SIMULATORE DI ALGORITMI PER LA PREVENZIONE ELA RILEVAZIONE DEI DEADLOCK

Avesani Federica
Docente: Pravadelli Graziano

Indice

Capitolo 1
Installazione ed Esecuzione

1.1 Installazione

Scaricare il file *deadlock.tar.gz* dalla pagina del corso di Sistemi Operativi.
Aprire il terminale ed eseguire il seguente comando per estrarre il file:
tar -x deadlock.tar.gz e salvare il file *deadlock.jar* estratto nella propria home.

1.2 Esecuzione

Controllare che sulla macchina sia presente la versione *1.5.0* di *jdk* o maggiore.
Da terminale aprire la cartella dove è stato salvato il file *deadlock.jar* estratto nella fase di installazione e lanciare il programma con il comando **java -jar deadlock.jar**

Il seguente menu testuale verrà visualizzato:

```
*** Scegli l'algoritmo da eseguire ***
   1) Grafo di Allocazione Risorse (RAG)
   2) Algoritmo del Banchiere
   3) Grafo di Allocazione Risorse con Archi di Reclamo
   4) Algoritmo di Rilevazione
Inserisci il numero dell'algoritmo scelto:
```

Capitolo 2
Esecuzione degli Algoritmi

2.1 Grafo di Allocazione Risorse (RAG)

I grafi di allocazione risorse sono uno strumento per verificare se una sequenza di allocazione porta ad un deadlock.
Se il grafo non contiene cicli non c'è nessun deadlock. Se il grafo contiene un ciclo occorre distinguere due casi:

- se ogni risorsa nel ciclo ha una istanza, siamo in presenza di deadlock
- se ci sono più istanze per ogni risorsa implicata nel ciclo dipende dallo schema di allocazione.

Per eseguire questo algoritmo digitare il numero 1 e premere invio.
Le seguenti richieste sono visualizzate per l'inserimento dei dati:

```
Inserisci il numero di processi:
Inserisci il numero di risorse:
```

Per ogni risorsa inserire il numero di istanze che possiede:

```
Inserisci il numero di istanze per la risorsa 1:
...
Inserisci il numero di istanze per la risorsa n:
```

Con i dati vengono inizializzate le strutture dati per gestire le richieste e gli assegnamenti.
E' possibile ora inserire le richieste:

```
Inserimento delle RICHIESTE:

Inserisci il processo che effettua la richiesta:
Inserisci la risorsa richiesta:
Ci sono altre richieste(y/n)?
```

Finché ci sono richieste digitare *y*, quando non ci sono più richieste digitare *n*.
Il passo successivo è l'inserimento degli assegnamenti:

```
Inserimento degli ASSEGNAMENTI

Inserisci la risorsa assegnata:
Inserisci il processo che la detiene:
Ci sono altri assegnamenti(y/n)?
```

Finché ci sono richieste digitare *y*, quando non ci sono più richieste digitare *n*.

Tutte le richieste e tutti gli assegnamenti verranno stampati con gli archi da processi a risorse quando il processo richiede la risorsa e con gli archi da risorsa a processo se il processo detiene la risorsa.
Per esempio:

```
RICHIESTE:
P1->R1, P2->R2, P3->R3;
ASSEGNAMENTI:
R1->P2, R2->P3, R3->P1, R3->P2;
```

L'algoritmo analizza la presenza o meno di cicli. Ogni ciclo trovato verrà nominato da *1* a *n*.

Se ci sono cicli con una istanza per risorsa la seguente frase verrà visualizzata:

```
Le risorse implicate nel ciclo n hanno una sola istanza ed impegnata.
Il sistema si trova in uno stato di deadlock formato dai processi del
ciclo n.
```

Se ci sono più istanze per risorsa e tutte le istanze sono impegnate in un ciclo la seguente frase verrà visualizzata:

```
Il ciclo n può causare deadlock poiché tutte le istanze delle risorse
implicate nel ciclo sono state assegnate a qualche processo, ma non è
detto.
```

Se non sono tutte assegnate la seguente frase verrà visualizzata:

```
Non tutte le istanze delle risorse implicate nel ciclo sono assegnate
Il ciclo n non può generare deadlock
```

Prima di procedere con il controllo dei cicli che possono formare deadlock verrà controllato se ci sono processi, che detengono risorse, e non fanno parte ci cicli.
Se ci sono la seguente frase verrà visualizzata:

```
Il processo Pi non è in nessun ciclo
```

Non appartenendo a nessun ciclo la risorsa verrà liberata e l'algoritmo procederà con l'eliminare i cicli che contengono la risorsa appena liberata. Questo passo viene ripetuto per tutte le risorse liberate.

