile contient `t::fin` où `t=(Bin(x,left,right))` on retourne `DFS (left::right::fin) (x::acc)`

Parcours en profondeur : programmation

- On peut utiliser une liste pour simuler la pile.
- Les opérations Push et Pop s'implémentent en $O(1)$.
- Si la pile contient `t::fin` où `t=(Bin(x,left,right))` on retourne `DFS (left::right::fin) (x::acc)`
- La pile passe de `t::fin` à `left::right::fin`. On a :
  1. Dépilé `t`,
  2. puis empilé `right`,
  3. puis empilé `left`.

Autres parcours en profondeur : exercice

Adapter l'algorithme pour les parcours en profondeur **infixe**.

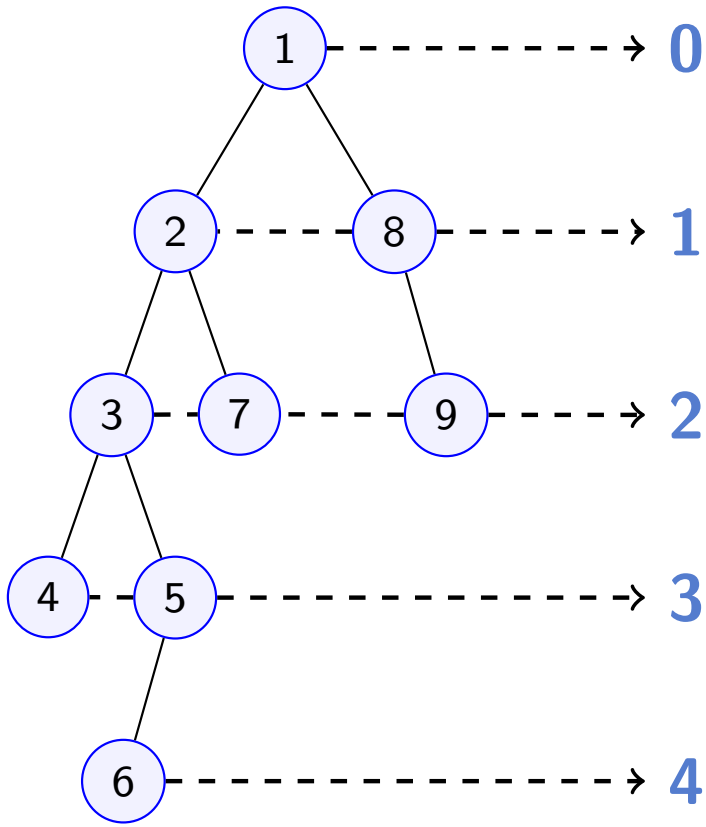
Parcours en largeur (BFS, Breadth-First Search)

Ce type de parcours visite les nœuds par **profondeurs croissantes**.

- On visite d'abord les nœuds à profondeur 0 (= la racine),
- Puis les nœuds à profondeur 1 (= les fils de la racine),
- Puis les nœuds à profondeur 2,
- etc.

Parcours en largeur : exemple

Visite des nœuds par **profondeurs croissantes**.



Parcours en largeur : 1 2 8 3 7 9 4 5 6

Parcours en largeur

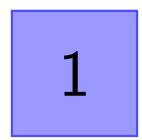
- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

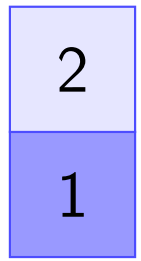


Enqueue(1)

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

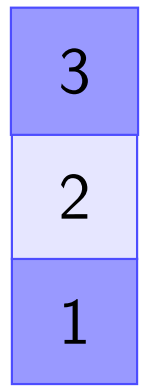


Enqueue(1) Enqueue(2)

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

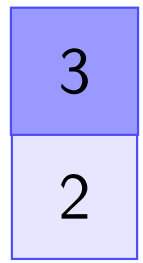


Enqueue(1) Enqueue(2) Enqueue(3)

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue



Enqueue(1) Enqueue(2) Enqueue(3) Dequeue

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

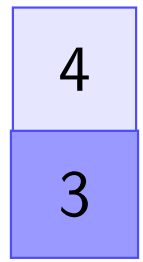
Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

Enqueue(1) Enqueue(2) Enqueue(3) Dequeue Dequeue

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue



Enqueue(1) Enqueue(2) Enqueue(3) Dequeue Dequeue Enqueue(4)

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

Enqueue(1) Enqueue(2) Enqueue(3) Dequeue Dequeue Enqueue(4) Dequeue

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

Enqueue(1) Enqueue(2) Enqueue(3) Dequeue Dequeue Enqueue(4) Dequeue Dequeue

Parcours en largeur

- Même algorithme en remplaçant la **pile** par une **file**.

Rappel sur les files. 2 opérations : Enqueue(x) et Dequeue

Pour le parcours, ce sont des sous-arbres qu'on mémorise

Parcours en largeur

Parcours en largeur : programmation

- But : implémenter **Enqueue**, **Dequeue** en temps $O(1)$.
  $\rightsquigarrow$ soit utiliser une liste doublement chaînée,
  $\rightsquigarrow$ soit simuler la file par **2 piles**.

