



# Système d'Exploitation 2

## Chapitre 1 : Les interblocages (Deadlock)

(partie 1)

1 ère Année Second Cycle

Dr. M. Baba Ahmed

# Plan (partie 1)

- Introduction
- Définition de l'interblocage
- Conditions d'interblocage
- Graphes d'allocation de ressources
- Approches de gestion de l'interblocage

# Introduction (rappel)

- ❑ L'exécution d'un processus nécessite **un ensemble de ressources** qui peuvent être physique (mémoire principale, disques, périphériques...etc) ou logique (variable, logiciel, fichier ..etc) qui lui sont attribué par le **système d'exploitation SE**
- ❑ lorsqu'un processus demande un accès a une ressource déjà allouée a un autre processus, le **SE** décide de le mettre en attente jusqu'à la libération de la ressource

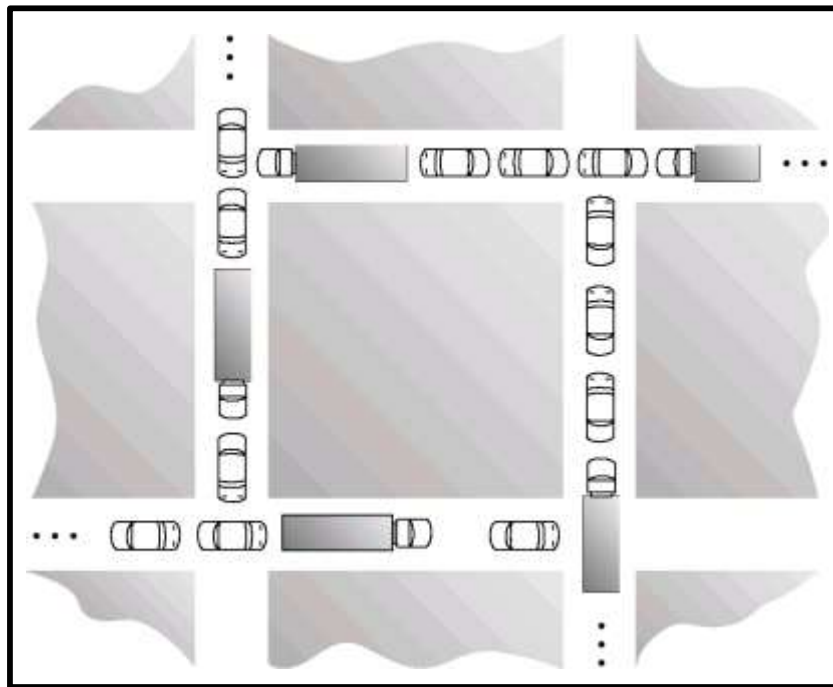
# Introduction

- ❑ Dans un système multiprogrammé plusieurs processus partagent un nombre fini de ressources.
  - Un processus demande des ressources, si ces ressources ne sont pas disponibles, ce processus doit se mettre en attente.

Problème de compétition entre les processus pour l'utilisation des ressources

- ❑ Il peut arriver qu'un processus reste indéfiniment en attente
  - Les ressources demandées par ce processus sont allouées à des processus qui sont eux-mêmes en attente d'autres ressources.
- ❑ Cette situation est appelée **interblocage**, **étrainte fatale** ou "**deadlock**".

# Exemple d'Interblocage (circulation routiere)



- ❑ Dans ce cas aucun ne pourra s'engager

# Exemple d'Interblocage (Accès a une base de données)

- ❑ Supposons deux processus A et B qui demandent des accès aux enregistrements (R1,R2) d'une base de données.
- ❑ On arrive à une situation d'interblocage si :
  - ▶ Le processus A a verrouillé l'enregistrement R1 et demande l'accès à l'enregistrement R2.
  - ▶ Le processus B a verrouillé l'enregistrement R2 et demande l'accès à l'enregistrement R1.

interblocage! aucun processus ne peut compléter à moins qu'un des processus ne puisse être suspendu ou puisse retourner en arrière

# Exemple (sémaphore)

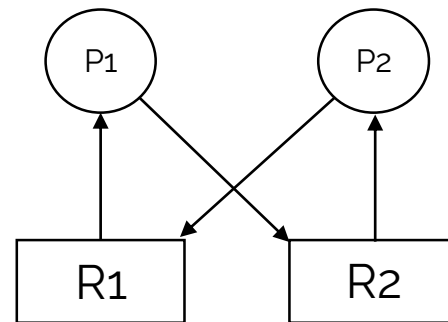
<pre>ressource_1; ressource_2;</pre>	
Processus (P1)	Processus (P2)
<pre>void process_P1 {     P(ressource_1);     P(ressource_2);     V(ressource_1);     V(ressource_2); }</pre>	<pre>void process_P2 {     P(ressource_2);     P(ressource_1);     V(ressource_2);     V(ressource_1); }</pre>

- ❑ Le processus P1 prend la ressource 1 et le processus P2 prend la ressource 2, P1 demande la ressource 2 et P2 demande la ressource 1

Interblocage des deux processus P1 et P2, aucun des processus ne peut finir l'exécution

# Définition de l'Interblocage

❑ Le processus **P1** détient la ressource **R1** et attend une Autre ressource **R2** qui est utilisée par le processus **P1**, tandis que le processus **P2** détient la Ressource **R2** et attend une autre ressource **R1** qui est utilisée par le processus **P1**.



❑ Un ensemble de processus est dans une situation d'**interblocage** si chaque processus attend un **événement** que seul un autre processus de l'ensemble peut provoquer.

- **Événement**: libération d'une ressource



# Modèle du système

- ▶ Un système comporte un nombre fini de ressource devant être distribué parmi

Un certain nombre de processus concurrents.

- Les ressources sont groupées en classes (types : Imprimantes, mémoires, lecteurs CD, registres, fichiers,...).
- Chaque classe de ressources peut comporter un nombre fini d'exemplaires (copies identiques).

