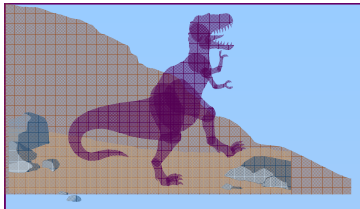
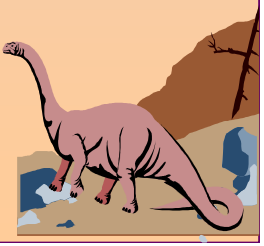
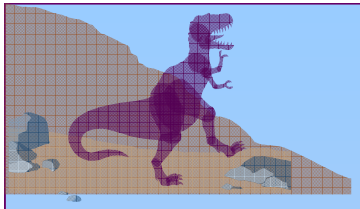


Capitolo 8: Stallo dei processi

- Modello del sistema
- Caratterizzazione delle situazioni di stallo
- Metodi per la gestione delle situazioni di stallo
- Prevenire le situazioni di stallo
- Evitare le situazioni di stallo
- Rilevamento delle situazioni di stallo
- Ripristino da situazioni di stallo
- Approccio combinato per la gestione dello stallo

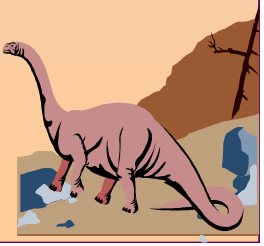


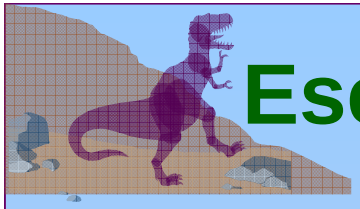


Il problema dello stallo

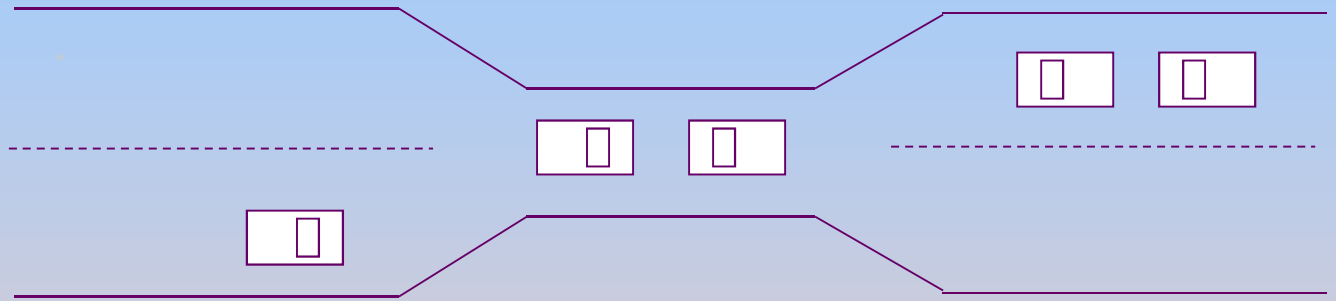
- Un insieme di processi bloccati, in cui ciascun processo detiene una risorsa e attende di accedere a una risorsa in possesso di un altro processo dell'insieme.
- Esempio
  - Il sistema dispone di 2 unità a nastri.
  - P_1 e P_2 possiedono ciascuno una di queste unità e ciascuno richiede un'altra unità a nastri-
- Esempio
  - semafori *A* e *B*, inizializzati a 1

P_0	P_1
wait (A);	*wait(B)*
wait (B);	*wait(A)*

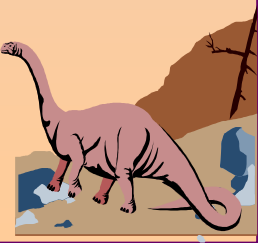


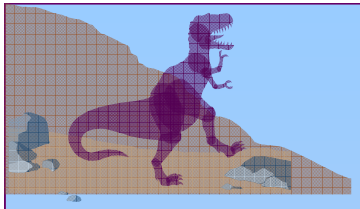


Esempio dell’attraversamento del ponte



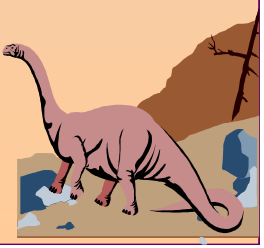
- Traffico in una sola direzione.
- Ciascuna sezione del ponte può essere considerata una risorsa.
- Se si verifica uno stallo, può essere risolto se una macchina fa retromarcia: prelazione di risorse e ristabilimento di uno stato sicuro (*rollback*).
- Più macchine potrebbero dover fare retromarcia, in caso di stallo.
- È possibile si verifichi un’attesa indefinita (*starvation*).

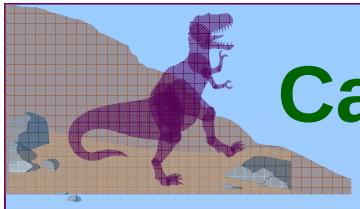




Modello del sistema

- Tipi di risorse $R_1, R_2, \ldots, R_m$
  cicli di CPU, spazio di memoria, dispositivi di I/O
- Ciascun tipo di risorsa R_i ha W_i istanze.
- Ciascun processo utilizza una risorsa nella seguente sequenza:
  - richiesta
  - uso
  - rilascio

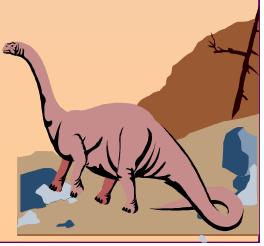


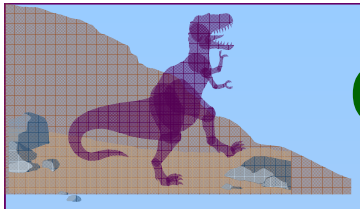


Caratterizzazione delle situazioni di stallo

Si può avere una situazione di stallo solo se si verificano contemporaneamente le seguenti quattro condizioni.

- **Mutua esclusione:** solo un processo alla volta può utilizzare una risorsa.
- **Possesso e attesa:** un processo in possesso di almeno una risorsa attende di acquisire risorse già in possesso di altri processi.
- **Impossibilità di prelazione:** una risorsa può essere rilasciata dal processo che la possiede solo volontariamente, dopo aver terminato il proprio compito.
- **Attesa circolare:** deve esistere un insieme $\{P_0, P_1, \ldots, P_n\}$ di processi tale che P_0 attende una risorsa posseduta da P_1, P_1 attende una risorsa posseduta da P_2, …, P_{n-1} attende una risorsa posseduta da P_n, e P_0 attende una risorsa posseduta da P_0.

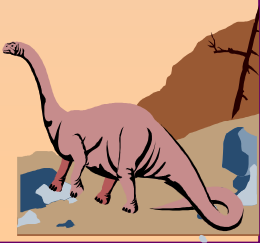


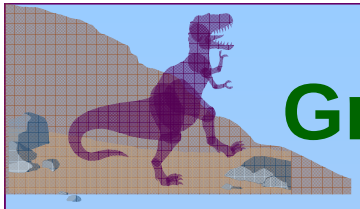


Grafo di assegnazione delle risorse

Un insieme di vertici V e di archi E.