Parcours en largeur : programmation

- But : implémenter **Enqueue**, **Dequeue** en temps $O(1)$.
  $\rightsquigarrow$ soit utiliser une liste doublement chaînée,
  $\rightsquigarrow$ soit simuler la file par **2 piles**.
- Si la file contient `t::fin` où `t=(Bin(x,left,right))`
  on voudrait retourner `BFS (fin @ [left;right]) (x::acc)`

Parcours en largeur : programmation

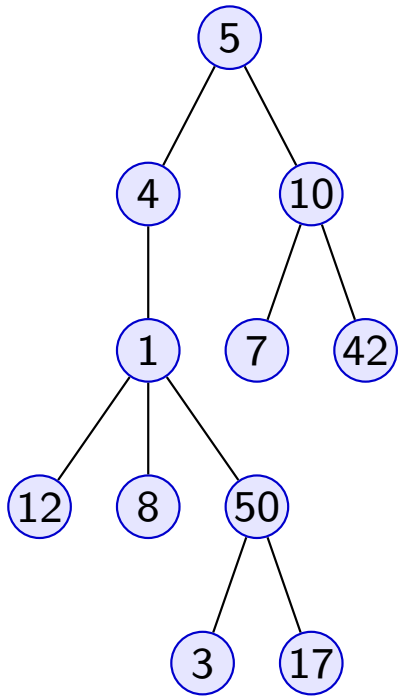
- But : implémenter **Enqueue**, **Dequeue** en temps $O(1)$.
  $\rightsquigarrow$ soit utiliser une liste doublement chaînée,
  $\rightsquigarrow$ soit simuler la file par **2 piles**.
- Si la file contient `t::fin` où `t=(Bin(x,left,right))`
  on voudrait retourner `BFS (fin @ [left;right]) (x::acc)`
- Non efficace à cause de la concaténation `@`.
- Implémentation efficace ? Simulation d'une file par **2 piles**.

Plan

Arbres planaires

Définition informelle : arbres dans lesquels

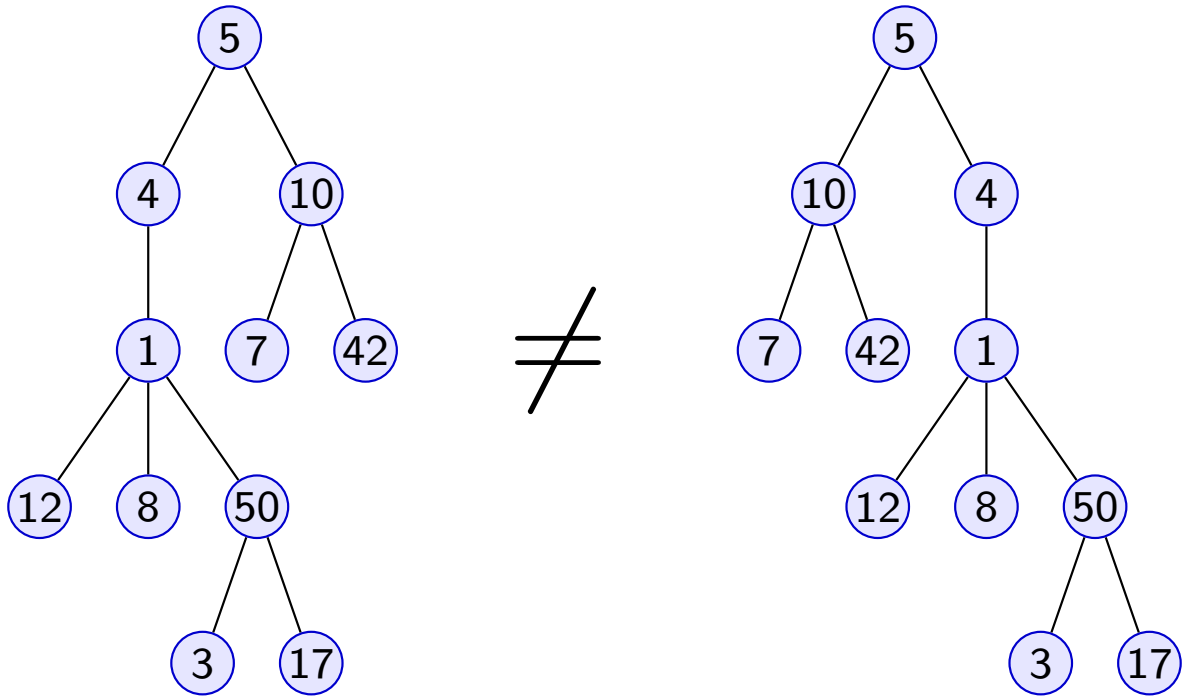
- l'arité n'est pas contrainte.
- l'ordre des fils à une importance.



Arbres planaires

Définition informelle : arbres dans lesquels

- l'arité n'est pas contrainte.
- l'ordre des fils à une importance.



Arbres planaires

Définition informelle : arbres dans lesquels

- l'arité n'est pas contrainte.
- l'ordre des fils à une importance.

Naturels (ex. arbres de décision en IA, arbres de récursion).

Arbres planaires

Définition informelle : arbres dans lesquels

- l'arité n'est pas contrainte.
- l'ordre des fils à une importance.

Naturels (ex. arbres de décision en IA, arbres de récursion).

Définition récursive des arbres planaires

- Constitué d'un nœud, et

Arbres planaires

Définition informelle : arbres dans lesquels

- l'arité n'est pas contrainte.
- l'ordre des fils à une importance.

Naturels (ex. arbres de décision en IA, arbres de récursion).

Définition récursive des arbres planaires

- Constitué d'un nœud, et
- d'une liste d'arbres planaires.

Arbres planaires

Définition informelle : arbres dans lesquels

- l'arité n'est pas contrainte.
- l'ordre des fils à une importance.

Naturels (ex. arbres de décision en IA, arbres de récursion).

Définition récursive des arbres planaires

- Constitué d'un nœud, et
- d'une liste d'arbres planaires.

Note. Cette définition exclut l'arbre vide.

Plan

Énumération (de squelettes) d'arbres

Combien y a-t-il de squelettes

- d'arbres binaires à n nœuds ?
- d'arbres binaires pleins à n nœuds ?
- d'arbres planaires à n nœuds ?