# Protocole d'accès aux ressources

- ❑ Pour acquérir une ressource chaque processus suit le protocole suivant :

Demander (Ri)  
<Utilisation>  
Liberer (Ri)

Demander() et Libérer() sont généralement des appels systèmes (Ex: Open File et Close File)

- ❑ **Demande** : Si la demande n'est pas satisfaite immédiatement, le processus doit se mettre en attente, il ne pourra passer à l'étape suivante (utilisation) que si la ressource lui a été allouée.
- ❑ **Libération** : Toute ressource doit être libérée(restituée), au bout d'un temps fini, après son utilisation par un processus.

# Conditions d'Interblocage

□ L'interblocage demande la présence simultanée de 4 conditions

1. **Exclusion mutuelle** : les ressources ne sont pas partageables, un seul processus à la fois peut utiliser la ressource.
2. **Possession et attente** : un processus qui détient des ressources peut demander de nouvelles ressources (sans relâcher celles qu'il détient)
3. **Pas de réquisition** : Les ressources déjà détenues ne peuvent pas être retiré de force a un processus elle doivent être libérées par le processus qui les détient.
4. **Attente circulaire** : Il existe un ensemble de  $k$  processus  $P_1, P_2, \dots, P_k$  tels que :  
 $P_1$  attend une ressource détenue par  $P_2$ ,  $P_2$  attend une ressource détenue par  $P_3$  .....  $P_k$  attend une ressource détenue par  $P_1$

# Conditions d'Interblocage

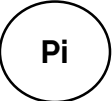

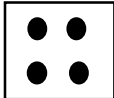
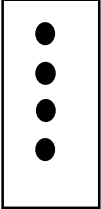
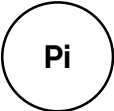
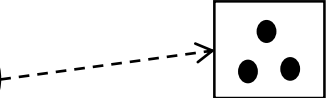
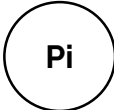
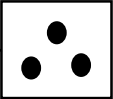
Remarque :

- ❑ Une attente circulaire est un interblocage
- ❑ Les 3 premières conditions (exclusion mutuelle, possession et attente, et pas de réquisition) n'impliquent pas nécessairement l'interblocage, car l'attente circulaire pourrait ne pas être vérifiée

# Graphe d'allocation de ressources

- ❑ Un ensemble de sommets  $V$  et d'arêtes  $E$
- ❑  $V$  est partitionné en 2 ensembles:
  - $P = \{P_1, P_2, \dots, P_n\}$ , l'ensemble qui consiste de tous les processus dans le système
  - $R = \{R_1, R_2, \dots, R_m\}$ , l'ensemble qui consiste de tous les types de ressources dans le système
- ❑ Arêtes **requête** : arêtes dirigées  $P_i \rightarrow R_j$
- ❑ Arêtes **affectation** : arêtes dirigées  $R_k \rightarrow P_l$

# Graphe d'allocation de ressources (notation)

- ❑ Processus : 
- ❑ Ressource dont il y a un seul exemplaire : 
- ❑ Ressource dont il y a **N** exemplaires (instances) : (ex : 4)  **R<sub>j</sub>** 
- ❑ P<sub>i</sub> demande un exemplaire de R<sub>j</sub>, dont il y en a 3:   **R<sub>j</sub>**
- ❑ P<sub>j</sub> détient (et utilise) un exemplaire de R<sub>j</sub> :   **R<sub>j</sub>**

# Graphe allocation de ressources avec interblocage

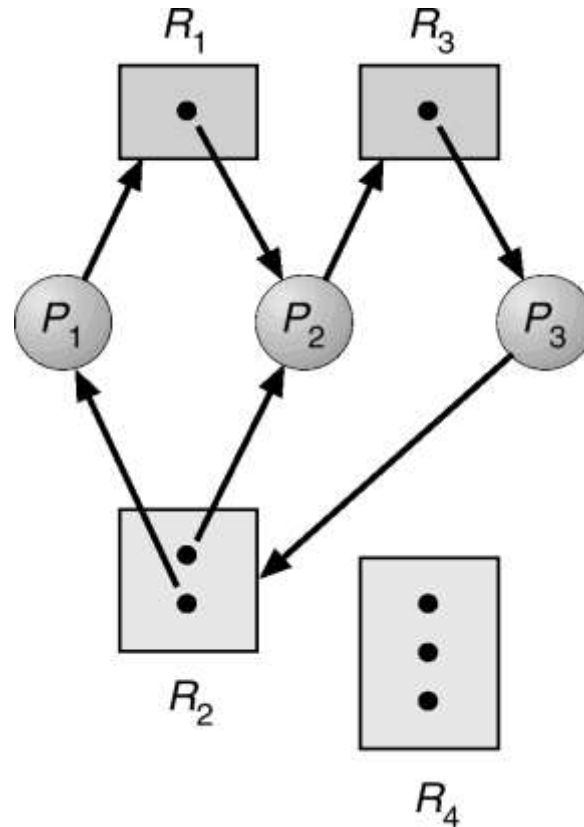
Nous avons deux cycles :