Tutti i cicli rimanenti portano ad uno stato di deadlock e verrà stampato l'elenco dei cicli che portano allo stato di deadlock.
Per esempio:

```
DEADLOCK FORMATO DAI CICLI:
Ciclo 1: P1 -> R1 -> P2 -> R2 -> P3 -> R3 -> P1
Ciclo 2: P2 -> R2 -> P3 -> R3 -> P2
```

Se non ci sono cicli la seguente frase verrà visualizzata:

```
Il sistema si trova in uno stato SAFE
```

2.2 Algoritmo del Banchiere

A differenza del Grafo di Allocazione risorse che non può essere applicato a sistemi di allocazione delle risorse con più istanze di ciascun tipo di risorsa, l'Algoritmo del Banchiere può essere applicato a tali sistemi.
Quando un processo entra nel sistema, deve dichiarare il numero massimo delle istanze di ciascuna risorsa di cui necessita; questo numero non può superare il totale delle risorse del sistema.
Quando un utente richiede un gruppo di risorse, il sistema deve stabilire se l'allocazione di queste risorse lo lasci in stato sicuro. Se tale condizione è verificata, le risorse vengono allocate, altrimenti il processo deve attendere fino a che qualche altro processo rilascia un numero sufficiente di risorse.

Le seguenti richieste sono visualizzate per l'inserimento dei dati:

```
Inserisci il numero di processi:
Inserisci il numero di risorse:
```

Con i dati vengono inizializzate le strutture dati per inserire i dati della matrice Allocation e Max per le quali viene richiesto l'inserimento dei dati per ogni singolo processo.

```
Inserisci i dati per la matrice Allocation:
Inserisci le risorse per P0:
...
Inserisci le risorse per Pn:
```

Per esempio per 4 processi e 3 risorse l'inserimento avviene nel seguente modo:

```
Inserisci il numero di processi: 4
Inserisci il numero di risorse: 3

Inserisci i dati per la matrice Allocation:
Inserisci le risorse per P0: 0 1 4
Inserisci le risorse per P1: 2 0 1
Inserisci le risorse per P2: 1 2 1
Inserisci le risorse per P3: 1 0 3
```

Le risorse devono essere inserite separate da uno spazio. Se vengono inserite più risorse di quelle dichiarate il sistema ritorna il seguente errore:

```
Sono state inseriti troppi valori rispetto a quelli dichiarati.
Reinserisci l'ultimo dato:
```

Se viene inserito un valore non numerico il seguente errore viene visualizzato e viene richiesto l'inserimento dell'ultimo dato per procedere con l'esecuzione:

```
Sono stati inseriti dei valori non validi.
Reinserisci l'ultimo dato:
```

Finito l'inserimento dei dati per la matrice Allocation viene richiesto l'inserimento dei dati per la matrice Max per ogni singolo processo:

```
Inserisci i dati per la matrice Max:
Inserisci i dati per il processo P0: R1 .. Rn
...
Inserisci i dati per il processo Pn: R1 .. Rn
```

Per esempio con 4 processi e 3 risorse l'inserimento avviene nel seguente modo:

```
Inserisci i dati per la matrice Max:
Inserisci i dati per il processo P0: 4 1 4
Inserisci i dati per il processo P1: 3 1 4
Inserisci i dati per il processo P2: 5 7 13
Inserisci i dati per il processo P3: 1 1 6
```

Anche in questo caso le risorse devono essere inserite separate da uno spazio. Se vengono inseriti dati per più risorse rispetto a quelle dichiarate o dati non numerici il programma ritorna gli stessi errori visti per la matrice Allocation.

