Algorithmes et structures de données

Aous Karoui
aous.karoui@univ-lemans.fr

Esteban Loiseau
esteban.loiseau@univ-lemans.fr

Cours 2
Comprendre la récursivité

Récursivité

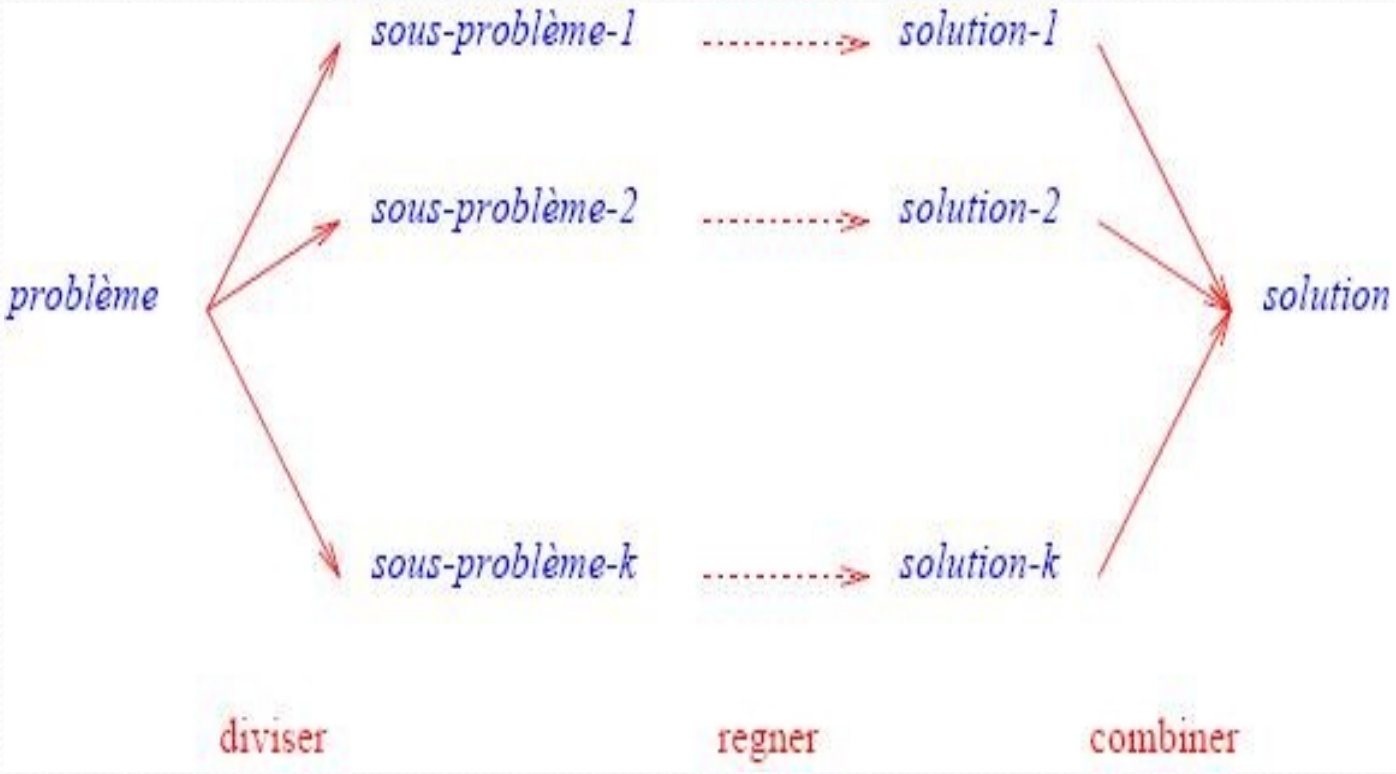
- Deux types de fonctions récursives
  - Récursivité directe
    - Fonction qui s'appelle elle-même : fonction A appelle A
  - Récursivité indirecte
    - Fonction A, appelle une autre fonction B, qui appelle C, …. qui appelle A..

Récursivité

- Diviser pour régner
  - Trois étapes à chaque niveau de récursivité :
    - Diviser le problème en un certain nombre de sous-problèmes.
    - Régner sur les sous-problèmes en les résolvant récursivement.
    - Si la taille d'un sous-problème est assez réduite, on peut le résoudre directement.
    - Combiner les solutions aux sous-problèmes en une solution complète pour le problème initial.