$P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$

$P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$

Aucun processus ne peut terminer

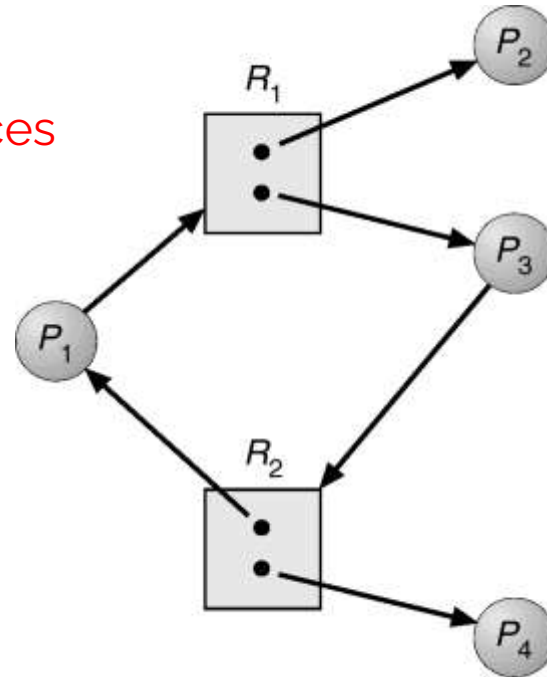
Aucune possibilité de sortir



# Graphe allocation de ressources avec cycle, mais pas d'interblocage (pourquoi?)

Pas d'attente circulaire les ressources

Peuvent devenir disponible





# Terminaison de processus et libération de ressources

❑ **Hypothèse:** Un processus qui a saisi toutes les ressources dont il a besoin peut terminer

- Cette terminaison pourrait conduire à la libération de ressources
- Qui pourraient être saisies par d'autre processus qui les attendent
- Ce qui pourrait conduire à la terminaison d'autres processus

❑ Il n'y a pas d'interblocage si tous les processus peuvent terminer de cette manière

# Constatations

- ❑ Les cycles dans le graphe d'allocation de ressources ne signalent pas nécessairement une attente circulaire
  - S'il n'y a pas de cycles dans le graphe, aucun interblocage
  - S'il y a un cycle :
    - Si seulement une ressource par type → **interblocage**
    - Si plusieurs ressources par type → **possibilité d'interblocage**
- ❑ Il faut se poser la question:
  - Y-a-t-il au moins un processus qui peut terminer et si oui, quels autres processus peuvent terminer en conséquence?

# Approches de gestion de l'interblocage

1. Ignorer le problème (Politique de l'autruche)
2. Détection et guérison
  - Réagir en cas d'interblocage
3. Évitement : en allouant les ressources avec précaution
  - Les interblocages sont possibles, mais sont évités (avoidance)
4. Prévention : en empêchant l'apparition d'une condition nécessaire (Exclusion mutuelle, ...)
  - Concevoir le système de façon qu'un interblocage soit impossible

# La politique de l'Autruche

- ❑ « Plonger la tête dans le sable en prétendant qu'il n'y a aucun problème ! »
  - Ignorer le problème, qui donc doit être résolu par l'utilisateur.
- ❑ Malheureusement, c'est l'approche la plus utilisée!
  - Windows et UNIX adoptent cette stratégie
- ❑ Arguments :
  - Les interblocages surviennent rarement
  - Le coût de la prévention ou de la détection est élevé

# Détection

- ❑ **Cas 1:** Une seule instance par ressource
  - Interblocage s'il existe un **cycle** dans le graphe
- ❑ **Cas 2:** Plusieurs instances par ressource
  - L'existence d'un cycle n'implique pas forcément un interblocage
  - La non-existence de cycle implique qu'il n'y a pas interblocage

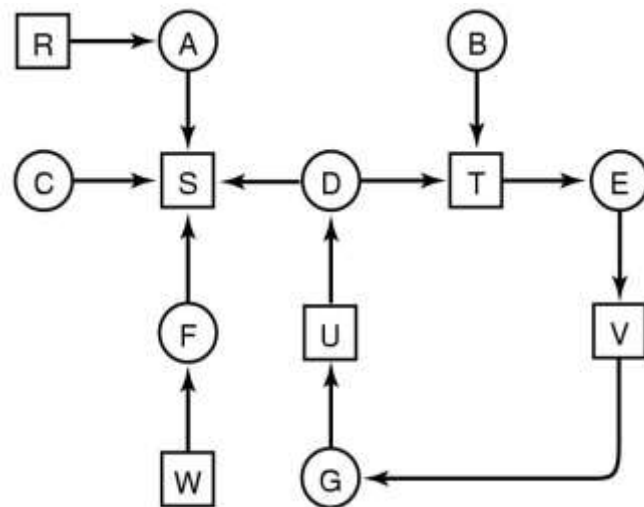
# Cas 1 : une seule instance

- ❑ Représenter les demandes et les allocations de ressources par un graphe
  - Le graphe représente l'état du système à un instant donné.
- ❑ Utiliser un algorithme de recherche d'un cycle dans un graphe
  - **Algorithme coûteux** : En général, nombreux processus et ressources

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

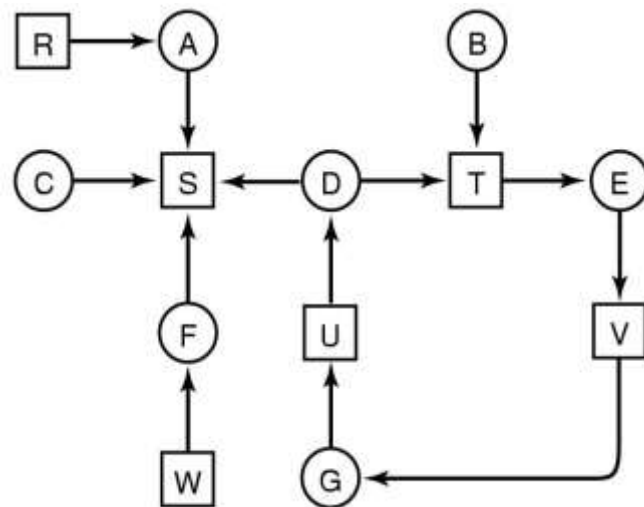


--	--	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)