Énumération (de squelettes) d'arbres

Combien y a-t-il de squelettes

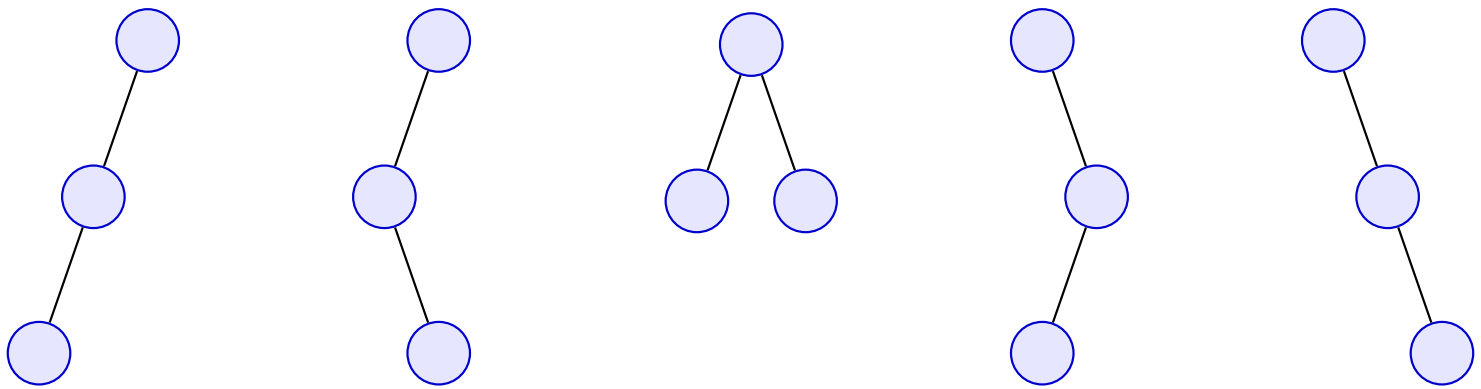
- d'arbres binaires à n nœuds ?
- d'arbres binaires pleins à n nœuds ?
- d'arbres planaires à n nœuds ?

Intérêt de ces questions

- représentation différentes de mêmes objets,
- génération aléatoire,
- analyse d'algorithmes **en moyenne**.

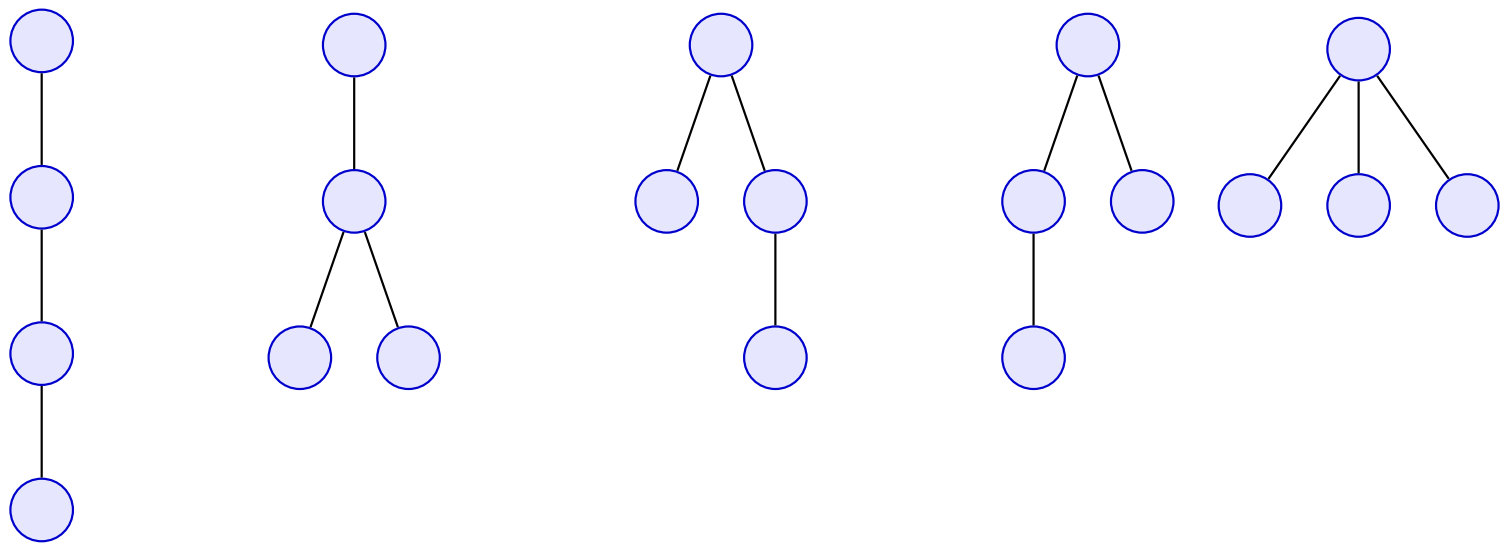
Énumération d'arbres binaires

Combien y a -t-il de squelettes d'arbres **binaires à 3 nœuds** ?



Énumération d'arbres planaires

Combien y a -t-il de squelettes d'arbres **planaires à 4 nœuds** ?



Comptage

Il y a autant

- de squelettes d'arbres binaires à n nœuds.
- de squelettes d'arbres binaires pleins à $2n+1$ nœuds.
- de squelettes d'arbres planaires à $n+1$ nœuds.
- de mots de Dyck de longueur $2n$.
- de chemins de Dyck de longueur $2n$.

Comptage

Il y a autant

- de squelettes d'arbres binaires à n nœuds.
- de squelettes d'arbres binaires pleins à $2n+1$ nœuds.
- de squelettes d'arbres planaires à $n+1$ nœuds.
- de mots de Dyck de longueur $2n$.
- de chemins de Dyck de longueur $2n$.