Récursivité

- Diviser pour régner



Récursivité

Exemple

- Exemple de récursivité : calcul de factorielle
  - Définition $n\,!=n\times(n-1)\times(n-2)\ldots\times 1$

Exemple : 4 !

$$4\times 3\,!$$
$$4\times(3\times 2\,!)$$
$$4\times(3\times(2\times 1\,!))$$
$$4\times(3\times(2\times(1\times 0\,!)))$$
$$4\times(3\times(2\times(1\times 1)))$$
$$4\times(3\times(2\times 1))$$
$$4\times(3\times 2)$$
$$4\times 6$$
$$24$$

$$n!=\begin{cases}1 \text{ si } n=0\\ n\times(n-1)! \quad \text{si } n>0\end{cases}$$

Récursivité

Exemple

- Fonction factorielle

```
Entier Fonction factorielle(Entier n)
Début
    Si (n==0) Renvoyer (1);
SiNon Renvoyer (n*factorielle(n-1));
FinSi
Fin
```

Toute fonction récursive est constituée de 2 parties :

→ Un cas particulier non récursif

→ Un cas général qui réduit le problème d'ordre n à un problème d'ordre n-1 jusqu'à arriver au cas particulier.

Récursivité

Conception

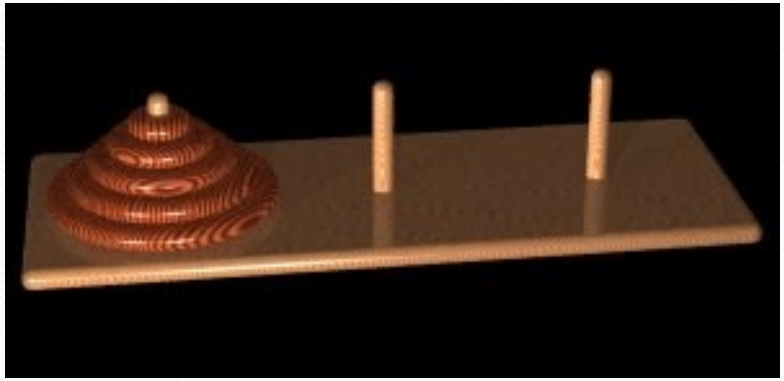
- Ce que ça donne sur le calcul de factorielle
  - La version itérative utilise moins d'espace mémoire

```
int factorielle(int n)
{
int som,produit=1;
    for (som=1;som<=n;som++)
    produit= produit * som;
return produit;
}
```

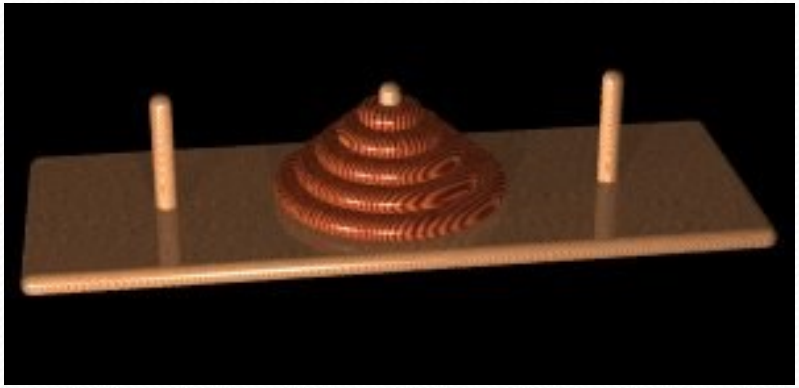
Récursivité

Exemple

- La tour de Hanoï : les règles
  - positions légales : tous les disques doivent être en place sur les axes ; aucun disque ne doit être posé sur un disque plus petit.
  - mouvements : un mouvement consiste à déplacer un disque à la fois, d'un axe à un autre (à condition, bien sûr, que la position obtenue soit légale).