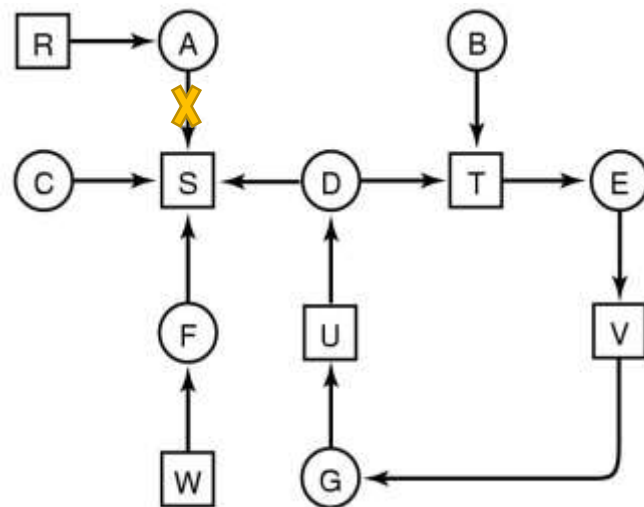
A							
---	--	--	--	--	--	--	--



# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

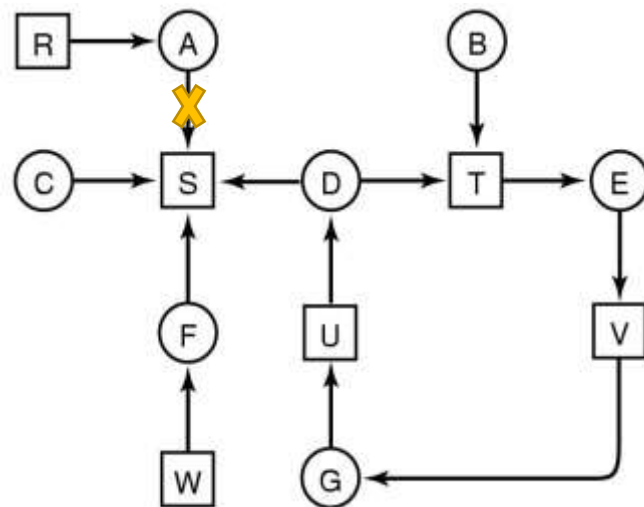


A							
---	--	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

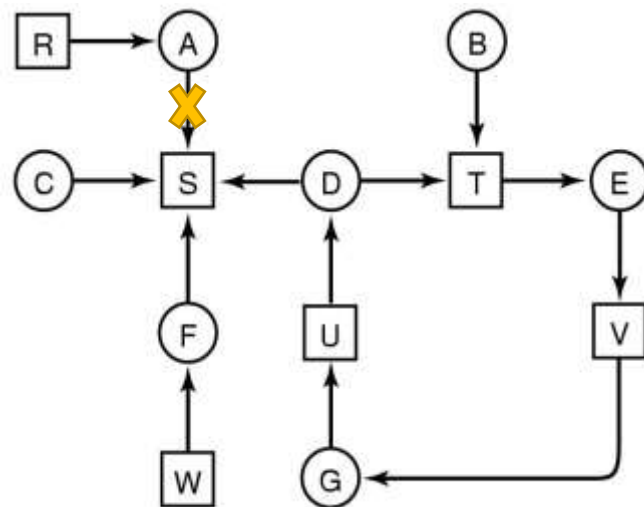


A	S						
---	---	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

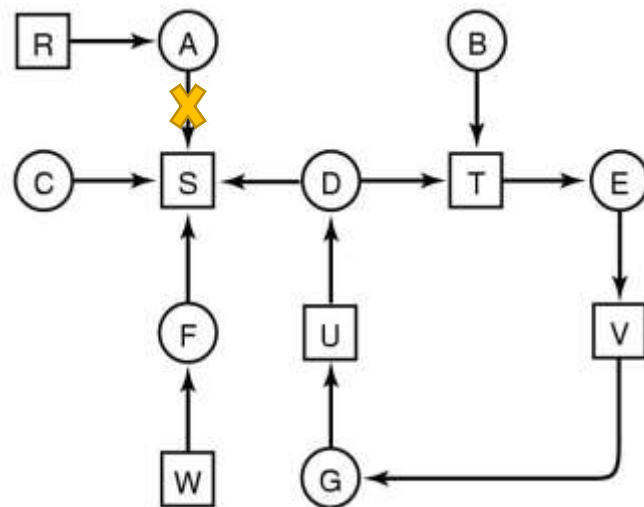


A							
---	--	--	--	--	--	--	--

## Example :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  Cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