- V è composto da due sottoinsiemi:
  - $P = \{P_1, P_2, \ldots, P_n\}$, che rappresenta tutti i processi del sistema.
  - $R = \{R_1, R_2, \ldots, R_m\}$, che rappresenta tutti i tipi di risorsa del sistema.
- Arco di richiesta: un arco orientato $P_1 \rightarrow R_j$
- Arco di assegnazione: un arco orientato $R_j \rightarrow P_i$



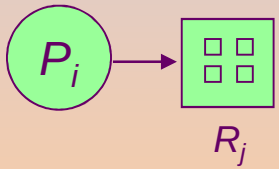


Grafo di assegnazione delle risorse *(Cont.)*

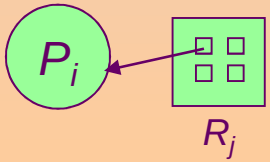
- Processo

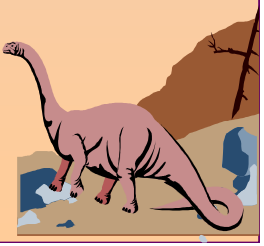
- Tipo di risorsa con 4 istanze

- P_i richiede un'istanza del tipo di risorsa R_j

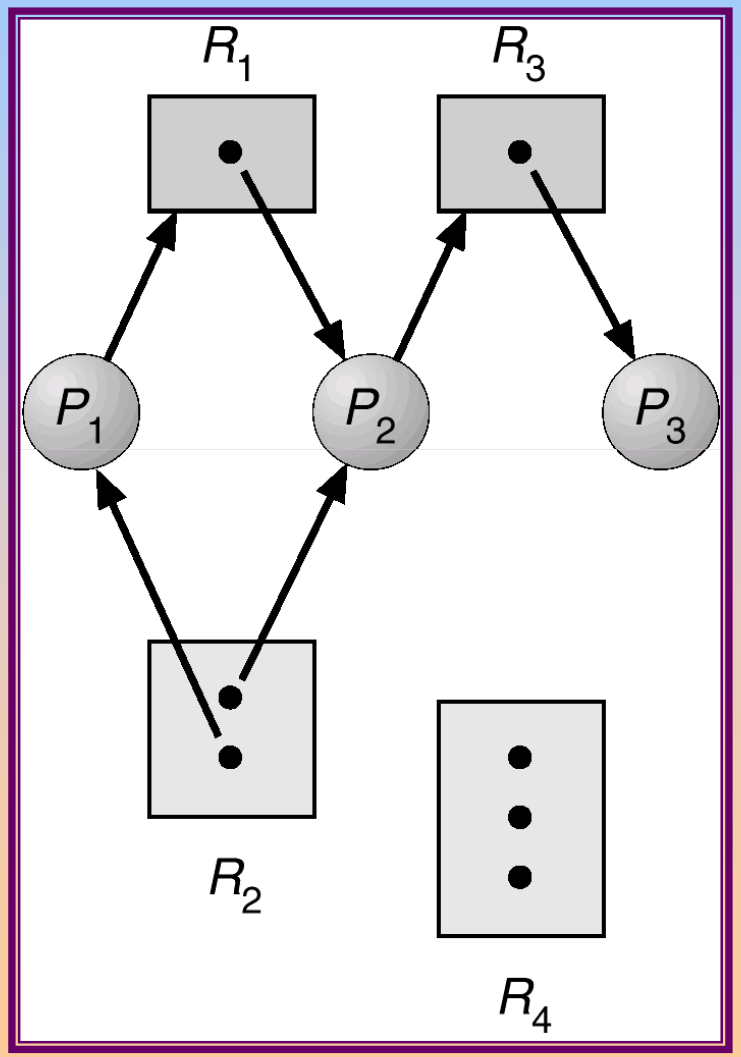


- P_i possiede un'istanza del tipo di risorsa R_j

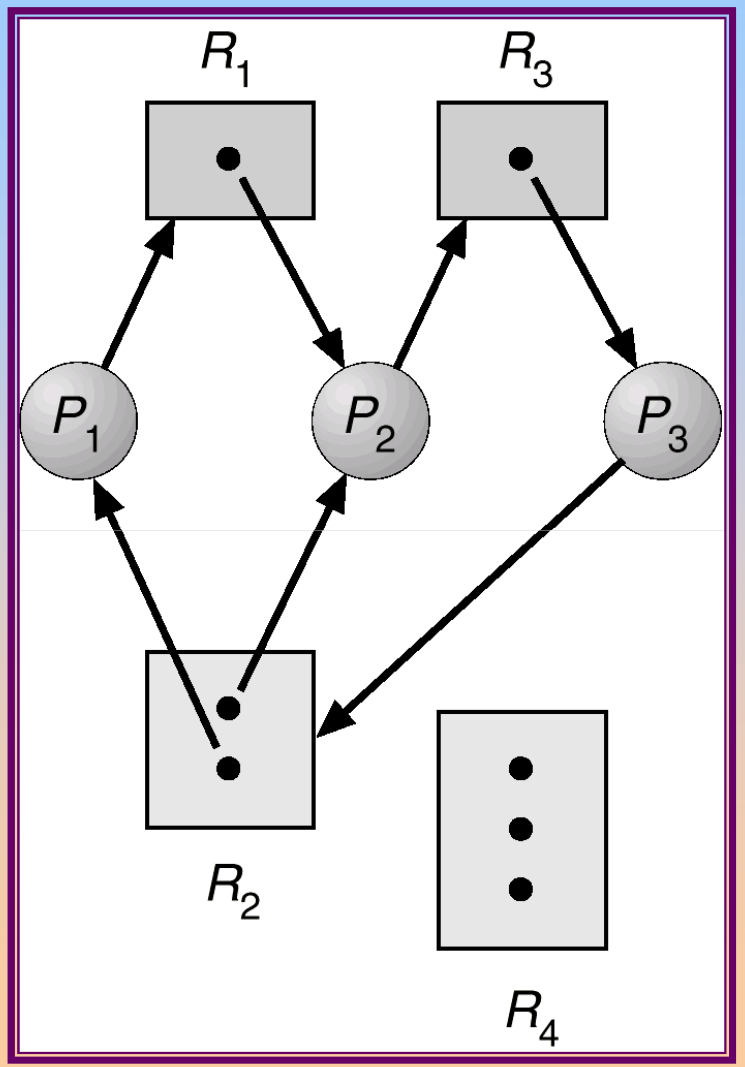




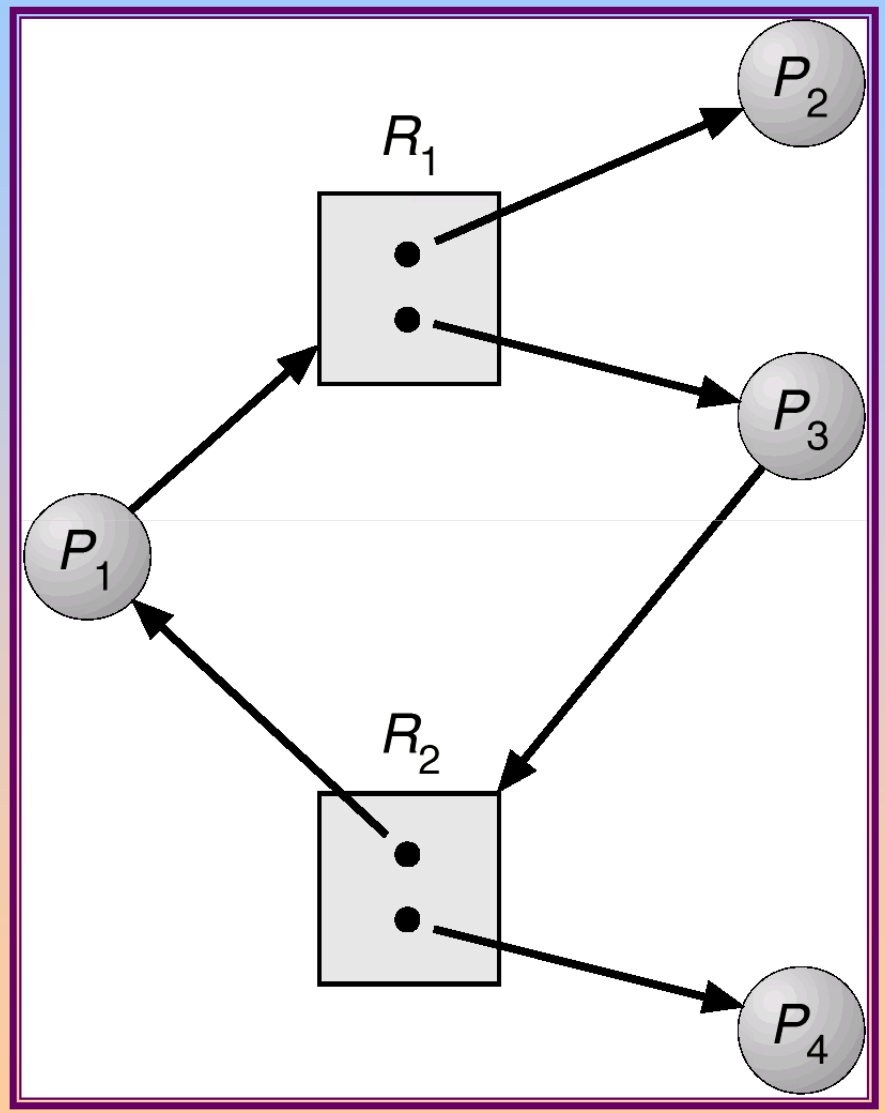
Grafo di assegnazione delle risorse

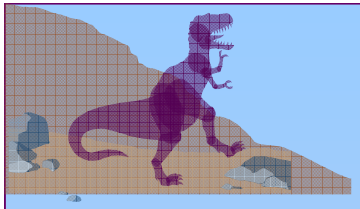


Grafo di assegnazione delle risorse con stallo



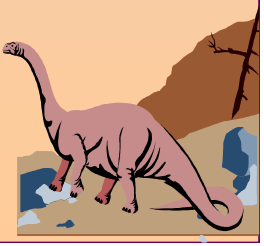
Grafo di assegnazione delle risorse con un ciclo ma senza stallo

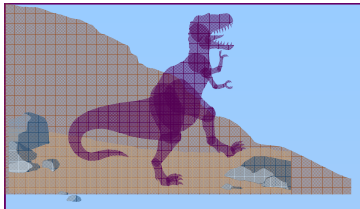




Osservazioni

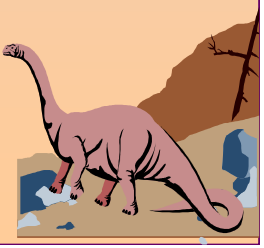
- Se il grafo non contiene cicli $\Rightarrow$ non si verificano situazioni di stallo.
- Se il grafo contiene un ciclo $\Rightarrow$
  - se c'è solo un'istanza per tipo di risorsa, allora si verifica una situazione di stallo.
  - se vi sono più istanze per tipo di risorsa, allora c'è la possibilità che si verifichi una situazione di stallo.

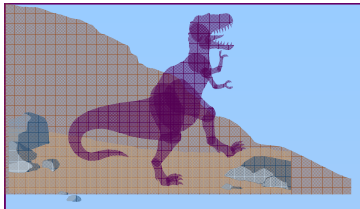




Metodi per la gestione delle situazioni di stallo

- Assicurare che il sistema non entri *mai* in stallo
- Consentire al sistema di entrare in stallo, individuarlo e quindi eseguire il ripristino.
- Ignorare del tutto il problema “fingendo” che le situazioni di stallo non possano mai verificarsi nel sistema; è la soluzione usata nella maggior parte dei sistemi operativi, compreso UNIX.