Il programma calcola automaticamente il contenuto della matrice Need e insieme ai dati inseriti per le matrici Allocation e Max viene visualizzato nel seguente modo:

```
I dati inseriti per Allocation sono
P0  R1 R2 ... Rn
...
Pn R1 R2 ... Rn

I dati inseriti per Max sono
P0  R1 R2 ... Rn
...
Pn R1 R2 ... Rn
```

```
La matrice Need (Max - Allocation) risultante è:
P0  R1 R2 ... Rn
...
Pn R1 R2 ... Rn
```

Per esempio:

```
I dati inseriti per Allocation sono
P0  0 1 4
P1  2 0 1
P2  1 2 1
P3  1 0 3

I dati inseriti per Max sono
P0  4 1 4
P1  3 1 4
P2  5 7 13
P3  1 1 6

La matrice Need (Max - Allocation) risultante è:
P0  4 0 0
P1  1 1 3
P2  4 5 12
P3  0 1 3
```

Viene ora richiesto l'inserimento delle risorse disponibili al tempo 0:

```
Inserisci le risorse disponibili al tempo 0:
```

Prima di iniziare l'esecuzione vengono creati due nuovi vettori:
un vettore boolean finish[] inizializzato a false e un vettore work[] che inizialmente corrisponde alle risorse disponibili al tempo 0.

A questo punto l'algoritmo comincia l'esecuzione per verificare se con i dati inseriti il sistema si trova in uno stato SAFE.

A partire dal processo 0, per ogni processo viene verificato se le risorse possono essere allocate con le risorse disponibili.
I controlli che vengono effettuati sono i seguenti:

```
Controllo se le risorse del processo Pi possono essere allocate:
```

se finish[i]=false && need[i]<work le risorse possono essere allocate e viene visualizzato, per esempio, il seguente messaggio:

```
i=1 finish[1]=false, need[1][]=(1 1 3) <= work[]=(1 5 7)
Sommo le risorse allocate a P1 al vettore work[]:
work[] = work[]+alloc[1][]
work[] = 1 5 7 + 2 0 1 = 3 5 8
finish[1]=true
```

Se le risorse non possono essere allocate il seguente messaggio viene visualizzato:

```
Non possono essere allocate le risorse per il processo 'Pi'
```

Per esempio

```
Non possono essere allocate le risorse per il processo 'P0'
i=0  finish[0]=false
need[0][]>work
4 0 0 > 1 5 7
```

finish[i]=true indica che le risorse del processo i sono già state allocate nei passi precedenti.
Il messaggio restituito è il seguente:

```
Le risorse per il processo 'Pi' sono già state allocate.
```

Se l'algoritmo porta ad uno stato SAFE viene stampata la sequenza di assegnazione delle risorse ai processi:

```
Sequenza SAFE:
< P1, P3, P0, P2 >
```

Se l'allocazione porta ad uno stato UNSAFE viene stampato il seguente messaggio:

```
DEADLOCK formato dai Processi:
Pi, ... , Pj
```

Se il sistema rilava uno stato SAFE è possibile soddisfare ulteriori richieste da parte dei processi.
Viene chiesto all'utente se ci sono ulteriori richieste tramite il seguente messaggio:

```
Ci sono altre richieste da parte di altri processi(y/n)?
```

Se viene digitato n l'algoritmo termina. Se ci sono altre richieste digitare y e premere invio.
Si può inserire una richiesta per volta nel seguente modo:

```
Inserisci il processo che ha effettuato la richiesta:
Inserisci le risorse richieste dal processo Pi:
```

Inseriti i dati vengono controllate 2 condizioni necessarie per verificare se la richiesta mantiene il

sistema in uno stato SAFE:

1) La richiesta è minore o uguale a need (req_vec[i] <= need[i])?
   `Controllo se req_vec[i] <= need[i][]`

Se la condizione 1 non è verificata il processo termina viene ripristinato lo stato precedente in cui il sistema si trovava in uno stato safe e il seguente messaggio viene visualizzato:

Dopo la visualizzazione di questo messaggio viene ripristinato lo stato in cui il sistema si trovava in uno stato safe e viene chiesto se ci sono ulteriori richieste.

Se la condizione è verificata il seguente messaggio viene visualizzato:

```
req_vec[i] <= need[i][]
Prima condizione verificata
1 0 0 <= 4 0 0
```

Si passa dunque al controllo della seconda condizione:

2) La richiesta è minore uguale della disponibilità (req_vec[i] <= avail[]) ?
   `Procedo con la verifica di req_vec[] <= avail[]`

Se la condizione 2 non è verificata il processo termina e il seguente messaggio viene visualizzato:
Dopo la visualizzazione di questo messaggio viene ripristinato lo stato in cui il sistema si trovava in uno stato safe e viene chiesto se ci sono ulteriori richieste.