Position initiale

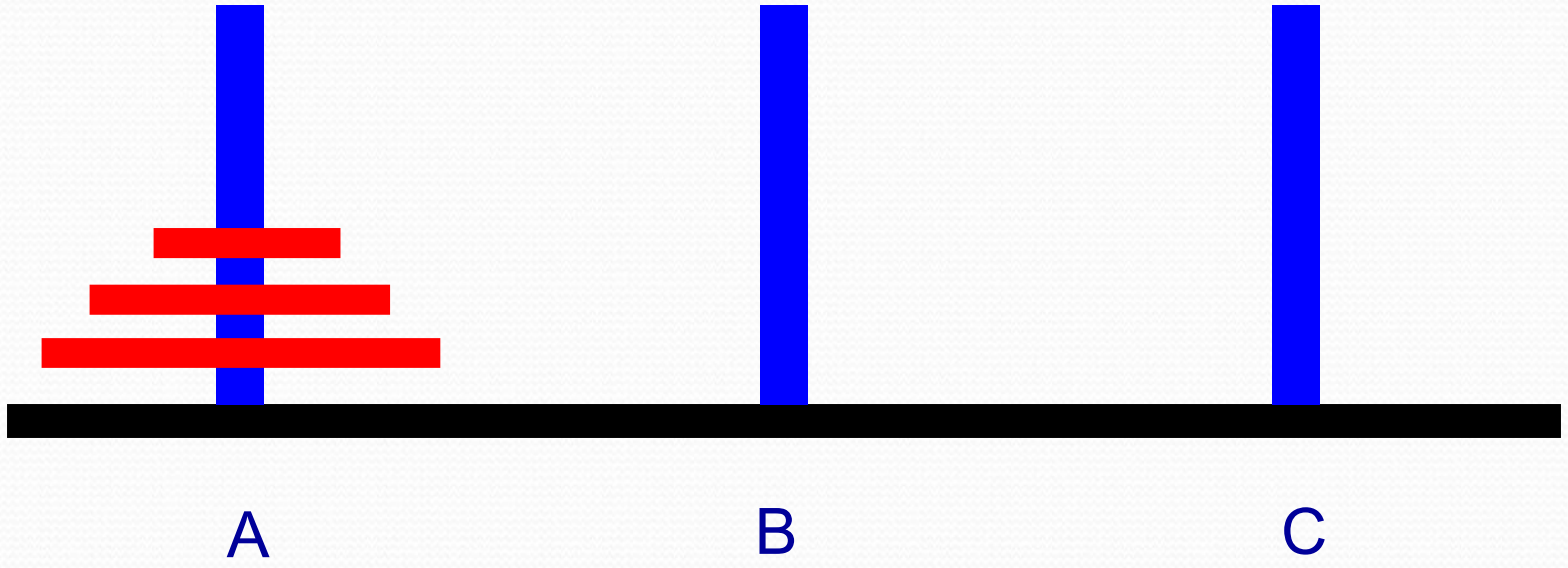


Position finale

Récursivité

Exemple (3 disques)