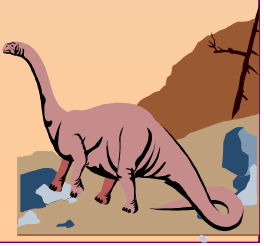


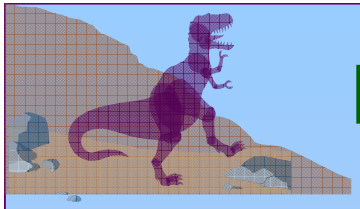


Prevenire le situazioni di stallo

Si può prevenire il verificarsi di uno stallo assicurando che almeno una delle quattro condizioni necessarie non si verifichi.

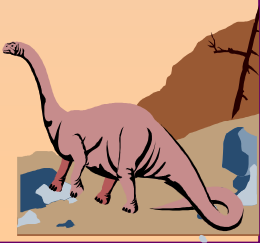
- **Mutua esclusione**: non richiesta per le risorse condivisibili; deve invece valere per le risorse non condivisibili.
- **Possesso e attesa**: occorre garantire che ogni volta che un processo richiede una risorsa, non ne possegga altre.
  - Ogni processo, prima di iniziare la propria esecuzione, deve richiedere tutte le risorse che gli servono, e queste gli devono essere assegnate; oppure occorre permettere a un processo di richiedere risorse solo se non ne possiede.
  - Basso utilizzo delle risorse; possibile attesa indefinita.

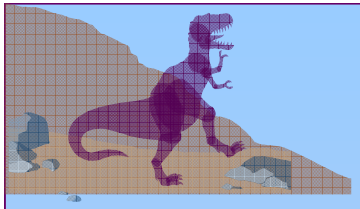




Prevenire le situazioni di stallo *(Cont.)*

- **Impossibilità di prelazione**
  - Se un processo che possiede una o più risorse ne richiede un'altra che non gli si può assegnare immediatamente, allora si esercita la prelazione su tutte le risorse attualmente in suo possesso.
  - Le risorse si aggiungono alla lista delle risorse che il processo sta attendendo.
  - Il processo viene nuovamente avviato solo quando può ottenere sia le vecchie risorse sia quelle che sta richiedendo.
- **Attesa circolare**: impone un ordinamento totale all'insieme di tutti i tipi di risorse e un ordine crescente di numerazione per le risorse richieste da ciascun processo.

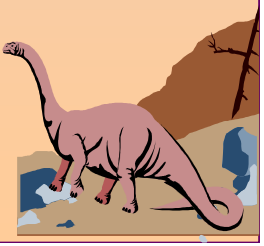


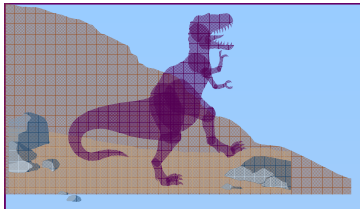


Evitare le situazioni di stallo

Un metodo alternativo per evitare le situazioni di stallo consiste nel richiedere ulteriori informazioni sui modi di richiesta delle risorse.