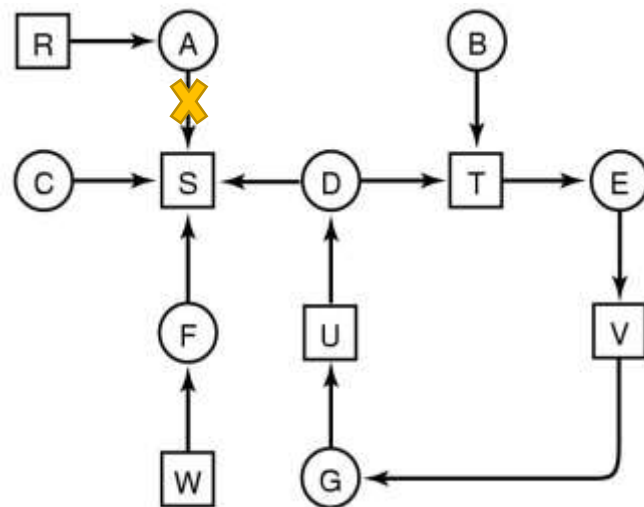


--	--	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

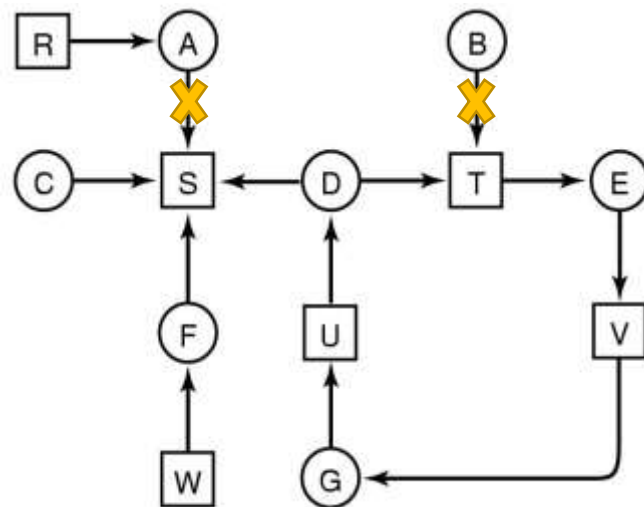


B							
---	--	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

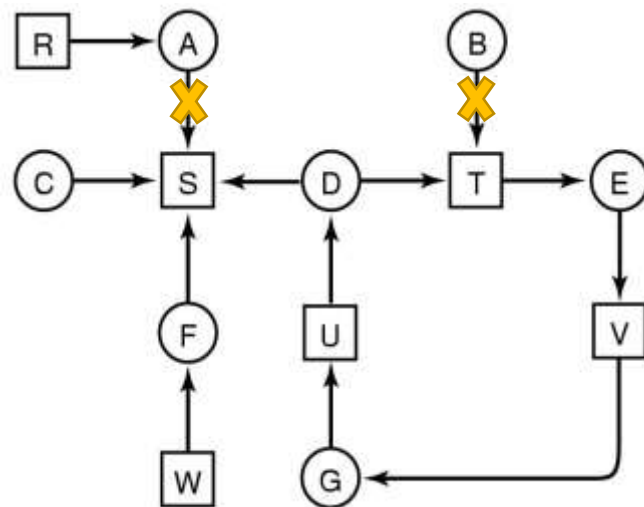


B							
---	--	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

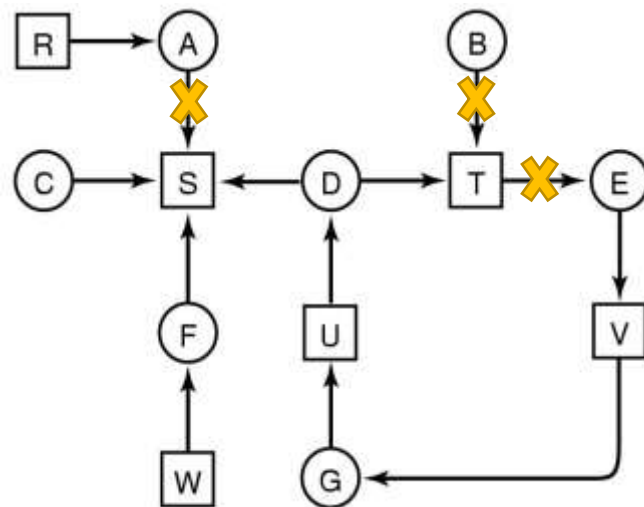


B	T						
---	---	--	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)



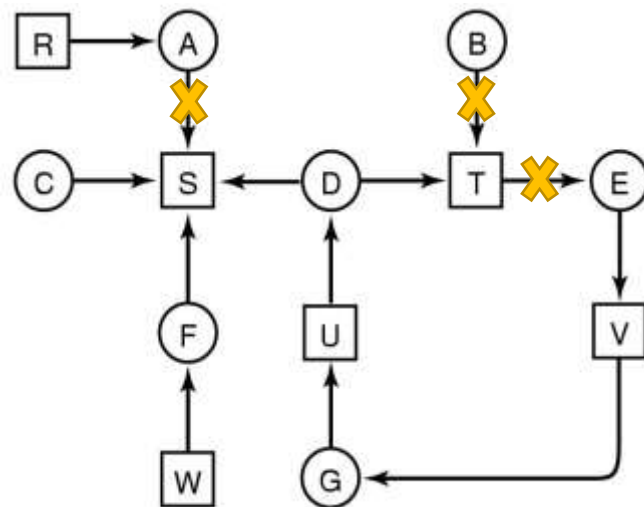
B	T						
---	---	--	--	--	--	--	--



# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$   $\exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