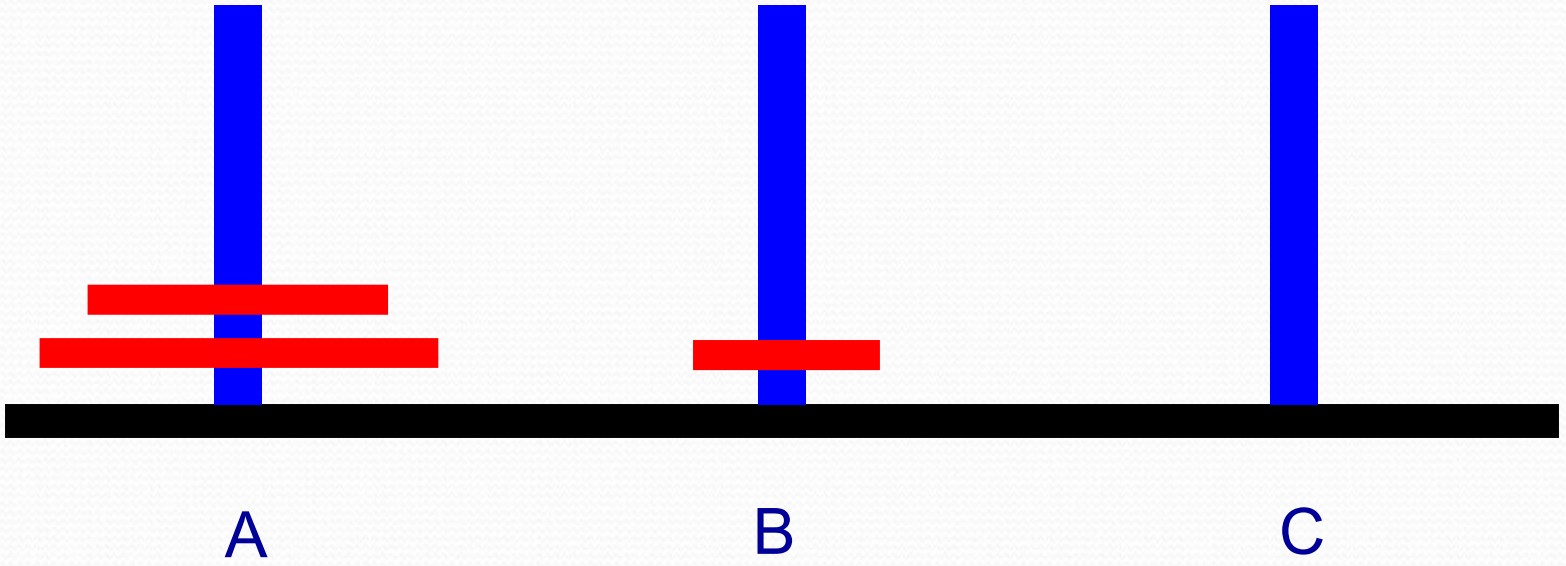
- La tour de Hanoï



Récursivité

Exemple(3 disques)

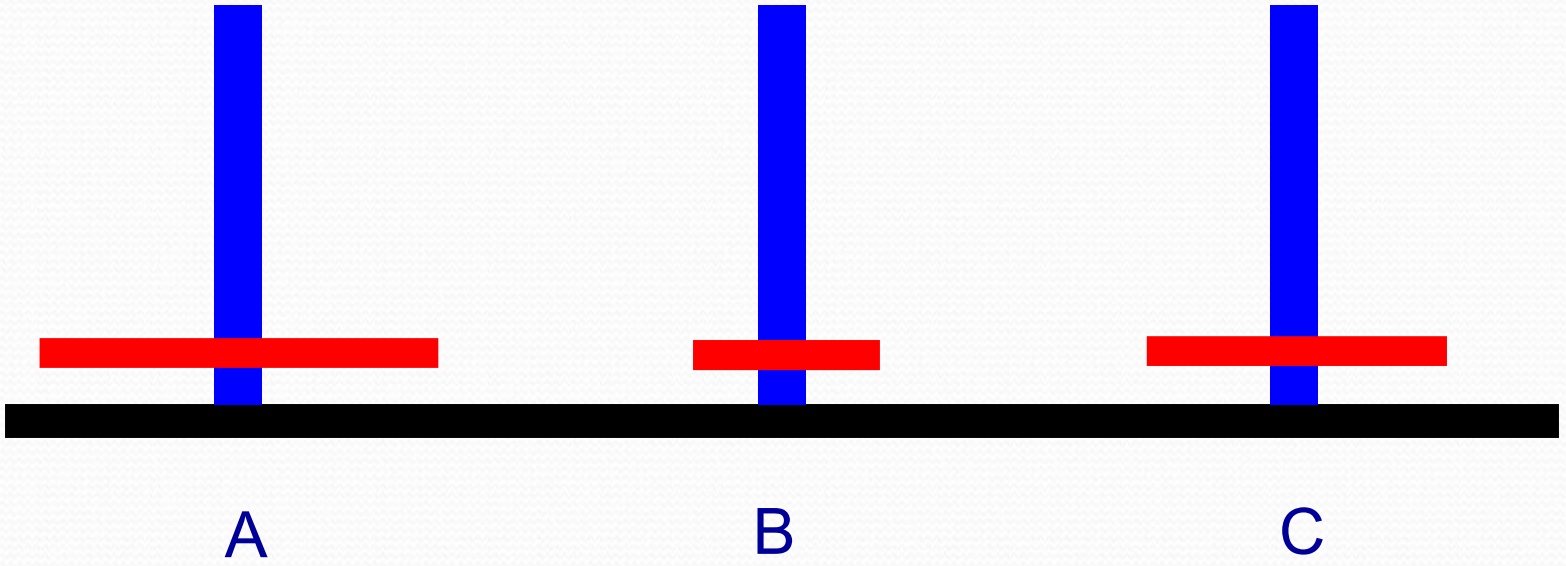
- La tour de Hanoï



Récursivité

Exemple (3 disques)