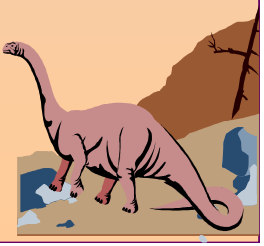
- Il modello più utile e più semplice richiede che ciascun processo dichiari il *numero massimo* delle risorse di ciascun tipo di cui necessita.
- L'algoritmo per evitare lo stallo esamina dinamicamente lo stato di assegnazione delle risorse per garantire che non possa esistere una condizione di attesa circolare.
- Lo *stato* di assegnazione dele risorse è definito dal numero dirisorse disponibili e assegnate, e dalle richieste massime dei processi.

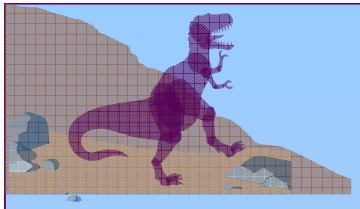




Stato sicuro

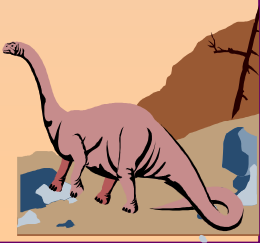
- Uno stato si dice sicuro se il sistema è in grado di assegnare risorse a ciascun processo (fino al suo massimo) in un certo ordine e impedire il verificarsi di uno stallo.
- Un sistema si trova in uno stato sicuro solo se esiste una **sequenza sicura**.
- La sequenza <P_1, P_2, …, P_n> è sicura se per ogni P_i, le richieste che P_i può ancora fare si possono soddisfare impiegando le risorse attualmente disponibili + le risorse possedute da tutti i P_j, con $j < i$.
  - Se le risorse necessarie al processo P_i non sono immediatamente disponibili, allora P_i può attendere che tutti i P_j abbiano finito.
  - A quel punto, P_i può ottenere tutte le risorse di cui ha bisogno, completare il compito assegnato, restituire le risorse assegnate, e terminare.
  - Quando P_i termina, P_{i+1} può ottenere le risorse richieste, e così via.

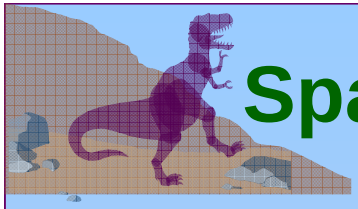




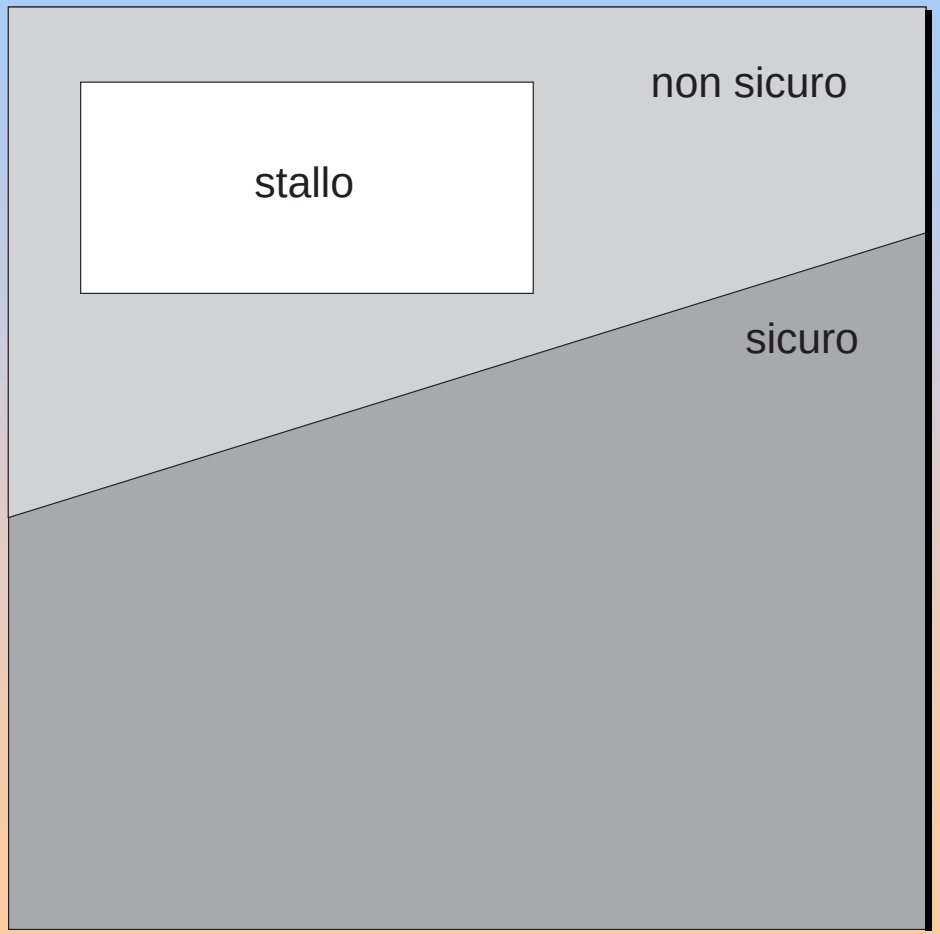
Osservazioni

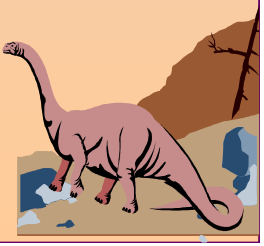
- Se un sistema è in uno stato sicuro ⇒ non si verificano situazioni di stallo.
- Se un sistema è in uno stato non sicuro ⇒ possibilità di stallo.
- Evitare lo stallo ⇒ assicurare che il sistema non si trovi mai in uno stato non sicuro.