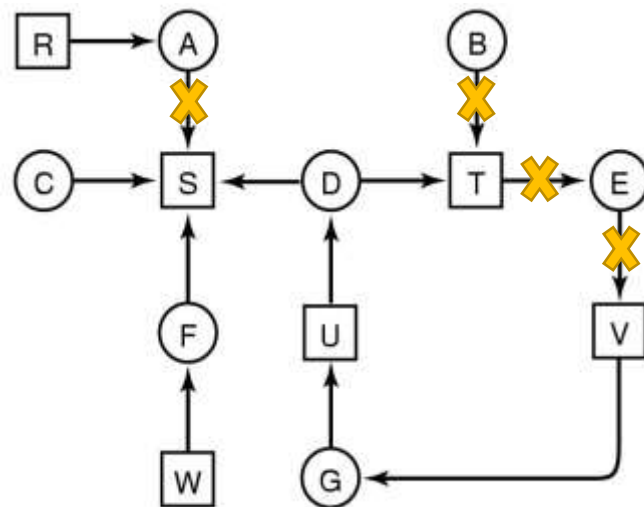


B	T	E					
---	---	---	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

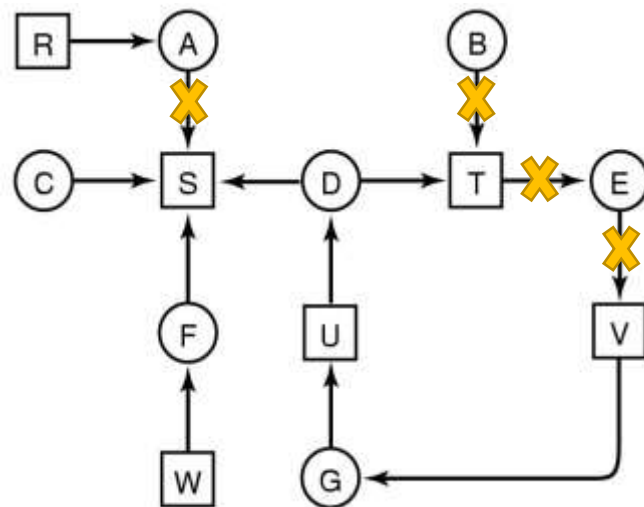


B	T	E					
---	---	---	--	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

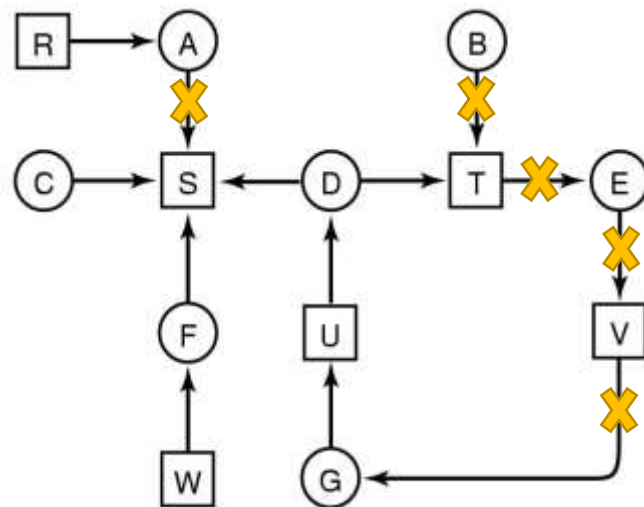


B	T	E	V				
---	---	---	---	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

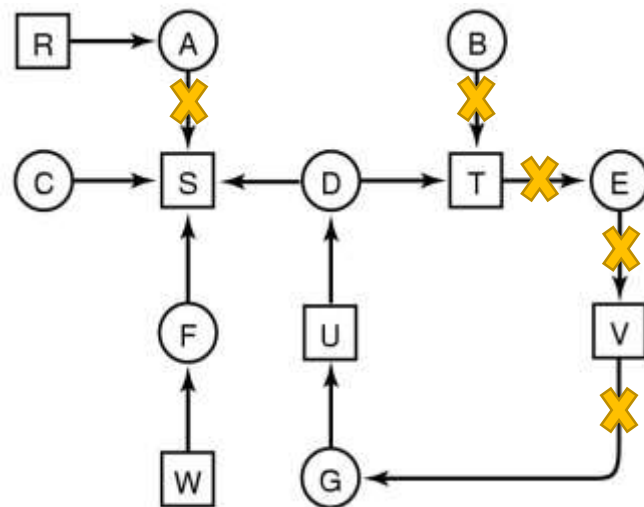


B	T	E	V				
---	---	---	---	--	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

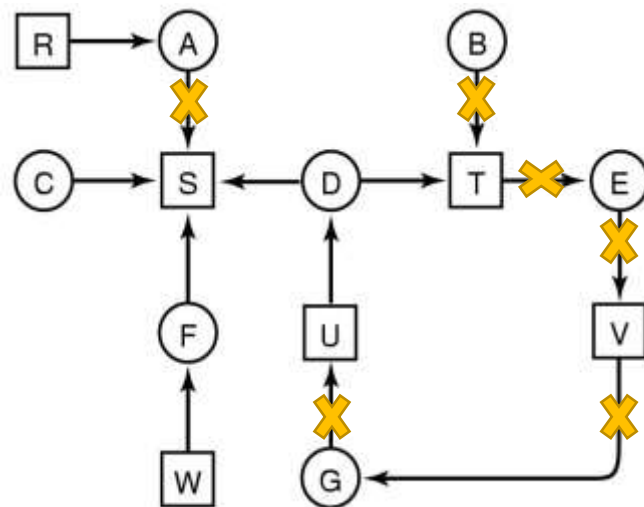


B	T	E	V	G			
---	---	---	---	---	--	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

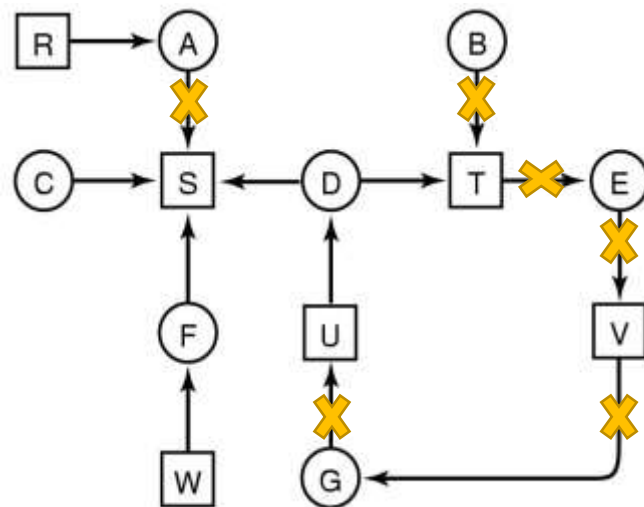


B	T	E	V	G			
---	---	---	---	---	--	--	--

## Example :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow \exists$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

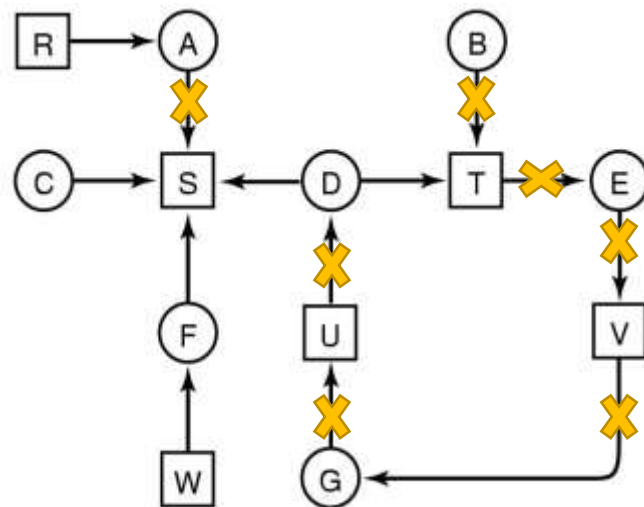


B	T	E	V	G	U		
---	---	---	---	---	---	--	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)