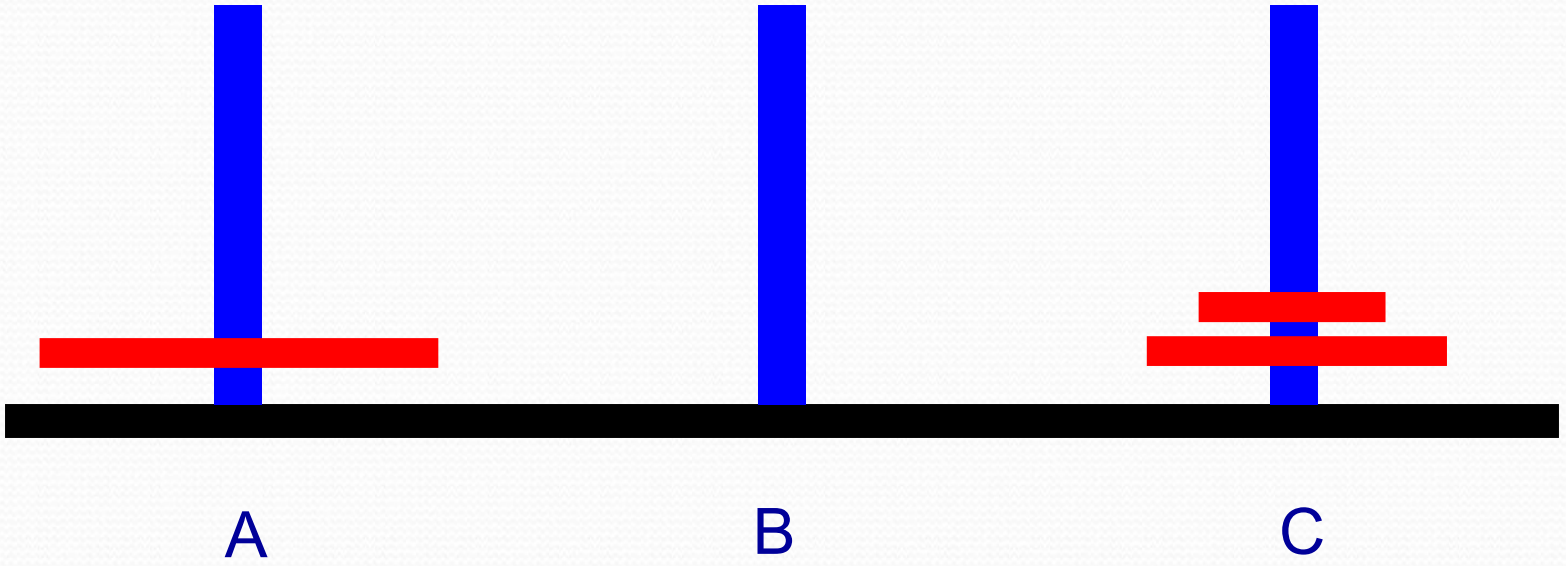
- La tour de Hanoï



Récursivité

Exemple (3 disques)