Se la condizione 2 è verificata il seguente messaggio viene visualizzato:

```
Seconda condizione verificata.
req_vec[i] <= avail[]
```

Viene quindi calcolata la nuova situazione e viene stamapata a video.
Per esempio se il processo P0 richiede (1,0,0) vengono aggiornate la matrice ALLOC e il vettore AVAIL e la situazione rispetto a quella vista a pagina X è la seguente

```
ALLOC
P0 1 1 4
P1 2 0 1
P2 1 2 1
P3 1 0 3
MAX
P0 4 1 4
P1 3 1 4
P2 5 7 13
P3 1 1 6
NEED
P0 3 0 0
```

```
P1 1 1 3
P2 4 5 12
P3 0 1 3

Risorse disponibili (AVAIL): 0 5 7
```

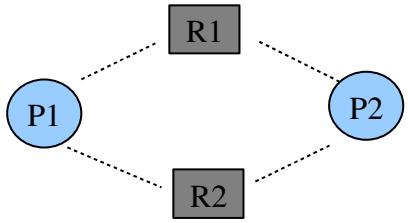
Viene quindi effettuata una simulazione per vedere se con questa richiesta il sistema si trova in uno stato SAFE. I passi per la verifica dello stato safe sono uguali ai precedenti.
Diversa è la situazione se il sistema si trova in uno stato UNSAFE. Determinato lo stato di deadlock il sistema ripristina la situazione allo stato safe precedente e la richiesta viene ignorata.
Viene richiesto se ci sono ulteriori richieste partendo dallo stato safe precedente.
Se dopo la richiesta il sistema mantiene uno stato SAFE la richiesta viene resa effettiva e successive richieste partono da questo nuovo stato.

2.3 Grafo di Allocazione Risorse con Archi di Reclamo

Rispetto al Grafo di Allocazione Risorse visto nel paragrafo 2.1 viene introdotto un altro tipo di arco nel grafo di allocazione: l'arco di reclamo rappresentato con
linea tratteggiata.
Un arco di reclamo
Pi - -> Rj indica che il processo Pj può richiedere la risorsa Rj.

Quando un processo richiede effettivamente una risorsa l'arco di reclamo diventa un arco di richiesta

Questo algoritmo funziona solamente con 1 istanza per risorsa e la risorsa viene soddisfatta solamente se l'allocazione non genera un ciclo.
Esempio svolto:



In questo esempio il processo P1 reclama le risorse R1 e R2 e il processo P2 reclama le risorse R1 e R2. Vediamone in dettaglio lo svolgimento:

Selezionando l'algoritmo 3 la seguente schermata viene visualizzata:

```
*** GRAFO DI ALLOCAZIONE RISORSE CON ARCHI DI RECLAMO   ***
```

Viene chiesto di inserire il numero di processi e di risorse per inizializzare le strutture dati che

verranno utilizzate durante lo svolgimento:

```
Inserisci il numero di processi: 2
Inserisci il numero di risorse: 2
```

Il secondo passo consiste nell'inserire, per ogni processo, tutte le risorse che intendono utilizzare durante la loro esecuzione:

```
Inserisci le risorse che i processi intendono usare durante la loro
esecuzione:

Risorse reclamate dal processo P1 (separate da uno spazio): 1 2
Risorse reclamate dal processo P2 (separate da uno spazio): 1 2
```

Il sistema è ora in attesa delle richieste da parte dei processi:

```
 ***   INSERIMENTO RICHIESTE   ***

Inserisci il processo che effettua la richiesta: 1
Quale risorsa chiede il processo P1? 2
```

Se la risorsa non è stata reclamata il seguente messaggio viene visualizzato:

```
Il processo non ha reclamato la risorsa all'inizio dell'esecuzione
Inserisci il processo che effettua la richiesta:
```

Se viene inserito un dato non corretto, il numero un processo o di una risorsa non esistente, vengono visualizzati rispettivamente i seguenti messaggi:

```
Inserisci il processo che effettua la richiesta: 3
Dato inserito non corretto. Sono stati dichiarati 2 processi.
Inserisci il processo che effettua la richiesta:

Inserisci il processo che effettua la richiesta: 1
Quale risorsa chiede il processo P1? 3
Dato inserito non corretto. Sono state dichiarate 2 risorse.
Quale risorsa chiede il processo P1?
```