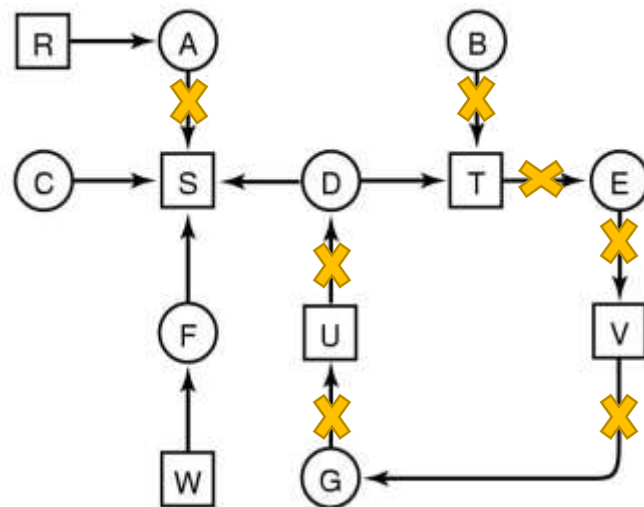
B	T	E	V	G	U		
---	---	---	---	---	---	--	--



# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

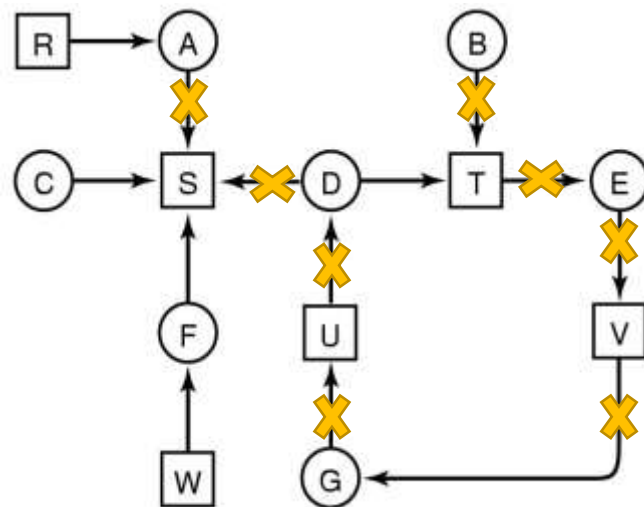


B	T	E	V	G	U	D	
---	---	---	---	---	---	---	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

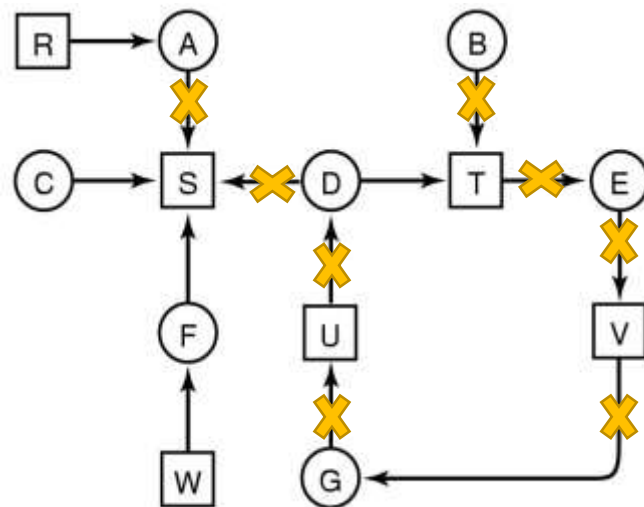


B	T	E	V	G	U	D	
---	---	---	---	---	---	---	--

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

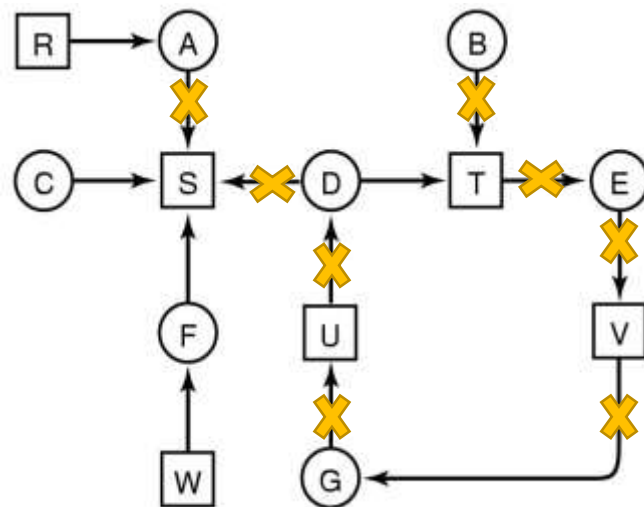


B	T	E	V	G	U	D	S
---	---	---	---	---	---	---	---

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)

