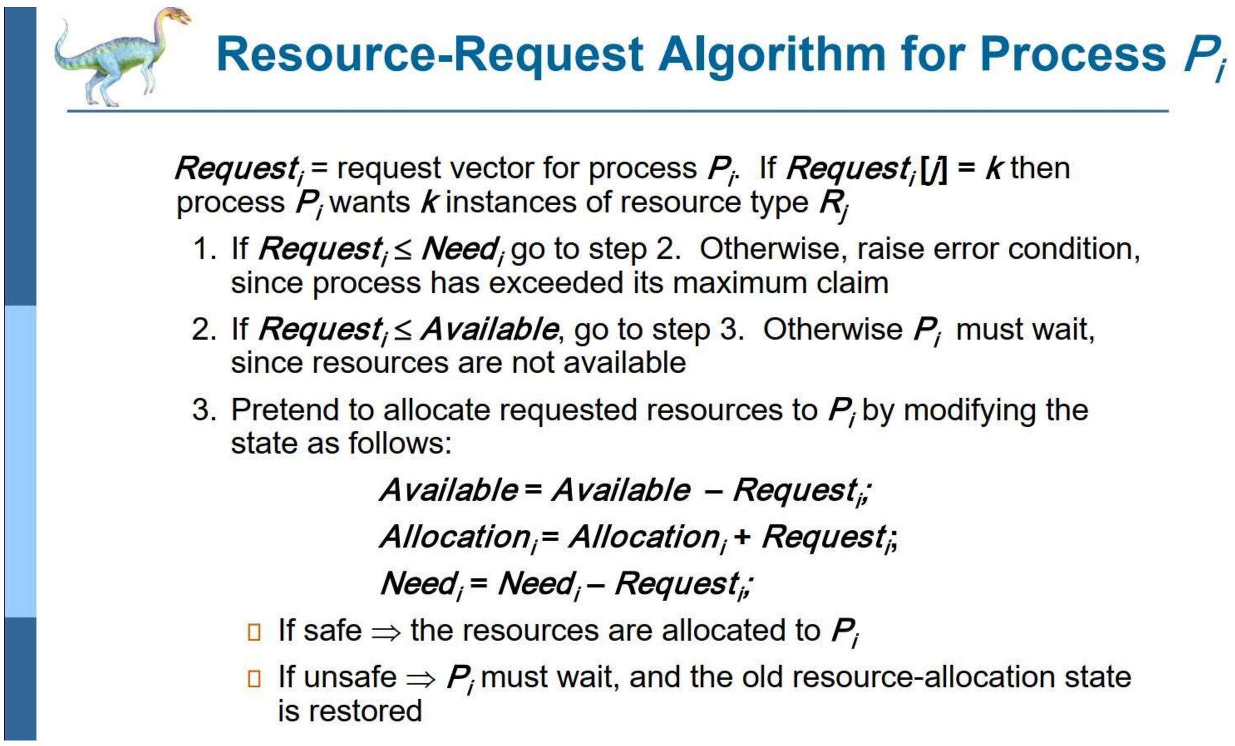
به نام خدا

تمرین چهارم تئوری سیستم عامل
فرزان رحمانی – ۹۹۵۲۱۲۷۱

۱. در حالت b و d بن بست وجود دارد.
با استفاده از الگوریتم زیر به حل سوال می پردازیم.