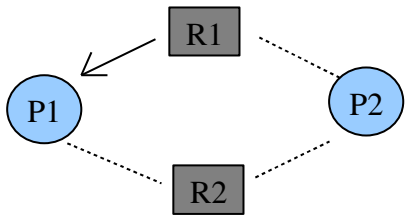
Supponiamo che le richieste arrivino nel seguente ordine:
P1 -> R1, P2 -> R1, P2 -> R2

```
 ***   INSERIMENTO RICHIESTE   ***

Inserisci il processo che effettua la richiesta: 1
Quale risorsa chiede il processo P1? 1
```

La risorsa è stata reclamata e i dati inseriti sono corretti. Procedo con l'esecuzione dell'algoritmo.
Se la risorsa non è mai stata richiesta da nessun processo il seguente messaggio viene visualizzato:

```
La risorsa R1 è stata assegnata al processo P2.
```

Graficamente:



Viene quindi simulata la richiesta di tutti i reclami non ancora soddisfatti. Se l'aggiunta di un qualsiasi reclamo porta ad un ciclo la richiesta non potrà essere soddisfatta.

```
Ora verranno simulati tutti i reclami rimanenti per verificare se una
successiva richiesta puÃ² portare in uno stato UNSAFE
Se un arco di reclamo forza un ciclo non viene effettuato l'assegnamento

*** INIZIO SIMULAZIONE  ***

Il processo P1 ha richiesto la risorsa R1(già assegnata ad un altro
processo nei passi precedenti).
Controllo se la simulazione ha generato un ciclo...
----> Non c'è ciclo con la simulazione: P1 --> R1
La risorsa R2 è stata assegnata al processo P1.
Controllo se la simulazione ha generato un ciclo...
----> Non c'è ciclo con la simulazione: P1 --> R2
La risorsa R2 è stata assegnata al processo P2.
Controllo se la simulazione ha generato un ciclo...
----> Non c'è ciclo con la simulazione: P2 --> R2

*** SIMULAZIONE  TERMINATA***


Simulazione andata a buon fine. Una qualsiasi richiesta successiva
manterrà il sistema in uno stato SAFE.
```

Viene quindi stampato l'elenco delle richieste (o assegnamenti) soddisfatti:

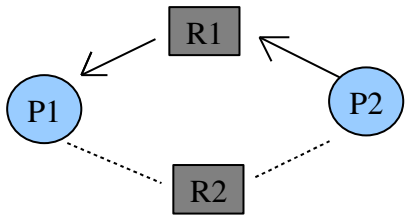
```
Situazione attuale:
ASSEGNAMENTI
R1 -> P2;
```

```
Ci sono altre richieste(y/n)?
```

Se non ci sono richieste digitare n e l'algoritmo verrà terminato. Se ci sono ulteriori richieste digitare y. Ora il processo P2 richiede la risorsa R1:

```
Inserisci il processo che effettua la richiesta: 2
Quale risorsa chiede il processo P2? 1
```

La risorsa R1 è già stata assegnata al processo P1. Il processo P2 richiede quindi la risorsa R1. Graficamente la situazione è la seguente:



A video viene il seguente messaggio viene visualizzato:

```
Il processo P2 ha richiesto la risorsa R1(già assegnata ad un altro
processo nei passi precedenti).

 Ora verranno simulati tutti i reclami rimanenti per verificare se una
successiva richiesta può portare in uno stato UNSAFE

*** INIZIO SIMULAZIONE  ***

La risorsa R2 è stata assegnata al processo P1.
Controllo se la simulazione ha generato un ciclo...
----> Non c'è ciclo con la simulazione: P1 --> R2
La risorsa R2 è stata assegnata al processo P2.
Controllo se la simulazione ha generato un ciclo...
----> Non c'è ciclo con la simulazione: P2 --> R2

*** SIMULAZIONE  TERMINATA***
```

Al termine della simulazione viene ripristinato lo stato precedente.
Se la simulazione è andata a buon fine viene resa effettiva l'ultima richiesta altrimenti l'algoritmo termina

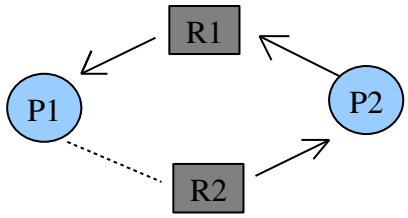
```
Simulazione andata a buon fine. Una qualsiasi richiesta successiva
manterrà lo stato SAFE.

Situazione attuale:
RICHIESTE
```

```
P2 -> R1;
ASSEGNAMENTI
R1 -> P1;

Ci sono altre richieste(y/n)?
```