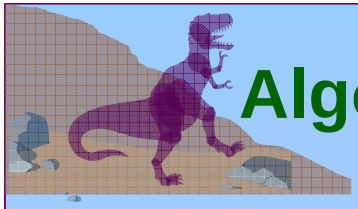




Spazi degli stati sicuri, non sicuri e di stallo

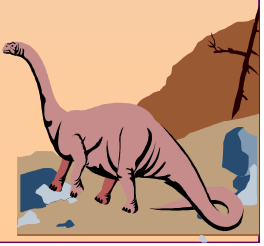




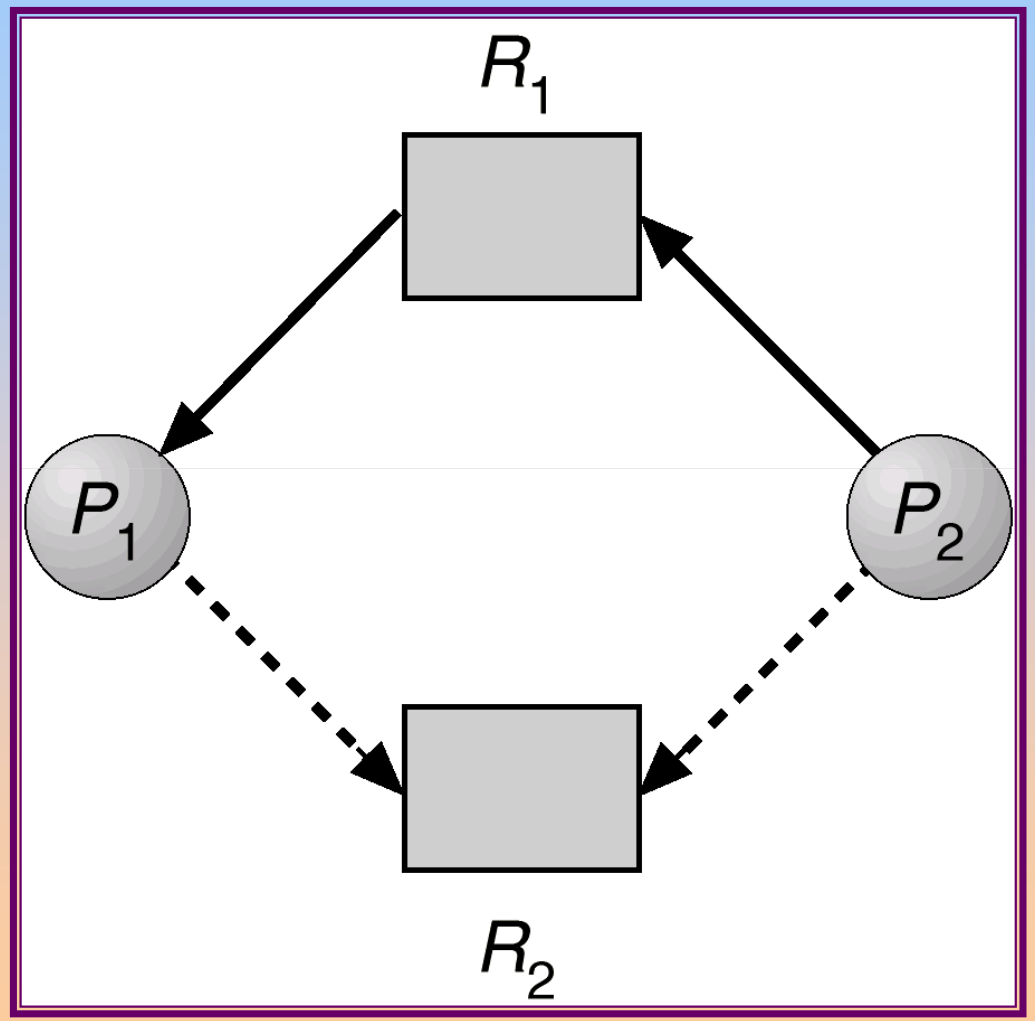


Algoritmo con grafo di assegnazione delle risorse

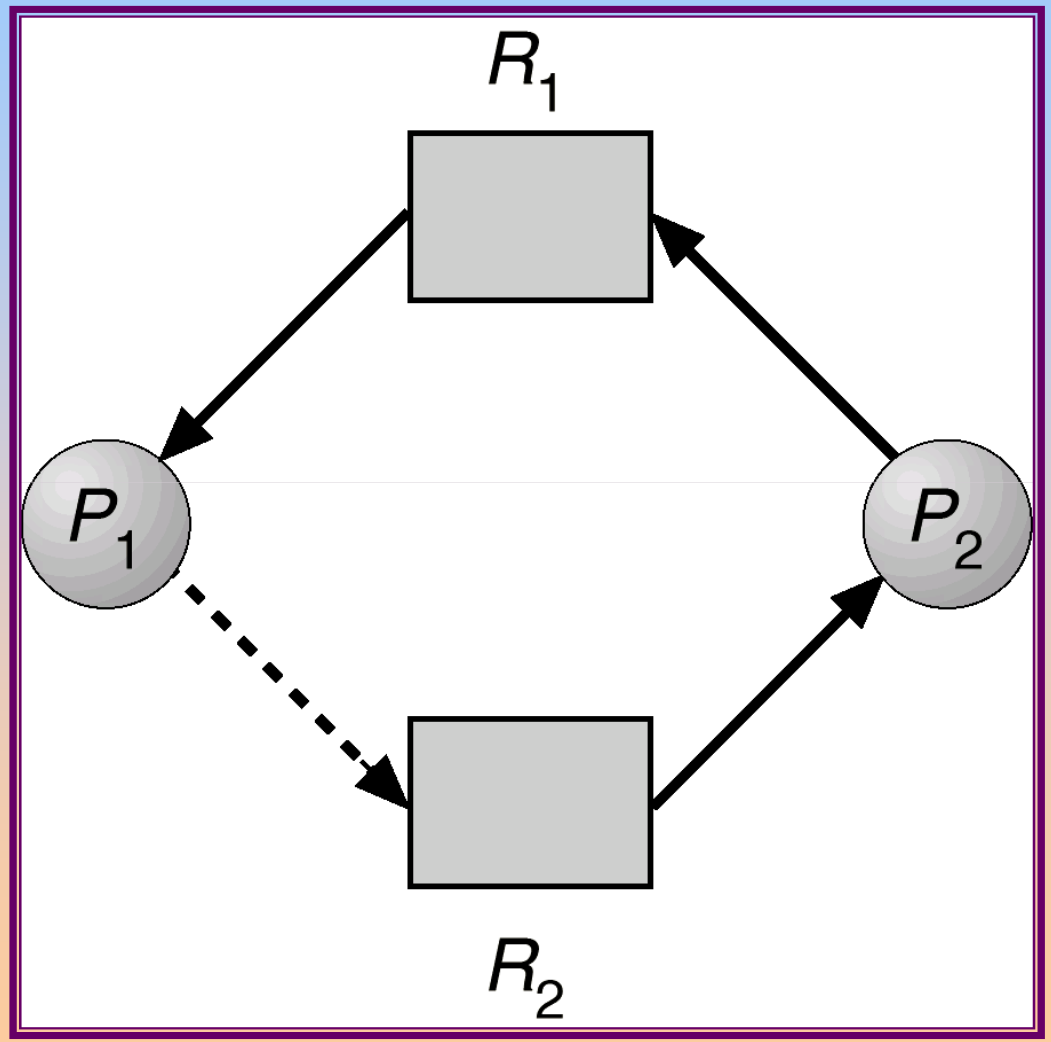
- **Arco di reclamo** $P_i \rightarrow R_j$ indica che il processo P_i può richiedere la risorsa R_j; rappresentato con una linea tratteggiata.
- Quando un processo richiede una risorsa, l'arco di reclamo diventa un **arco di richiesta**.
- Quando un processo rilascia una risorsa, l'arco di assegnazione ridiventa un arco di reclamo.
- Le risorse devono essere richiamate a priori nel sistema.

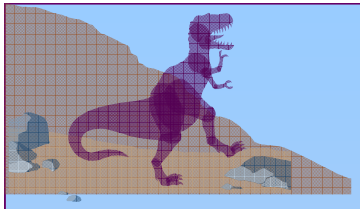


Grafo di assegnazione delle risorse per evitare le situazioni di stallo



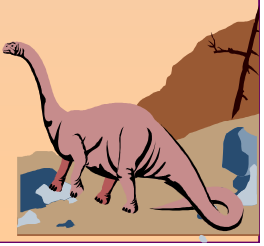
Uno stato non sicuro in un grafo di assegnazione delle risorse

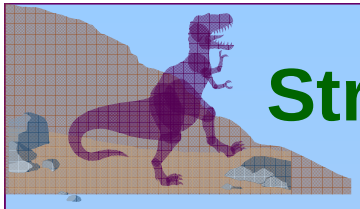




Algoritmo del banchiere

- Istanze multiple.
- Ciascun processo deve dichiarare a priori il numero massimo delle istanze di ciascun tipo di risorsa di cui necessita.
- Quando un processo richiede una risorsa, può dover attendere.
- Quando un processo ottiene tutte le sue risorse, deve restituirle entro un intervallo di tempo definito.