Resource-Request Algorithm for Process P_i

Request**$_i$* = request vector for process P_i. If ***Request**$_i$* **[*j*]** = ***k then process P_i wants ***k*** instances of resource type R_j

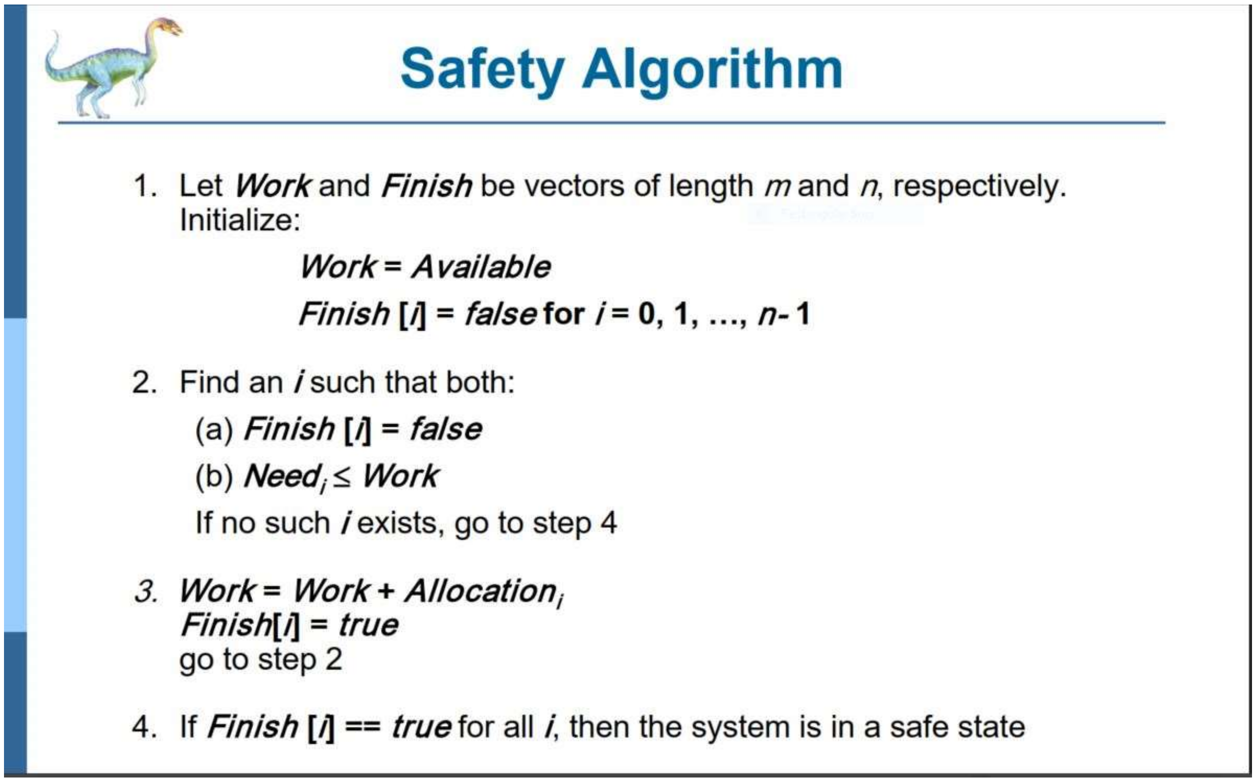
1. If ***Request**$_i$* ≤ ***Need**$_i$* go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim
2. If ***Request**$_i$* ≤ ***Available***, go to step 3. Otherwise P_i must wait, since resources are not available
3. Pretend to allocate requested resources to P_i by modifying the state as follows:

   Available = Available – Request**$_i$**;

   Allocation**$_i$**= Allocation**$_i$ **+ Request**$_i$**;

   Need**$_i$ **= Need**$_i$ **– Request**$_i$**;

   - If safe ⇒ the resources are allocated to P_i
   - If unsafe ⇒ P_i must wait, and the old resource-allocation state is restored



Safety Algorithm

1. Let ***Work*** and ***Finish*** be vectors of length *m* and *n*, respectively. Initialize:

   Work = Available

   ***Finish* [*i*] = *false* for *i* = 0, 1, …, *n*- 1**

2. Find an ***i*** such that both:

   (a) ***Finish* [*i*] = *false***

   (b) $\boldsymbol{Need_i \le Work}$

   If no such ***i*** exists, go to step 4

3. $\boldsymbol{Work = Work + Allocation_i}$
   Finish*[*i*] = *true
   go to step 2

4. If ***Finish* [*i*] == *true*** for all ***i***, then the system is in a safe state

(۱) فرض می کنیم منظور سؤال از اولین و آخرین واحد R۲ ، ترتیب برداشتن است. که در این صورت، فرقی برای نوبت برداشتن وجود ندارد و هر دو گزینه عملکرد یکسانی خواهند داشت.