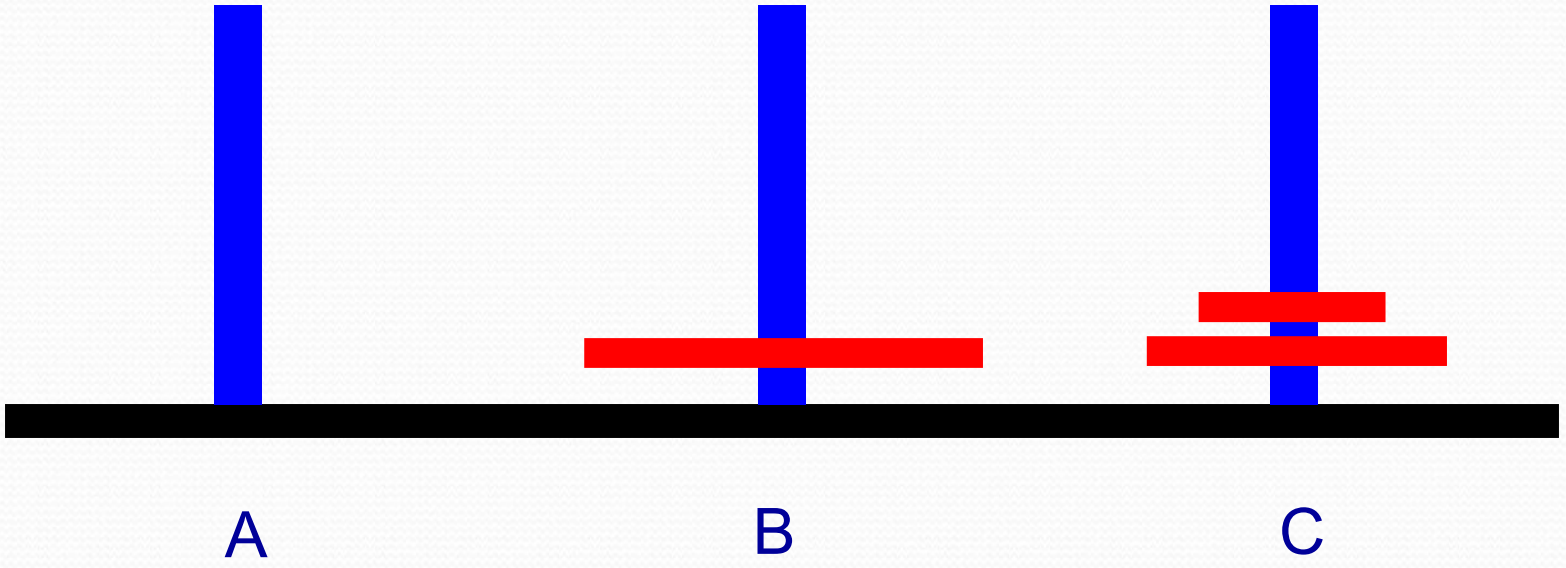
- La tour de Hanoï



Récursivité

Exemple (3 disques)

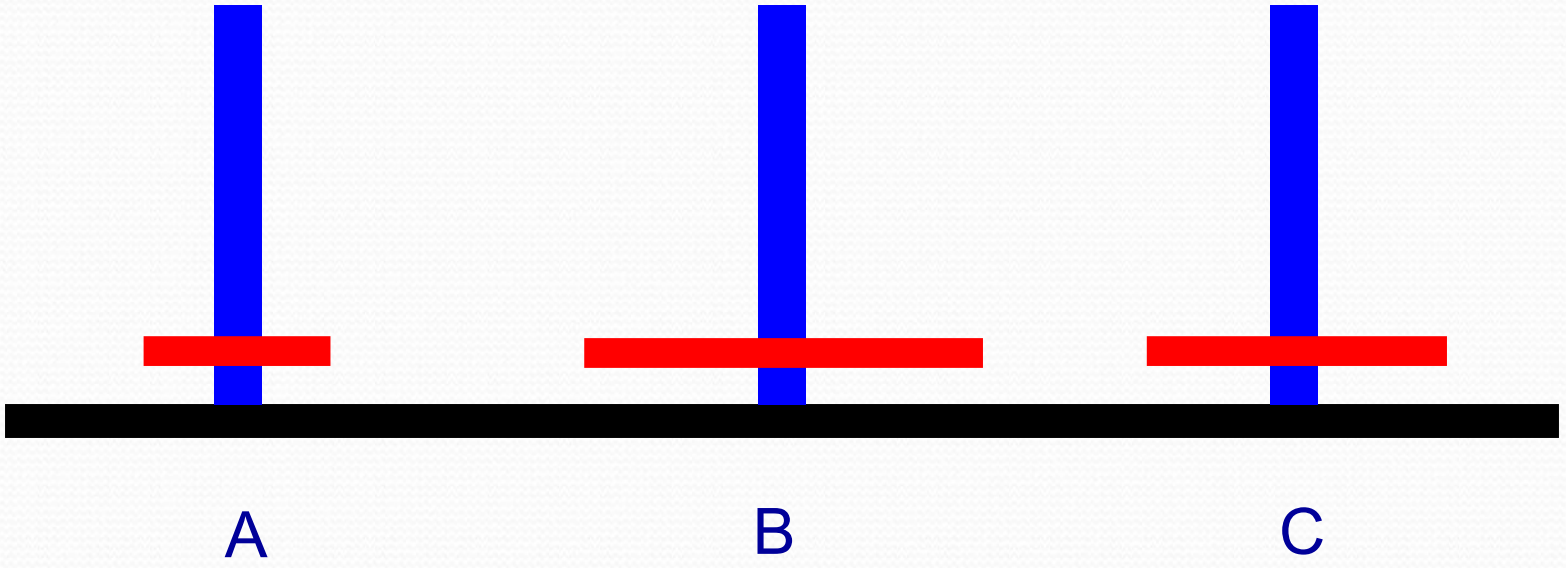
- La tour de Hanoï



Récursivité

Exemple (3 disques)