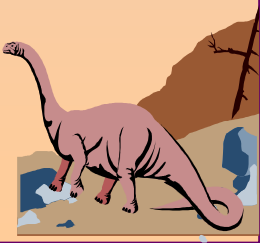


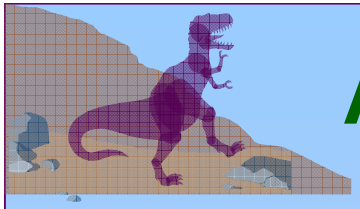
Strutture dati per l'algoritmo del banchiere

Sia *n* il numero di processi del sistema e *m* il numero dei tipi di risorsa.

- **Disponibili:** un vettore di lunghezza *m* indica il numero delle istanze disponibili per ciascun tipo di risorsa. Se *disponibili[j] = k*, significa che sono disponibili *k* istanze del tipo di risorsa R_j.
- **Massimo**: una matrice *n x m* definisce la richiesta massima di ciascun processo. *Massimo* [*i,j*] = *k* significa che il processo P_i può richiedere al più *k* istanze del tipo di risorsa R_j.
- **Assegnate**: una matrice *n x m* definisce il numero delle istanze di ciascun tipo di risorsa attualmente assegnate a ciascun tipo di processo. *Assegnate*[*i,j*] = *k* significa che al processo P_i sono assegnate *k* istanze del tipo di risorsa R_j.
- **Necessità**: una matrice *n* x *m* indica la necessità residua di risorse relativa a ogni processo. *Necessità*[*i,j*] = *k* significa che il processo P_i può aver bisogno di altre *k* istanze di R_j per completare il suo compito.

Necessità [*i,j*] = *Massimo*[*i,j*] – *Assegnate* [*i,j*].





Algoritmo di verifica della sicurezza

1. Siano *Lavoro* e *Fine* vettori di lunghezza rispettivamente *m* e *n*. Inizializza:

   Lavoro = Disponibile

   Fine[*i*] = *falso* per *i* - 1,3, …, *n.*

2. Cerca un indice *i* tale che:

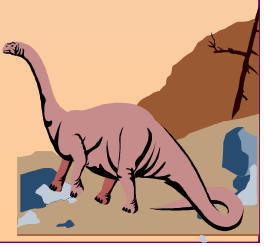
   (a) *Fine*[*i*] = *falso*

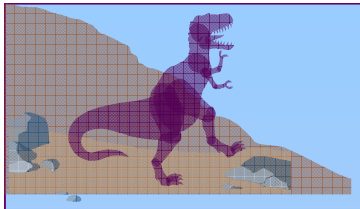
   (b) $Necessità_i \leq Lavoro$

   se tale *i* non esiste, esegue il passo 4.

3. *Lavoro* := *Lavoro* + $Assegnate_i$
   Fine[*i*] := *vero*
   torna al passo 2.

4. Se *Fine*[*i*] == *vero* per ogni *i*, allora il sistema è in uno stato sicuro.





Algoritmo di richiesta delle risorse

Richieste = vettore delle richieste per il processo P_i. Se $Richieste_i[j] = k$ allora il processo P_i richiede k istanze del tipo di risorsa R_j.

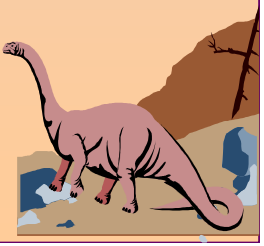
1. Se $Richieste_i \leq Necessità_i$ esegue il passo 2. Altrimenti, riporta una condizione d'errore, poiché il processo ha superato il numoero massimo di richieste.
2. Se $Richieste_i \leq Disponibiloi$, gesegue il passo 3. Altrimenti P_i deve attendere, poiché le risorse non sono disponibili.
3. Il sistema simula l'assegnazione al processo P_i delle risorse richieste modificando lo stato di assegnazione delle risorse:

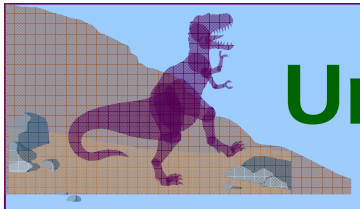
$$Disponibili := Disponibili = Richieste_i;$$

$$Assegnate := Assegnate_i + Richieste;$$

$$Necessità_i := Necessità_i – Richieste_i.$$

- *se lo stato è sicuro $\Rightarrow$ tle risorse sono assegnate a P_i.*
- *se lo stato è non sicuro $\Rightarrow$ P_i deve attendere, e si ripristina il vecchio stato di assegnazione delle risorse.*

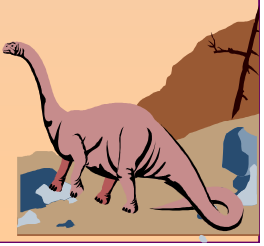


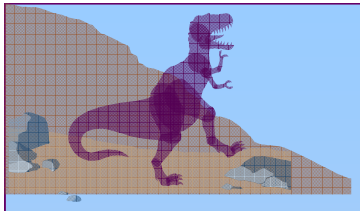


Un esempio di algoritmo del banchiere

- 5 processi da P_0 a P_4; 3 tipi di risorse: il tipo di risorse *A* ha 10 istanze, B ne ha 5i e *C* 7.
- Si supponga che all'istante T_0 si sia verificata la seguente istantanea:

	Assegnate	*Massimo*	*Disponibili*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	





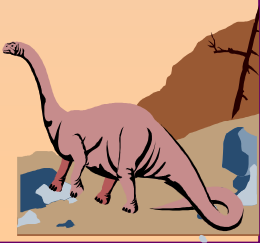
Un esempio *(Cont.)*

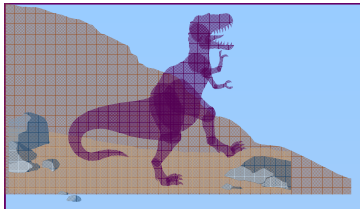
- Il contenuto della matrice *Necessità* è definito come *Massimo – Assegnate*.

Necessità

	A	*B*	*C*
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

- Il sistema si trova attualmente in uno stato sicuro poiché la sequenza < P_1, P_3, P_4, P_2, P_0> soddisfa i criteri di sicurezza.



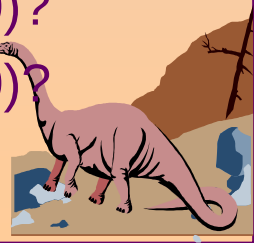


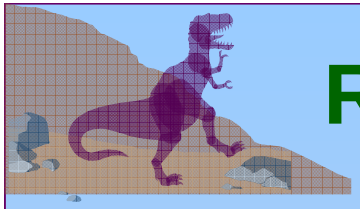
Un esempio *(Cont.)*

- Si verifica la condizione *Richieste* ≤ *Disponibili* (vale a dire, (1,0,2) ≤ (3,3,2) ⇒ *vera.*

	Allocation	*Necessità*	*Disponibili*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 1	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- L'esecuzione dell'algoritmo di verifica della sicurezza mostra che la sequanza <P_1, P_3, P_4, P_0, P_2> soddisfa il requisito di sicurezza.
- Può essere soddisfatta la richiesta da parte di P_4 di (3,3,0)?
- Può essere soddisfatta la richiesta da parte di P_0 di (0,2,0)?