Allocate

	R۰	R۱	R۲	R۳
P۰	۳	۰	۱	۱
P۱	۰	۱	۰	۰
P۲	۱	۱	۱	۰
P۳	۱	۱	۰	۱
P۴	۰	۰	۰	۰

Need

	R۰	R۱	R۲	R۳
P۰	۱	۱	۰	۰
P۱	۰	۱	۱	۲
P۲	۳	۱	۰	۰
P۳	۰	۰	۱	۰
P۴	۲	۱	۱	۰

Resource (کل منابع اولیه)

R۰	R۱	R۲	R۳
۶	۳	۴	۲

Available

R۰	R۱	R۲	R۳
۱	۰	۲	۰

$$Available = Resource - \sum_{i=0}^{4} Allocate_i$$

Available = [۶,۳,۴,۲] - [۵,۳,۲,۲]

Available = [۱,۰,۲,۰]

(a Request$_1$ = [۰,۰,۱,۰]

$\underset{[1,1,0,0]}{Reqest_1 \le Need_1} \longrightarrow$ Ok

Request$_1$ ≤ Availabe ⟶ ok ⟶ Available = [۱,۰,۲,۰] - [۰,۰,۱,۰] = [۱,۰,۱,۰]

Allocation$_1$ = [۰,۱,۰,۰] + [۰,۰,۱,۰] = [۰,۱,۱,۰] , Need$_1$ = Need$_1$ - Request$_1$

Need$_1$ = [۰,۱,۰,۲]

حال دنباله‌ی < P۳ , P۴ , P۰ , P۲ , P۱ > شرایط را ارضا می کند و قابل اجراست پس این حالت safe است و منابع تخصیص داده می شود.

Work = [۱,۰,۱,۰] , Finish = [F, F, F, F, F]

execute P۳

Finish[۳] = T = true ⟶ Finish[۴] = T , Work = [۲,۱,۱,۱] ⟶ Finish[۰] = True , work = [۵,۱,۲,۱] ⟶ Finish[۲] = true , work = [۶,۲,۳,۱] ⟶ Finish[۱] = T , Work = [۶,۳,۴,۲]

Work = [۲,۱,۱,۱]

(b Request$_4$ = [۰,۰,۱,۰]

Reqest$_1$ = [۰,۰,۱,۰]

Reqest$_4$ ≤ Need$_4$ و Reqest$_1$ ≤ Need$_4$ ⟶ OK

Reqest$_4$ ≤ Available , Reqest$_1$ ≤ Available ⟶ Available = [۱,۰,۲,۰] - [۰,۰,۱,۰] - [۰,۰,۱,۰] = [۱,۰,۰,۰]

Allocation$_4$ = [۰,۰,۱,۰] ~~Need$_4$~~

Allocation$_1$ = [۰,۱,۱,۰] ⟶ Need$_4$ = [۲,۱,۰,۰] Need$_1$ = [۰,۱,۰,۲]

work = Available = [۱,۰,۰,۰] , Finish = [F,F,F,F,F] هیچ کدام از فرایندها قابل اجرا نیستند ⟶ بن بست وجود دارد.