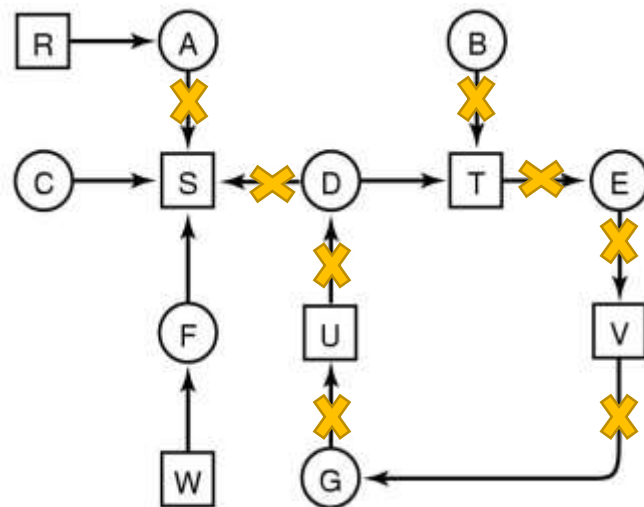


B	T	E	V	G	U	D	S
---	---	---	---	---	---	---	---

# Exemple :

1. Initialiser  $L$  une pile vide et désignez tous les arcs comme non marqué
2. Pour chaque nœud  $n$ :
3. Si le nœud  $n$  n'est pas dans la pile  $L$ , l'ajouter à  $L$   
Sinon  $\rightarrow$  cycle et l'algorithme prend fin.
4. Si il n'existe pas un arc non marqué sortant de  $n$ , dépiler  $n$   
Si la pile est vide, revenez à l'étape 2  
Sinon le nœud au sommet de la pile devient  $n$  et revenez à l'étape 4  
Sinon choisissez au hasard un arc sortant non marqué et marquez-le, pointer sur le nœud au bout de l'arc choisi qui devient  $n$  et revenez à l'étape 3

Si l'algorithme ne se termine pas à l'étape 3, il n'y a pas de cycle (d'interblocage)



B	T	E	V	G	U	D	
---	---	---	---	---	---	---	--



## Cas 2 : plusieurs instances

- ❑ Modélisation de l'allocation de ressources à travers des matrices en considérons que:
  - Le système est composé de  $n$  processus  $P_1, P_2, \dots, P_n$
  - Et de  $m$  classes (types) de ressources  $R_1, R_2, \dots, R_m$
- ❑ Représenter les demandes et les allocations de ressources par des matrices :
  - Cette représentation fait intervenir les structures de données suivantes :

# Matrices de modélisation d'allocations de ressources

1. Vecteur de ressources existantes (Existing resources):
  - **E**: Tableau[1..M] d'entiers
  - **E[i]** = le nombre maximum d'exemplaires de la ressource  $R_i$  (disponible au départ).  
Ex:  $E=(4,2,3,1)$
2. Vecteur des ressources disponibles (Available):
  - **Available**: Tableau[1..M] d'entiers / ou **A[j]**
  - **Available[j]**=nombre d'exemplaires libres de la ressource  $R_j$ .  
Ex:  $Available=(2,1,0,0)$



## Cas 2 : plusieurs instances

3. Matrice d'allocation (Allocation) :

- **Allocation:** Tableau[1..N, 1..M] d'entiers;
- **Allocation[i,j]**=nombre d'exemplaires de la ressource  $R_j$  alloués au processus  $P_i$ .

		<b>R1</b>	<b>R2</b>	<b>R3</b>	<b>R4</b>
Ex: Allocation	<b>P1</b>	0	0	1	0
	<b>P2</b>	2	0	0	1
	<b>P3</b>	0	1	2	0

# Matrices de modélisation d'allocations de ressources

4. Matrice des demandes en attente(Requests) :

- **Request:** Tableau[1..N, 1..M] De Entier ;
- **Request[i,j]** = nombre d'exemplaires de la ressource Rj demandés et non obtenus par le processus Pi.

		R1	R2	R3	R4
Ex: Request	P1	2	0	0	1
	P2	1	0	1	1
	P3	2	1	0	0

# Algorithme de détection d'Interblocage

$E = \text{Existing}[i]$

$A = \text{Available}[j]$

$C[i,j] = \text{Allocation}[i,j]$

$R[i,j] = \text{Request}[i,j]$

1. Rechercher un processus  $P[i]$  non marqué dont la rangée  $R[i]$  est inférieure à  $A$
2. Si ce processus existe, ajouter la rangée  $C[i]$  à  $A$ , marquer le processus et revenir à l'étape 1 ( $A=A+C[i,j](P_i)$  ou  $A=A+\text{Allocation}(P_i)$ )
3. Si ce processus n'existe pas, les processus non marqués sont en interblocage. L'algorithme se termine.

**Remarque :** étape 2 permet de garantir que l'allocation des ressources actuelle ne conduira pas à un état d'interblocage

# Algorithme de détection d'Interblocage

Exemple :

$$E = ( 4 , 2 , 3 , 1 )$$

$$A = ( \quad , \quad , \quad , \quad )$$

C =  
Allocation

	R1	R2	R3	R4
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R =  
Request

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

$$\text{Available} = E - \sum C[i, j]$$

# Algorithme de détection d'Interblocage

Exemple :

$$E = ( 4 , 2 , 3 , 1 )$$

$$A = ( 2 , 1 , 0 , 0 )$$

C =  
Allocation

	R1	R2	R3	R4
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R =  
Request

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

$$\text{Available} = E - \sum C[i, j] \quad E = \text{Available} + \sum C[i, j]$$

# Algorithme de détection d'Interblocage

Exemple :

$$E = ( 4 , 2 , 3 , 1 )$$

C =  
Allocation

	R1	R2	R3	R4
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

$$A = ( 2 , 2 , 2 , 0 )$$

R =  
Request

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0



$$\text{Available} = E - \sum C[i, j]$$

P3

# Algorithme de détection d'Interblocage

Exemple :

$$E = ( 4 , 2 , 3 , 1 )$$


C =  
Allocation

	R1	R2	R3	R4
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

$$A = ( 4 , 2 , 2 , 1 )$$

R =  
Request

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0



$$\text{Available} = E - \sum C[i, j]$$

$$P_3 > P_2$$

# Algorithme de détection d'Interblocage

Exemple :

$$E = ( 4 , 2 , 3 , 1 )$$




$$A = ( 4 , 2 , 3 , 1 )$$

C =  
Allocation

	R1	R2	R3	R4
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R =  
Request

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0



$$\text{Available} = E - \sum C[i, j]$$

Pas d'interblocage  
Tout les processus sont  
marqués

$$P_3 > P_2 > P_1$$