Successiva richiesta: P2 -> R2

```
Inserisci il processo che effettua la richiesta: 2
Quale risorsa chiede il processo P2? 2

La risorsa R2 è stata assegnata al processo P2.
```

Graficamente la situazione è la seguente:



Come si vede chiaramente dal grafo disegnato una successiva richiesta forma un ciclo.
La simulazione non andrà quindi a buon fine e la richiesta non potrà essere soddisfatta.

```
Ora verranno simulati tutti i reclami rimanenti per verificare se una
successiva richiesta può portare in uno stato UNSAFE
Se un arco di reclamo forza un ciclo non viene effettuato l'assegnamento

*** INIZIO SIMULAZIONE  ***

Il processo P1 ha richiesto la risorsa R2(già assegnata ad un altro
processo nei passi precedenti).
Controllo se la simulazione ha generato un ciclo...
La simulazione P1 --> R2 causa un CICLO

*** SIMULAZIONE  TERMINATA***

Simulazione non andata a buon fine. Almeno una delle richieste
successive porta ad uno stato UNSAFE.
```

La richiesta non viene soddisfatta e la situazione finale risulta la medesima descritta nel passo precedente.

```
Situazione finale:
RICHIESTE
P2 -> R1;
```

```
ASSEGNAMENTI
R1 -> P1;
```

L'algoritmo viene terminato e il seguente messaggio viene visualizzato:

```
Algoritmo Grafo di Allocazione Risorse con Archi di reclamo terminato.
   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***
```

2.3 Algoritmo di Rilevazione

L'algoritmo di rilevazione è basato sull'esplorazione di ogni possibile sequenza di allocazione per i processi che non hanno ancora terminato

.

Se la sequenza va a buon fine (safe) non c'è deadlock

La struttura dati è molto simile a quella dell'algoritmo del banchiere:

int available[m]: numero di istanze di Ri disponibili;
alloc[n][m]: matrice delle risorse allocate correntemente;
int req_vec[n][m]: matrice delle richieste;

Le seguenti richieste sono visualizzate per l'inserimento dei dati:

```
Inserisci il numero di processi:
Inserisci il numero di risorse:
```

Con i dati vengono inizializzate le strutture dati per inserire i dati della matrice Allocation e Request per le quali viene richiesto l'inserimento dei dati per ogni singolo processo.

```
Inserisci i dati per la matrice Allocation:
Inserisci le risorse per P0:
...
Inserisci le risorse per Pn:
```

Per esempio per 4 processi e 3 risorse l'inserimento avviene nel seguente modo:

```
Inserisci il numero di processi: 5
Inserisci il numero di risorse: 3

Inserisci i dati per la matrice Allocation:
Inserisci le risorse per P0: 0  1 5
Inserisci le risorse per P1: 2  2 1
Inserisci le risorse per P2: 1  2 1
```

```
Inserisci le risorse per P3: 1 10 3
Inserisci le risorse per P4: 1  2 3
```

Le risorse devono essere inserite separate da uno spazio. Se vengono inserite più risorse di quelle dichiarate il sistema ritorna il seguente errore:

```
Sono state inseriti troppi valori rispetto a quelli dichiarati.
Reinserisci l'ultimo dato:
```

Se viene inserito un valore non numerico il seguente errore viene visualizzato e viene richiesto l'inserimento dell'ultimo dato per procedere con l'esecuzione:

```
Sono stati inseriti dei valori non validi.
Reinserisci l'ultimo dato:
```

Finito l'inserimento dei dati per la matrice Allocation viene richiesto l'inserimento dei dati per la matrice Request per ogni singolo processo:

```
Inserisci i dati per la matrice Max:
Inserisci i dati per il processo P0: R1 .. Rn
...
Inserisci i dati per il processo Pn: R1 .. Rn
```

Per esempio con 5 processi e 3 risorse l'inserimento avviene nel seguente modo:

```
Inserisci i dati per la matrice Request:
Inserisci i dati per il processo P0: 4 1 4
Inserisci i dati per il processo P1: 1 1 4
Inserisci i dati per il processo P2: 5 1 9
Inserisci i dati per il processo P3: 1 1 3
Inserisci i dati per il processo P3: 0 0 0
```

Anche in questo caso le risorse devono essere inserite separate da uno spazio. Se vengono inseriti dati per più risorse rispetto a quelle dichiarate o dati non numerici il programma ritorna gli stessi errori visti per la matrice Allocation.