⟶ هیچ فرایندی وجود ندارد که Finish[i] = F و Need$_i$ ≤ Work ⟶ چون Finish = [false, false, false, false, false]

(C $\quad$ Request$_3$ = [۰,۰,۱,۰]

Requst$_4$ = (۲,۱,۱,۰]

Requst$_3$ ≤ Need$_3$ و Requst$_4$ ≤ Need$_4$ → Ok

~~Request$_3$ ≤ Ava.~~

Request$_3$ ≤ Available → Ok $\quad$ Available = [۱,۰,۲,۰] − [۰,۰,۱,۰] = [۱,۰,۱,۰]

Need$_3$ = [۰,۰,۰,۰], Allocate$_3$ = [۱,۱,۱,۱]

Request$_4$ $\not\leq$ Available → P$_۴$ must wait, since Resources aren't availble.

حال دنباله < P$_۳$, P$_۴$, P$_۰$, P$_۱$, P$_۲$ > شرایط را ارضای کنند و قابل اجراست پس این حالت safe است و منابع تخصیص داده می‌شود.

work = [۱,۰,۱,۰] $\quad$ finish = [F, F, F, F, F]

Finish[۳] = true
Work = [۲,۱,۲,۱] →

حال می‌توان به P$_۴$ منابع اختصاص داد چراکه
Request$_4$ ≤ work
و به درخواست آن پاسخ داد

Finish[4] = true
work = [۲,۱,۲,۱] →

Finish[۰] = T
work = [۵,۱,۳,۲] →

Finish[1] = T
work = [۵,۲,۳,۲]
↓
Finish(۲) = T
work = [۶,۳,۴,۲]

(d $\quad$ Reqest$_1$ = [۰,۰,۱,۰]

Reqest$_4$ = [۰,۰,۱,۰]

Reqest$_1$ ≤ Need$_1$, Reqest$_4$ ≤ need → Ok

Request$_1$ ≤ availabe, Reqest$_4$ ≤ Availabe → Ok

Available = [۰,۰,۲,۰] − (۰,۰,۱,۰] − [۰,۰,۱,۰] = [۰,۰,۰,۰]

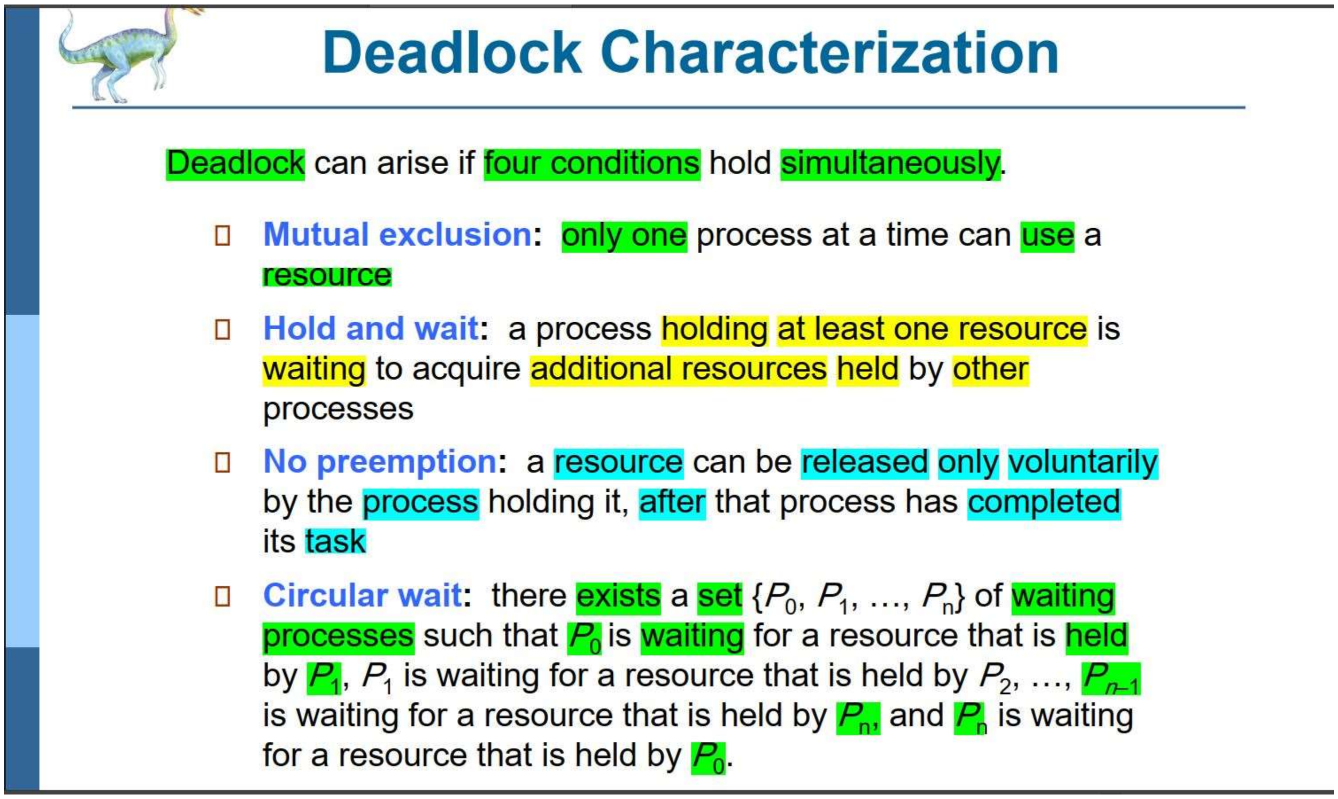
Allocation$_1$ = [۰,۱,۱,۰] $\quad$ Need$_1$ = [۰,۱,۰,۲]