Leur nombre est

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Nombres de Catalan [Lien 1,2] : 1, 1, 2, 5, 14, 42, 132, 429, 1430, ...

Comptage

Il y a autant

- de squelettes d'arbres binaires à n nœuds.
- de squelettes d'arbres binaires pleins à $2n+1$ nœuds.
- de squelettes d'arbres planaires à $n+1$ nœuds.
- de mots de Dyck de longueur $2n$.
- de chemins de Dyck de longueur $2n$.

Leur nombre est

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Nombres de Catalan [Lien 1,2] : 1, 1, 2, 5, 14, 42, 132, 429, 1430, ...

Il y a des **bijections simples** entre ces ensembles.

Mots de Dyck

Un **mot de Dyck** est une suite de symboles $\uparrow, \downarrow$ tq :

1. Il y a autant de $\uparrow$ que de $\downarrow$.
2. Quand on lit la suite de gauche à droite, il y a toujours au moins autant de $\uparrow$ que de $\downarrow$.

Exemple

- $\uparrow\uparrow\downarrow\downarrow$
- $\uparrow\downarrow\uparrow\downarrow$

Mots de Dyck

Un **mot de Dyck** est une suite de symboles $\uparrow, \downarrow$ tq :

1. Il y a autant de $\uparrow$ que de $\downarrow$.
2. Quand on lit la suite de gauche à droite, il y a toujours au moins autant de $\uparrow$ que de $\downarrow$.

Exemple

- $\uparrow\uparrow\downarrow\downarrow$ $\quad(())$
- $\uparrow\downarrow\uparrow\downarrow$ $\quad()()$

Un mot de Dyck est aussi appelé mot bien parenthésé.

Mots de Dyck

Un **mot de Dyck** est une suite de symboles $\uparrow, \downarrow$ tq :

1. Il y a autant de $\uparrow$ que de $\downarrow$.
2. Quand on lit la suite de gauche à droite, il y a toujours au moins autant de $\uparrow$ que de $\downarrow$.

Exemple

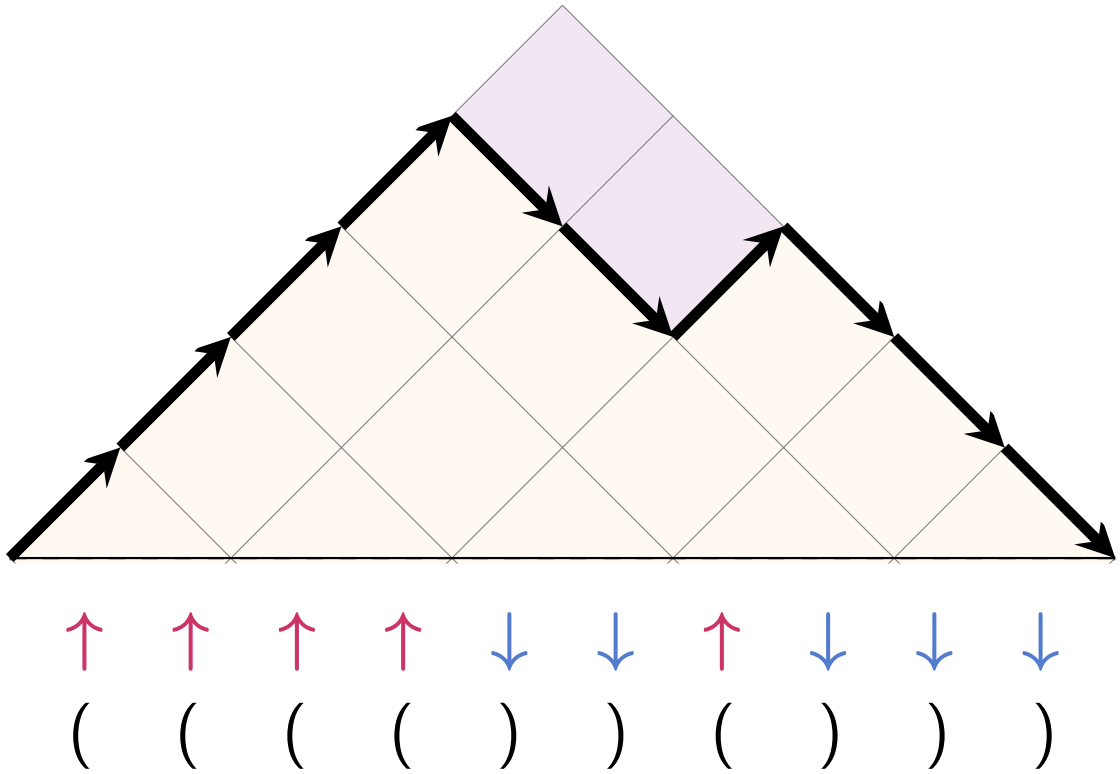
- $\uparrow\uparrow\downarrow\downarrow$ (())
- $\uparrow\downarrow\uparrow\downarrow$ ()()

Un mot de Dyck est aussi appelé mot bien parenthésé.