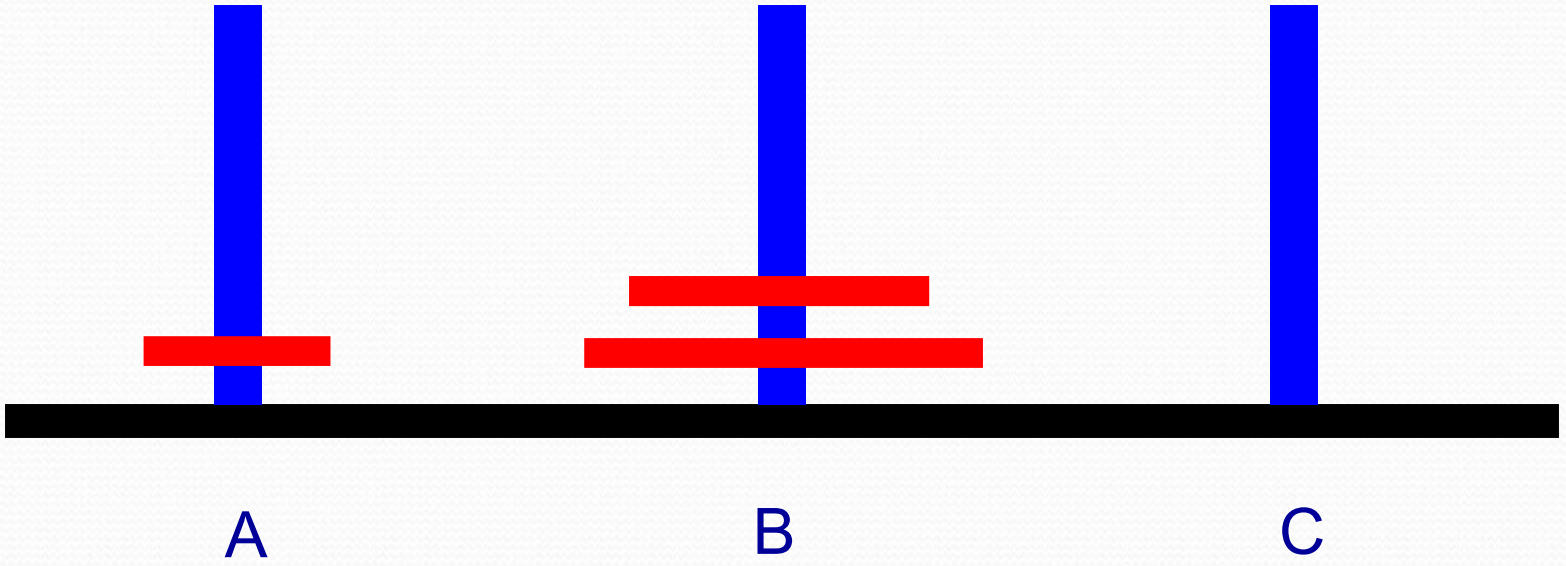
- La tour de Hanoï



Récursivité

Exemple (3 disques)