Allocation$_4$ = [۰,۰,۱,۰] $\quad$ Need$_4$ = [۲,۱,۰,۰]

work = [۱,۰,۰,۰], Finish = [F,F,F,F,F] → هیچ فرایندی وجود ندارد که Finish[i] = F و Need$_i$ ≤ Work ✓

همچنین چون تمام فرایندها تمام نشده‌اند و آرایه Finish شامل مقدارهای False است پس deadlock یا بن بست وجود دارد.

.٢

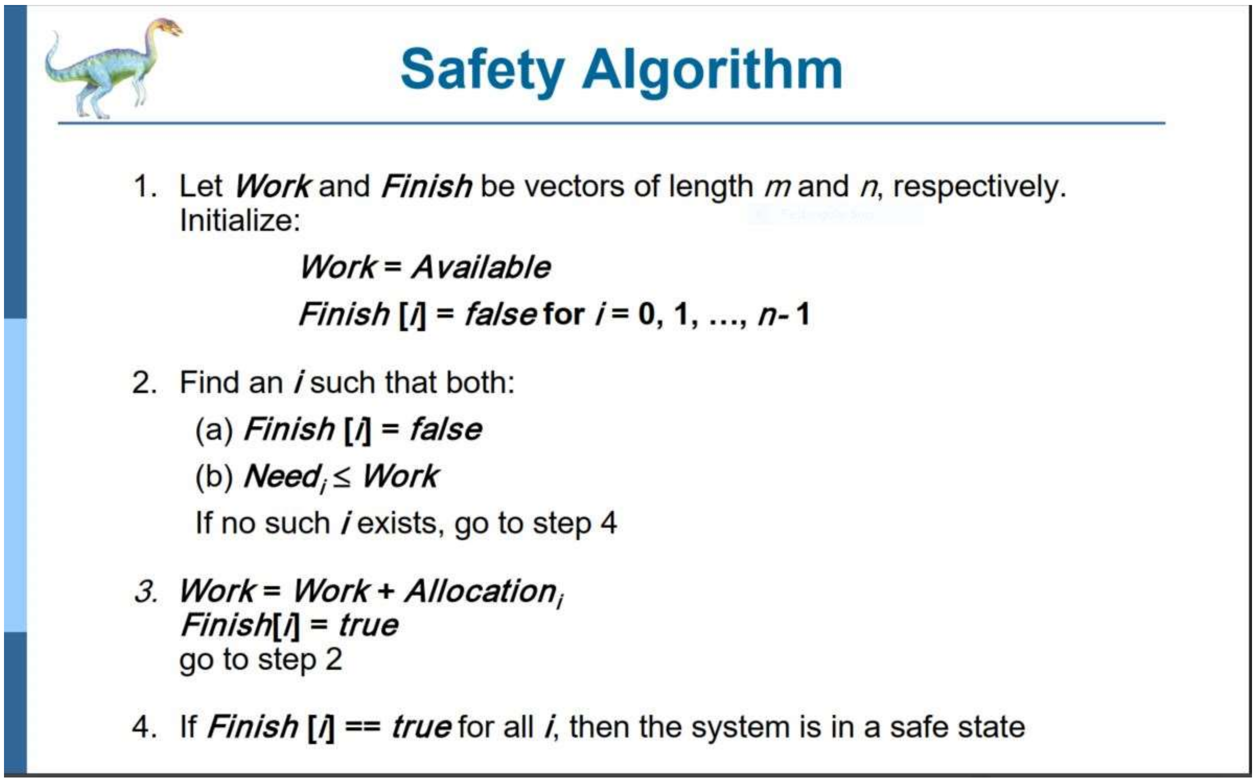


Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion:** only one process at a time can use a resource
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes
- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task
- **Circular wait:** there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0.

۳. از الگوریتم زیر استفاده می کنیم.



Safety Algorithm

1. Let ***Work*** and ***Finish*** be vectors of length *m* and *n*, respectively.
   Initialize:

   Work* = *Available

   ***Finish* [*i*] = *false* for *i* = 0, 1, …, *n*- 1**

2. Find an ***i*** such that both:

   (a) ***Finish* [*i*] = *false***

   (b) $\textbf{\textit{Need}}_i \leq \textbf{\textit{Work}}$