1. Autant d'ouvrantes que de fermantes.
2. Jamais plus de fermantes que d'ouvrantes quand on lit de gauche à droite.

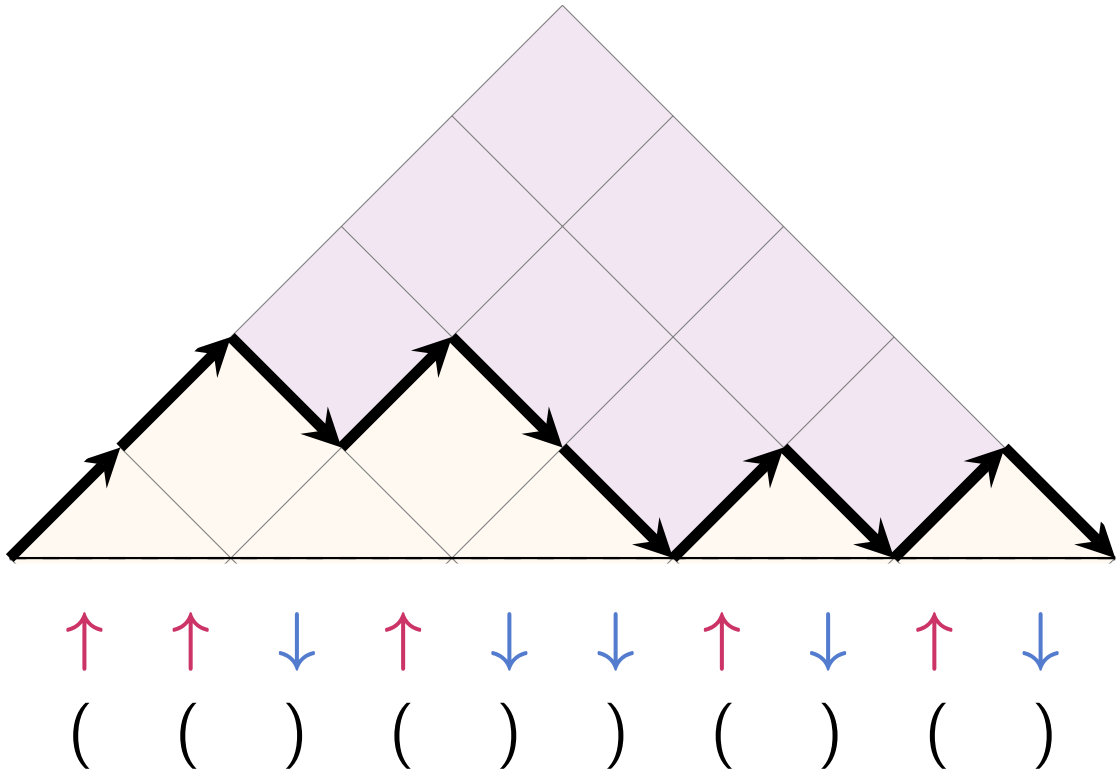
Chemins de Dyck

Simplement une représentation graphique des mots de Dyck.



Chemins de Dyck

Simplement une représentation graphique des mots de Dyck.

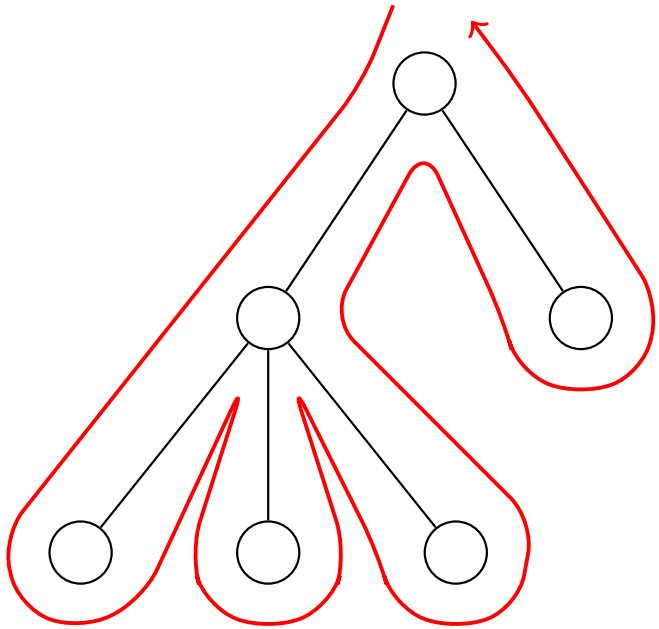


Chemins de Dyck

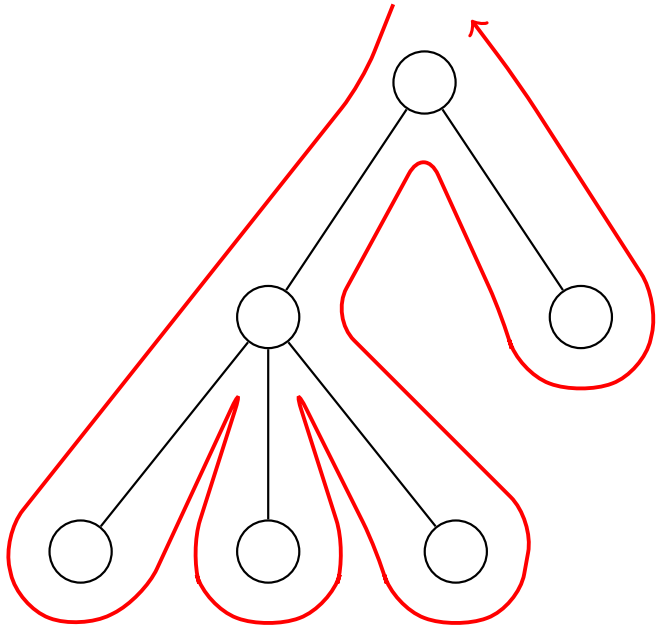
Ces chemins

- Partent de l'origine (0,0),
- 2 types de pas : haut ou bas.
- Restent au dessus de l'altitude 0.
- Terminent à l'altitude 0.