I dati inseriti per le matrici Allocation e Request vengono visualizzati nel seguente modo:

```
I dati inseriti per Allocation sono
P0  R1 R2 ... Rn
...
Pn R1 R2 ... Rn

I dati inseriti per Request sono
```

```
P0  R1 R2 ... Rn
...
Pn R1 R2 ... Rn
```

Per esempio:

```
I dati inseriti per Allocation sono
P0 0  1 5
P1 2  2 1
P2 1  2 1
P3 1 10 3
P4 1  2 3
I dati inseriti per Request sono
P0 4 1 4
P1 1 1 4
P2 5 1 9
P3 1 1 3
P3 0 0 0
```

Viene ora richiesto l'inserimento delle risorse disponibili al tempo 0:

```
Inserisci le risorse disponibili al tempo 0: 0 0 0
```

Supponiamo che le risorse disponibili al tempo 0 siano 0 per la risorsa R1, 0 per la risorsa R2, 0 per la risorsa R3. Lo svolgimento dell'esercizio in questo caso è il seguente:

Prima di iniziare l'esecuzione vengono creati due nuovi vettori:
un vettore booleano finish[] inizializzato a false e un vettore work[] che inizialmente corrisponde alle risorse disponibili al tempo 0.

A questo punto l'algoritmo comincia l'esecuzione per verificare se con i dati inseriti il sistema si trova in uno stato SAFE.

```
Copio il contenuto di AVAIL in WORK
work[] = avail[] = 0 0 0
```

Per ogni processo viene verificata la seguente condizione: !finish[i] && req_vec[i][] <= work[]

```
Condizione da verificare per procedere: !finish[i] && req_vec[i][] <=
work[]
```

se tale condizione è verificata il contenuto al vettore work vengono sommate le risorse allocate al processo preso in considerazione e viene visualizzato il seguente messaggio:

```
Condizione verificata
Sono state allocate le risorse per il processo Pi
```

```
finish[i] || req_vec[i][] <= work[]
  true    ||    0 0 0     <= 0 0 0
```

se la condizione non è verificata il seguente messaggio è visualizzato e si passa al processo successivo:

```
Condizione NON verificata
finish[i] || req_vec[i][] > work[]
  false   ||    5 1 9     > 1 2 3
```

Se la risorse per il processo Pi sono già state allocate il seguente messaggio viene visualizzato:

```
Le risorse per il processo Pi sono già state allocate.
```

Se l'allocazione delle risorse mantiene uno stato SAFE il seguente messaggio viene visualizzato:

```
Sequenza SAFE:
< P4, P3, P1, P0, P2 >
```

Altrimenti viene visualizzato il seguente messaggio:

```
DEADLOCK formato dai Processi:
Pi,.., Pn
```

in cui sono elencati tutti i processi per i quali non è stato possibile allocarne le risorse.
Finché ci sono sono nuove richieste il sistema visualizza il seguente messaggio.

```
Ci sono richieste da parte di altri processi(y/n)?
```

Se viene digitato y il sistema richiede l'inserimento dei nuovi dati:

```
Inserisci il processo che effettua la richiesta:
Inserisci le risorse richieste dal processo Pi:
```

Inseriti i dati viene aggiornata la matrice req_vec[][] in corrispondenza del processo Pi a cui vengono aggiunte le nuove risorse richieste.

```
Verifico se la nuova situazione porta in una situazione di DEADLOCK.
```

Se la richiesta mantiene lo stato SAFE i dati vengono aggiornati e lo svolgimento è lo stesso per il controllo dello stato SAFE iniziale, altrimenti viene ripristinato lo stato SAFE precedente e il seguente messaggio viene visualizzato:

```
Ripristino lo stato SAFE precedente per eventuali richieste successive.
Ci sono richieste da parte di altri processi(y/n)?
```

Se non ci sono più richieste digitare n. Il seguente messaggio viene visualizzato:

```
Algoritmo di Rilevazione terminato.
   ***   ***   ***   ***   ***
```