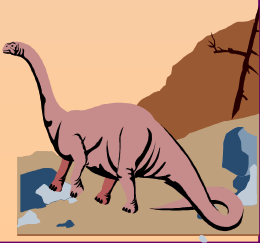


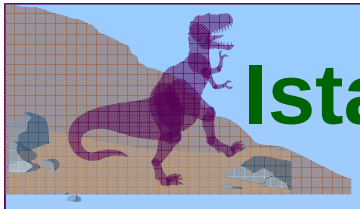


Rilevamento delle situazioni di stallo

Se un sistema non si avvale di un algoritmo per prevenire o per evitare le situazioni di stallo, è possibile che una situazione di stallo si verifichi effettivamente. In tal caso il sistema deve fornire i seguenti algoritmi:

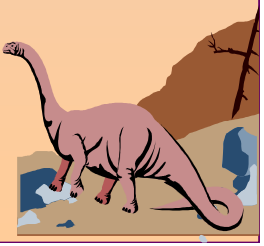
- un algoritmo che esamini lo stato del sistema per stabilire se si è verificato uno stallo
- un algoritmo che ripristini il sistema dalla condizione di stallo.



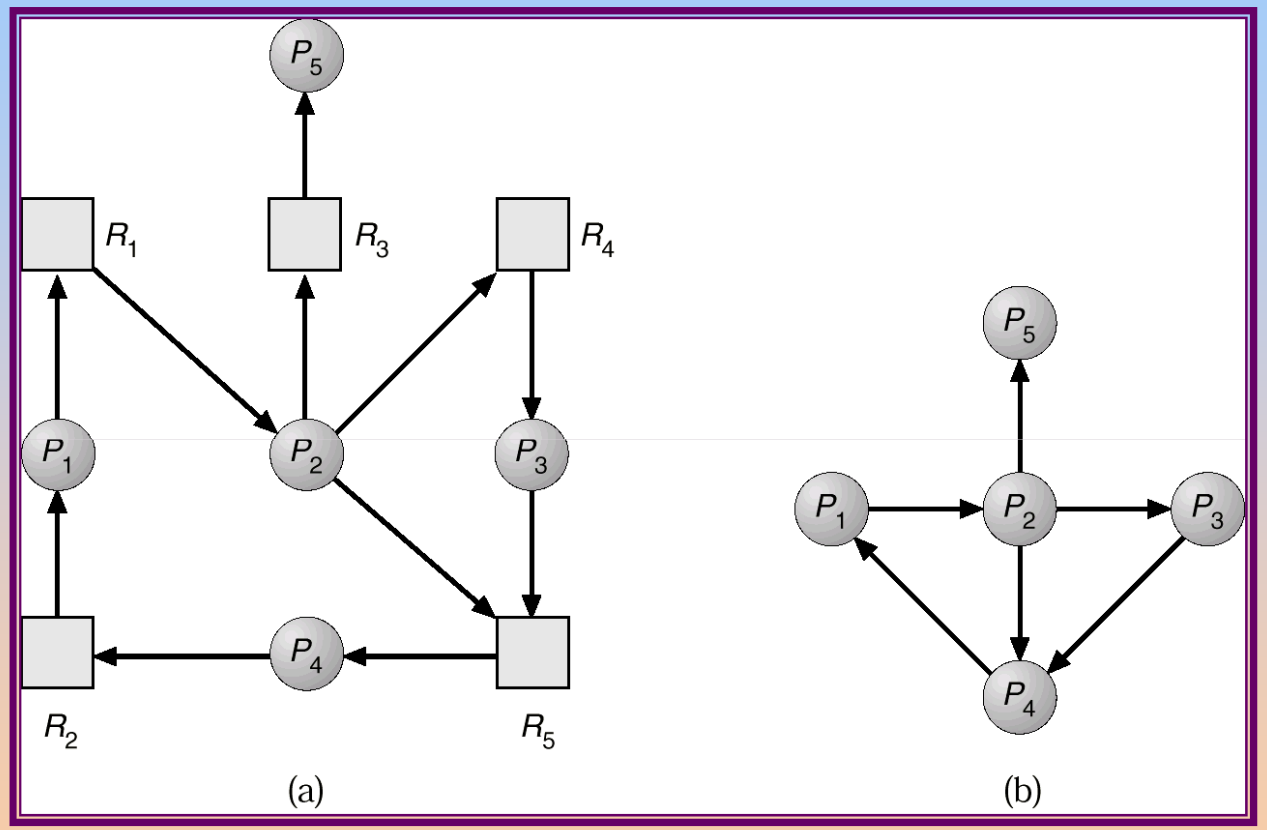


Istanza singola di ciascun tipo di risorsa

- Grafo d'attesa
  - I nodi sono processi.
  - $P_i \rightarrow P_j$ se P_i è in attesa di P_j.
- Per individuare le situazioni di stallo il sistema deve conservare il grafo d'attesa e invocare periodicamente un algoritmo che cerchi un ciclo all'interno del grafo.
- L'algoritmo per il rilevamento di un ciclo all'interno di un grafo richiede un numero di operazioni dell'odine di n^2, dove con n si indica il numero dei vertici del grafo.

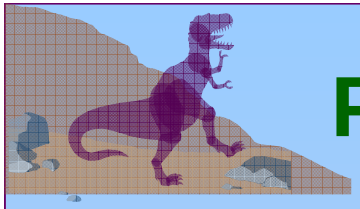


Grafo di assegnazione delle risorse e grafo d'attesa



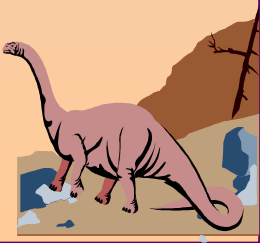
Grafo di assegnazione delle risorse

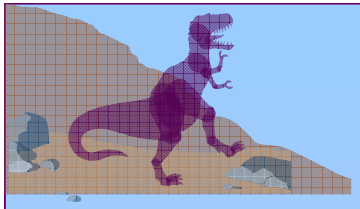
Grafo d'attesa corrispondente



Più istanze di ciascun tipo di risorsa

- *Disponibili.* Vettore di lunghezza *m* che indica il numero delle istanze disponibili per ciascun tipo di risorsa.
- *Assegnate.* Matrice *n* x *m* che definisce il numero delle istanze di ciascun tipo di risorse correntemente assegnate a ciascun processo.
- *Richieste.* Matrice *n* x *m* che indica la richiesta attuale di ciascun processo. Se *Richieste*[i_j] = *k*, significa che il processo P_i sta richiedendo altre *k* istanze del tipo di risorsa R_j.