Bijection arbres planaires $\leftrightarrow$ chemins de Dyck

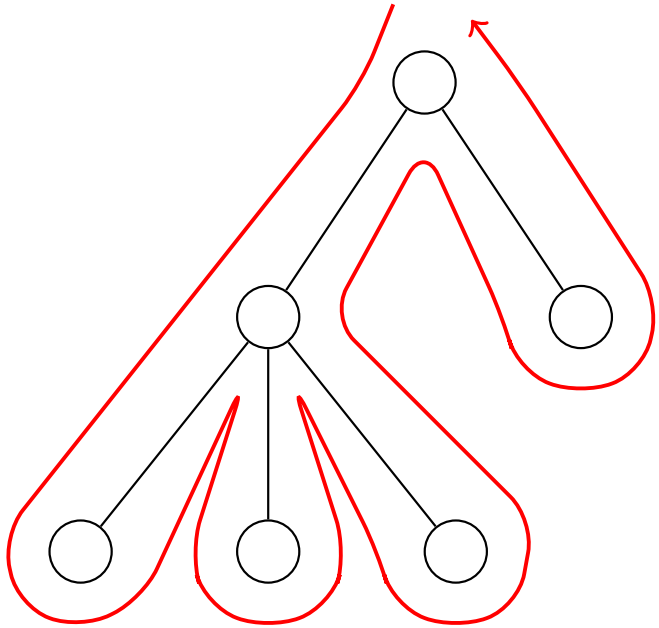


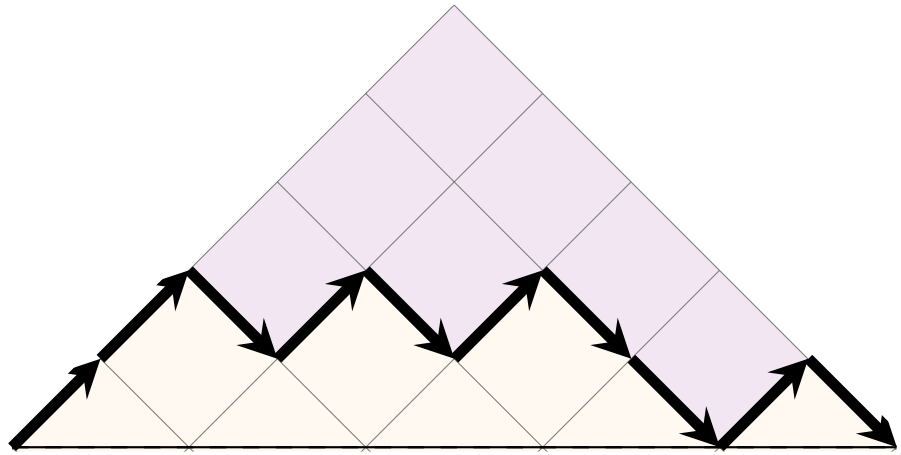
Bijection arbres planaires $\leftrightarrow$ chemins de Dyck



- Parcours en profondeur de l'arbre.
- On produit (lorsqu'on descend et) lorsqu'on monte.

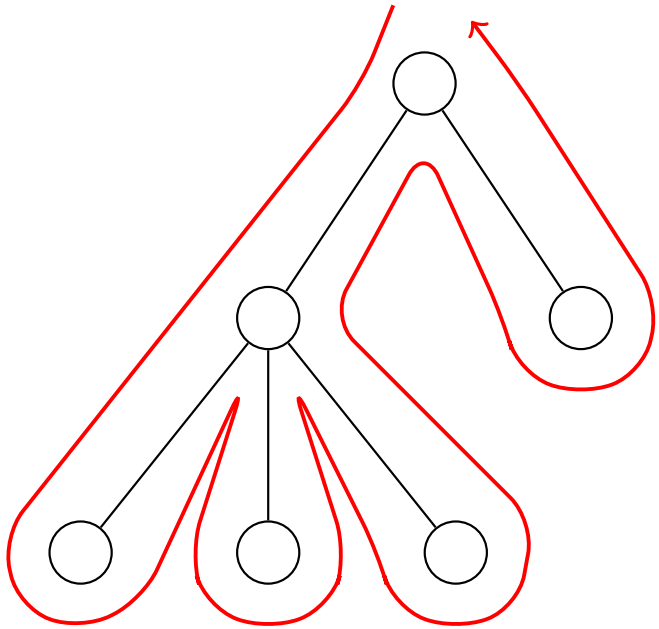
Bijection arbres planaires $\leftrightarrow$ chemins de Dyck

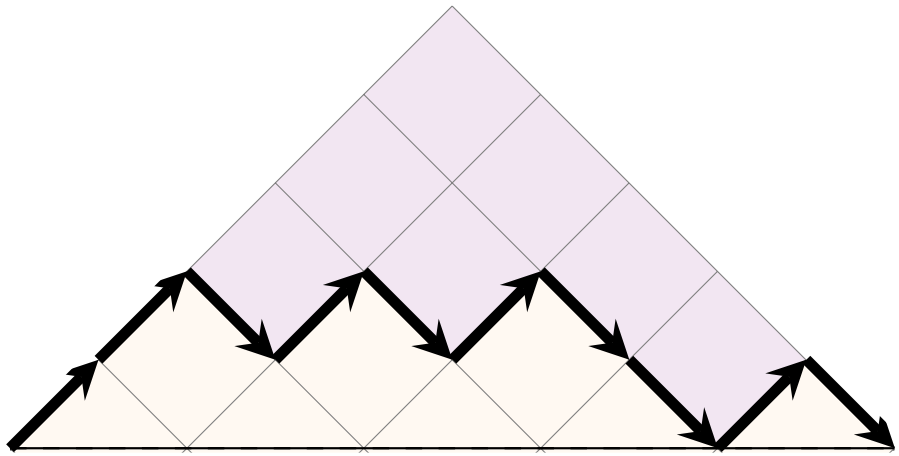




- Parcours en profondeur de l'arbre.
- On produit (lorsqu'on descend et) lorsqu'on monte.