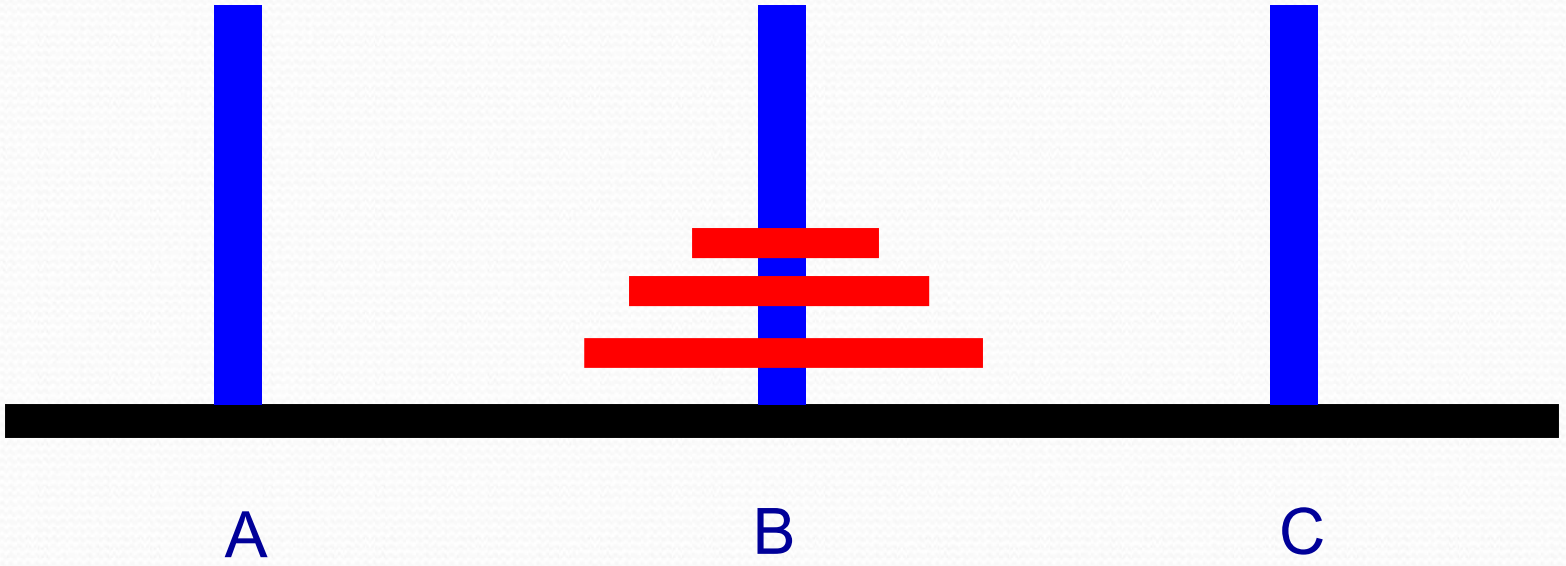
- La tour de Hanoï



Récursivité

Exemple (3 disques)

- La tour de Hanoï



Récursivité
Exemple

- Objectif
  - Ecrire la fonction :

```
Deplacer(n,A,B,C)
```

Déplacer n disques de la tour A à la tour B en utilisant la tour C comme tampon.

- Solution
  - Ne pas se focaliser sur la première étape mais sur la dernière : déplacer le dernier disque.
    - On suppose que les n-1 ont donc été placés en tour C pour pouvoir placer le disque n en tour B.

Récursivité

Exemple

- Principe
  - Pour déplacer n disques, il suffit de placer les n-1 plus petits sur une tour tampon, de déplacer le disque restant, puis de ramener les n-1 petits disques sur la tour de destination :

    `Deplacer(n-1,A,C,B) puis Deplacer(n-1,C,B,A)`

```
void Deplacer(int n, int deb, int fin, int temp) {
if (n > 0) {
    Deplacer(n-1,deb,temp, fin);
    Deplacer(n-1,temp,fin,deb);
}}
```

Récursivité

Conception

- Ce que ça donne sur la tour de Hanoi

```
void Deplacer(int n, int deb, int fin, int temp) {
int swap;
while (n > 0) {
    Deplacer(n-1,deb,temp, fin);
    n--;
    swap = deb;
    deb = temp;
    temp = swap;
}
}
```

Récursivité

Conception

- Ce que ça donne sur les suites de Fibonacci

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2} \quad \forall\ n \geq 2$$

- Version récursive

```
int fibonacci(int n)
{
if (n<=0) return 0;
else if (n==1) return 1;
else return fibonacci(n-1)+fibonacci(n-2);
}
```

Récursivité

Conception

- Suites de Fibonacci : version itérative

```
int fibonacci(int n){
int avder, der, courant;
if (n<=0) return 0;
else if (n==1) return 1;
else {
    avder = 0; der = 1;
    for (int i=2;i<=n;i++) {
        courant = avder + der;
        avder = der; der = courant;
    }
    return courant;
}
}
```

Récursivité

Conclusion

- Inconvénients de la récursivité
  - La pile d'exécution n'est pas bornée, elle croît et décroît au cours de l'exécution.
  - Les programmes consomment de l'espace mémoire.
  - La gestion de la pile prend du temps

Pour plus d'efficacité, utiliser la version itérative.

Pour plus de lisibilité et de facilité d'écriture, utiliser la version récursive.

Exercices

Exercice 1:

Donnez la version récursive de la procédure suivante :

```
Procedure compter(Entier n)
Variables
    Entier i ;
Début
Pour i de 1 A n
    Ecrire(i);
    i suivant;
Fin Pour
Fin
Fin Procedure
```

Exercices

Exercice 1:

Donnez la version récursive de la procédure suivante :

Procedure compter(Entier n)

```
Début
    Si n <= 1
      Ecrire(n) ;
    Sinon
      compter(n-1)
      Ecrire(n) ;
    FinSi
Fin Procedure
```