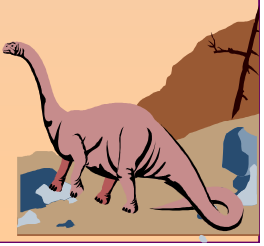


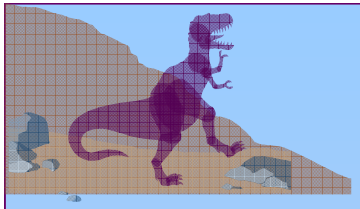


Algoritmo di rilevamento

1. Siano *Lavoro* e *Fine* due vettori di lunghezza rispettivamente *m* e *n*. Inizializza:
   (a) *Lavoro* = *Disponibili*
   (b) per *i* = 1,2, …, *n*, se *Assegnate* $\neq$ 0, allora *Fine*[i] = falso; altrimenti, *Fine*[i] = *vero*.
2. Cerca un indice *i* tale che:
   (a) *Fine*[*i*] == *falso*
   (b) $Richieste_i \leq Lavoro$

   Se tale *i* non esiste, esegue il passo 4.



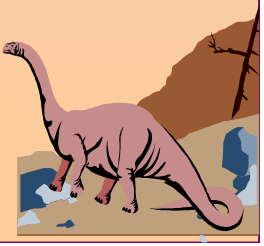


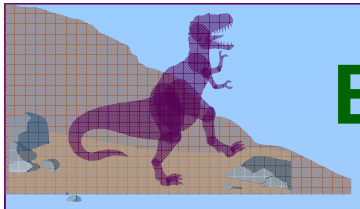
Algoritmo di rilevamento *(Cont.)*

3. *Lavoro* := *Lavoro* + *Assegnate*$_i$
   Fine[*i*] = *vero*
   torna al passo 2.

4. Se *Fine*[*i*] == falso, per qualche *i*, $1 \leq i \leq n$, allora il sistema è in stallo. Inoltre, se *Fine*[*i*] == *falso*, allora il processo P_i è in stallo.

Tale algoritmo richiede un numero di operazioni dell'ordine di *m* x n^2 per controllare se il sistema è in stallo.



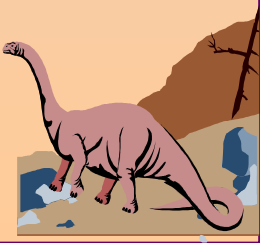


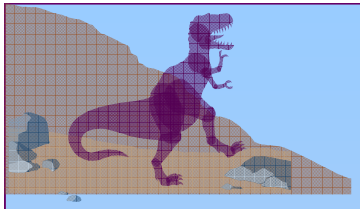
Esempio di algoritmo di rilevamento

- Cinque processi da P_0 a P_4; tre tipi di risorse: *A* (7 istanze), *B* (2 istanze) e *C* (6 istanze).
- Si supponga di avere, all'istante T_0:

	Assegnate	*Richieste*	*Disponibili*
	A B C	*A B C*	*A B C*
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

- La sequenza $<P_0, P_2, P_3, P_1, P_4>$ da come risultato *Fine*[*i*] = vero per ogni *i*.



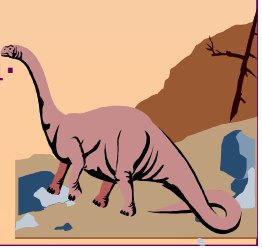


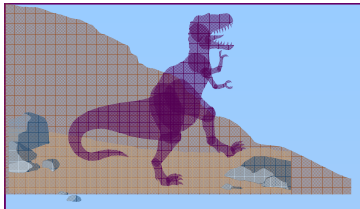
Esempio *(Cont.)*

- Si supponga ora che il processo P_2 richieda un'altra istanza di tipo *C*.

	Richieste		
	A	*B*	*C*
P_0	0	0	0
P_1	2	0	1
P_2	0	0	1
P_3	1	0	0
P_4	0	0	2

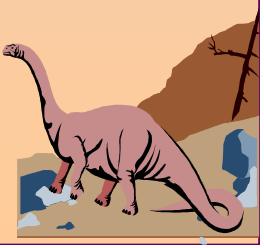
- Stato del sistema
  - Ora il sistema è in stallo Anche se si possono reclamare le risorse possedute dal processo P_0, il numero delle risorse disponibili non è sufficiente per soddisfare le richieste degli altri processi.
  - Si verifica uno stallo composto dei processi P_1, P_2, P_3 e P_4.

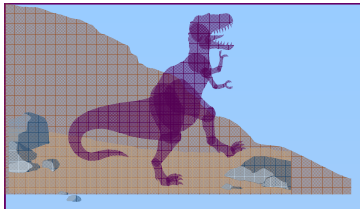




Uso dell'algoritmo di rilevamento

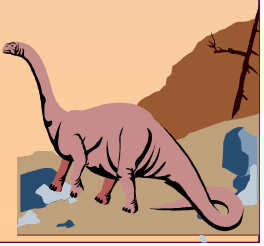
- Per sapere quando è necessario ricorrere all'algoritmo di rilevamento, occorre considerare i seguenti fattori:
  - *frequenza* (presunta) con la quale si verifica uno stallo
  - *numero* dei processi che sarebbero influenzati da tale stallo
- Non è conveniente richiedere l'algoritmo di rilevamento in momenti arbitrari, poiché nel grafo delle risorse possono coesistere molti cicli e, normalmente, non si può dire quale fra i tanti processi in stallo abbia "causato" lo stallo.

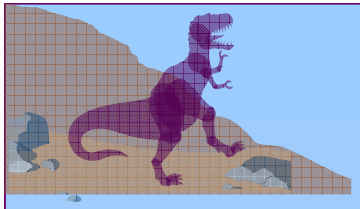




Ripristino da situazioni di stallo: terminazione di processi

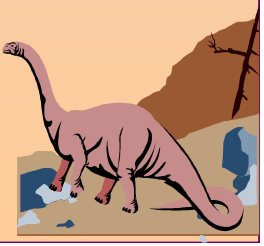
- Terminazione di tutti i processi in stallo.
- Terminaione di un processo alla volta fino all'eliminazione del ciclo di stallo.
- In quale ordine effttuare la terminazione?
  - Priorità dei processi.
  - Tempo trascorso dalla computazione e tempo ancora necessario per completare i compiti assegnati ai processi.
  - Quantità e tipo di risorse impiegate dai processi.
  - Quantità di ulteriori risorse di cui i processi hanno ancora bisogno per completare i propri compiti.
  - Numero di processi che si devono terminare.
  - Tipo di processi: interattivi o a lotti.

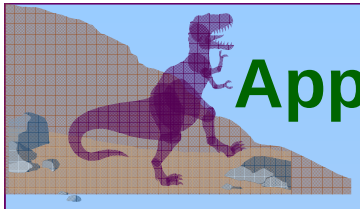




Ripristino da situazioni di stallo: prelazione di risorse

- Selezione di una vittima: minimizzare i costi.
- Rollback: ristabilimento di unprecedente stato sicuo, dal quale il processo può essere riavviato.
- Attesa indefinita (*starvation*): può accadere che si scelga sempre lo stesso processo come vittima.





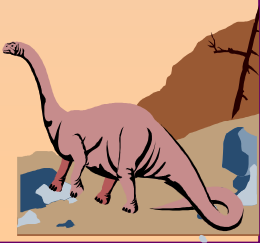
Approccio combinato per la gestione dello stallo

- La combinazione dei tre approcci di base
  - prevenire
  - evitare
  - rilevare

  consente l'uso dell'approccio ottimale per ciascuna delle risorse del sistema.

- Partizione delle risorse in classi ordinate gerarchicamente.

- Uso della tecnica più appropriata per la gestione dello stallo all'interno di ciascuna classe.



Stallo di traffico automobilistico per l’Esercizio 8.4