Bijection arbres planaires $\leftrightarrow$ chemins de Dyck





C'est une **bijection**

- Chaque arbre donne un chemin différent,
- Chaque chemin est obtenu depuis un arbre.

Arbres planaires représentés par arbres binaires

Représentation simple d'un arbre planaire : associer à chaque nœud

- son fils gauche,
- son frère droit.

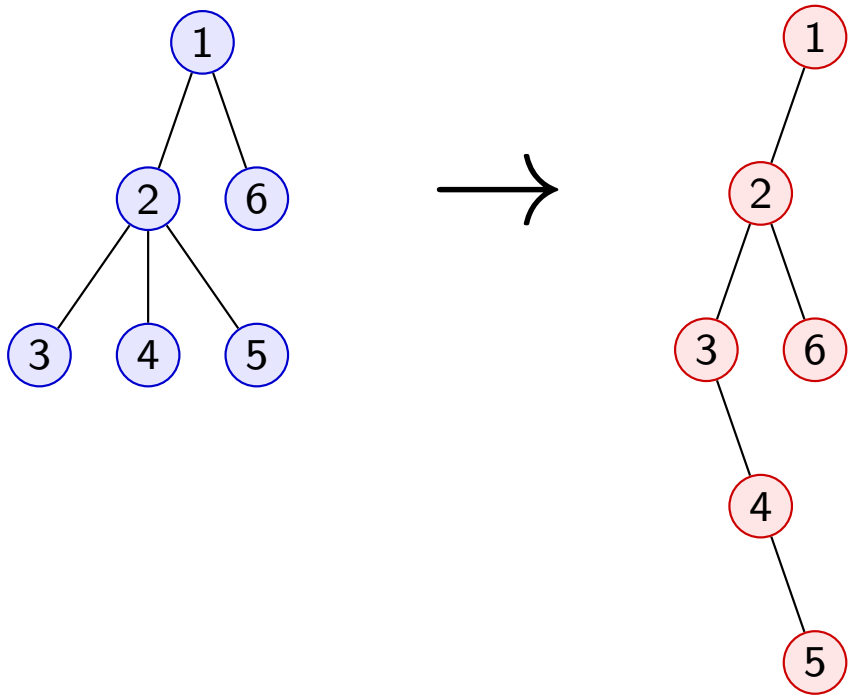
Par exemple, en C, on pourrait définir

```
struct planar_tree
{
  int key;
  struct planar_tree *left_child, *next_brother;
};
```

Ce codage donne l'idée d'une bijection.

Arbres planaires représentés par arbres binaires

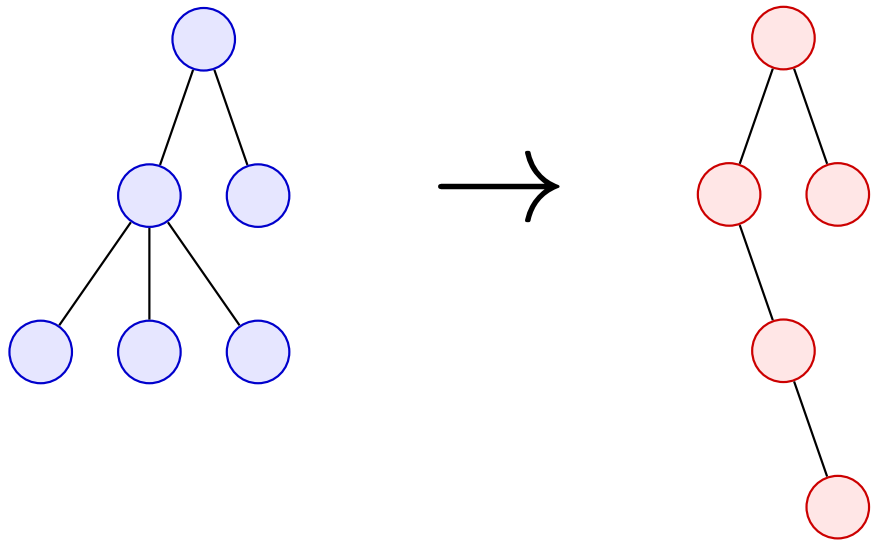
On donne comme fils d'un nœud dans l'arbre binaire son fils gauche et son frère droit dans l'arbre planaire.



La racine de l'arbre binaire obtenu n'a jamais de fils droit.

Arbres planaires représentés par arbres binaires

Pour obtenir une bijection, entre arbres planaires à $n+1$ nœuds et arbres binaires à n nœuds, on supprime la racine de l'arbre obtenu.



Nombre de chemins de Dyck de taille $2n$

- Il y a chemins de taille $2n$ allant de (0,0) à (0,0).

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).
- On doit retrancher les chemins descendant sous l'altitude 0.

  $\Rightarrow$ Combien de chemins descendent sous l'altitude 0 ?

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).
- On doit retrancher les chemins descendant sous l'altitude 0.

  $\Rightarrow$ Combien de chemins descendent sous l'altitude 0 ?

  Autant que de chemins de taille $2n$ allant de (0,0) à $(0,-2)$.

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).
- On doit retrancher les chemins descendant sous l'altitude 0.

  $\Rightarrow$ Combien de chemins descendent sous l'altitude 0 ?

  Autant que de chemins de taille $2n$ allant de (0,0) à $(0,-2)$.

Bijection entre

- chemins de taille $2n$ allant de (0,0) à $(0,-2)$.
- chemins de taille $2n$ allant de (0,0) à (0,0) en passant sous l'altitude 0.

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).
- On doit retrancher les chemins descendant sous l'altitude 0.

  $\Rightarrow$ Combien de chemins descendent sous l'altitude 0 ?

  Autant que de chemins de taille $2n$ allant de (0,0) à $(0,-2)$.

Bijection entre

- chemins de taille $2n$ allant de (0,0) à $(0,-2)$.
- chemins de taille $2n$ allant de (0,0) à (0,0) en passant sous l'altitude 0.

- Il y a tels chemins (choix des $n+1$ descentes).

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).
- On doit retrancher les chemins descendant sous l'altitude 0.

  $\Rightarrow$ Combien de chemins descendent sous l'altitude 0 ?

  Autant que de chemins de taille $2n$ allant de (0,0) à $(0,-2)$.

Bijection entre

- chemins de taille $2n$ allant de (0,0) à $(0,-2)$.
- chemins de taille $2n$ allant de (0,0) à (0,0) en passant sous l'altitude 0.

- Il y a $\binom{2n}{n+1}$ tels chemins (choix des $n+1$ descentes).

Nombre de chemins de Dyck de taille $2n$

- Il y a $\binom{2n}{n}$ chemins de taille $2n$ allant de (0,0) à (0,0).
- On doit retrancher les chemins descendant sous l'altitude 0.

  $\Rightarrow$ Combien de chemins descendent sous l'altitude 0 ?

  Autant que de chemins de taille $2n$ allant de (0,0) à $(0,-2)$.

Bijection entre

- chemins de taille $2n$ allant de (0,0) à $(0,-2)$.
- chemins de taille $2n$ allant de (0,0) à (0,0) en passant sous l'altitude 0.

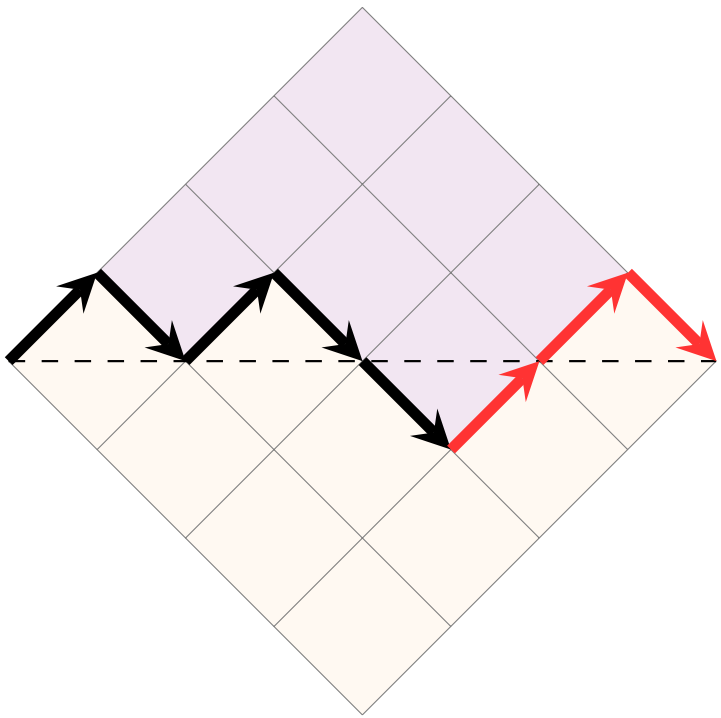
- Il y a $\binom{2n}{n+1}$ tels chemins (choix des $n+1$ descentes).
- Le nombre de chemins de Dyck de taille $2n$ est donc

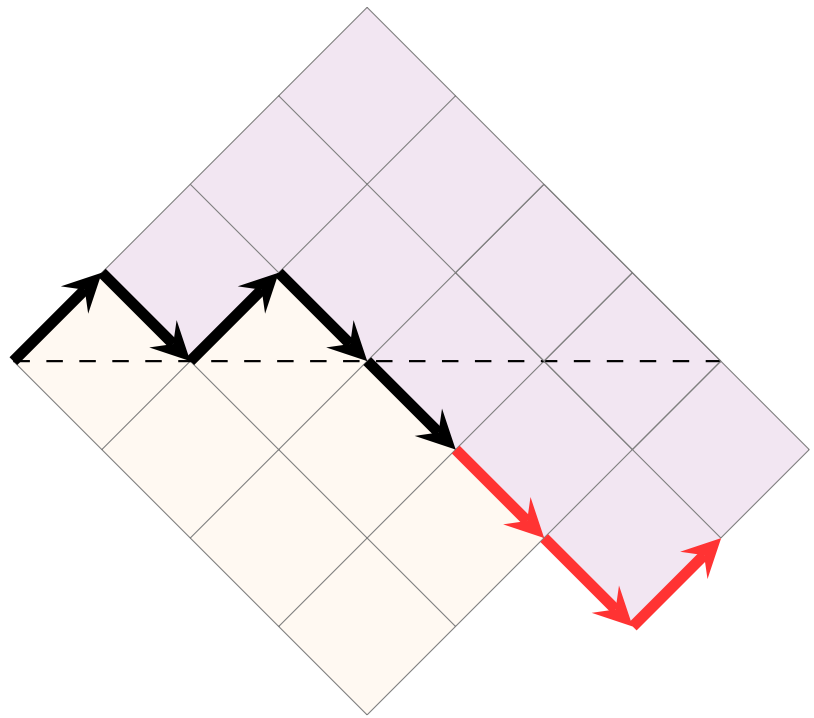
$$\binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n}.$$

Chemins descendant sous l'altitude 0

Bijection : inversion des pas à partir du premier point d'altitude -1.

Exemple 1





Chemins descendant sous l'altitude 0

Bijection : inversion des pas à partir du premier point d'altitude -1.

Exemple 2