   If no such ***i*** exists, go to step 4

3. $\textbf{\textit{Work}} = \textbf{\textit{Work}} + \textbf{\textit{Allocation}}_i$
   Finish*[*i*] = *true
   go to step 2

4. If ***Finish* [*i*] == *true*** for all *i*, then the system is in a safe state

	Allocation				Max				Need = Max − Allocation			
	A	B	C	D	A	B	C	D	A	B	C	D
$T_{۰}$	۳	۰	۱	۴	۵	۱	۱	۷	۲	۱	۰	۳
$T_{۱}$	۲	۲	۱	۰	۳	۲	۱	۱	۱	۰	۰	۱
$T_{۲}$	۳	۱	۲	۱	۳	۳	۲	۱	۰	۲	۰	۰
$T_{۳}$	۰	۵	۱	۰	۴	۶	۱	۲	۴	۱	۰	۲
$T_{۴}$	۴	۲	۱	۲	۶	۳	۲	۵	۲	۱	۱	۳

a) Available = [۰, ۳, ۰, ۱]   init: Work = [۰, ۳, ۰, ۱]
Finish = [F, F, F, F, F]

قابل اجرا
$T_{۲} \rightsquigarrow Need_2 \leq Work$
Finish[۲] = T = true → $Need_1 \leq Work$ Finish[1] = T → $Need_3 \leq Work$ Finish[۳] = T → ~~Need~~ $Need_0 \nleq work$
Work = [۳, ۴, ۲, ۲]   Work = [۵, ۶, ۳, ۲]   work = [۵, ۱۱, ۴, ۲] $Need_4 \nleq work$

Finish[i] == T = true برای همه‌ی فرایندها نیست. ← ~~ه~~ بن بست یا deadlock وجود دارد. ← حالت unsafe یا ناامن است.

T_0 و T_4 درگیر deadlock هستند. ⟸ Finish = [F, T, T, T, F]

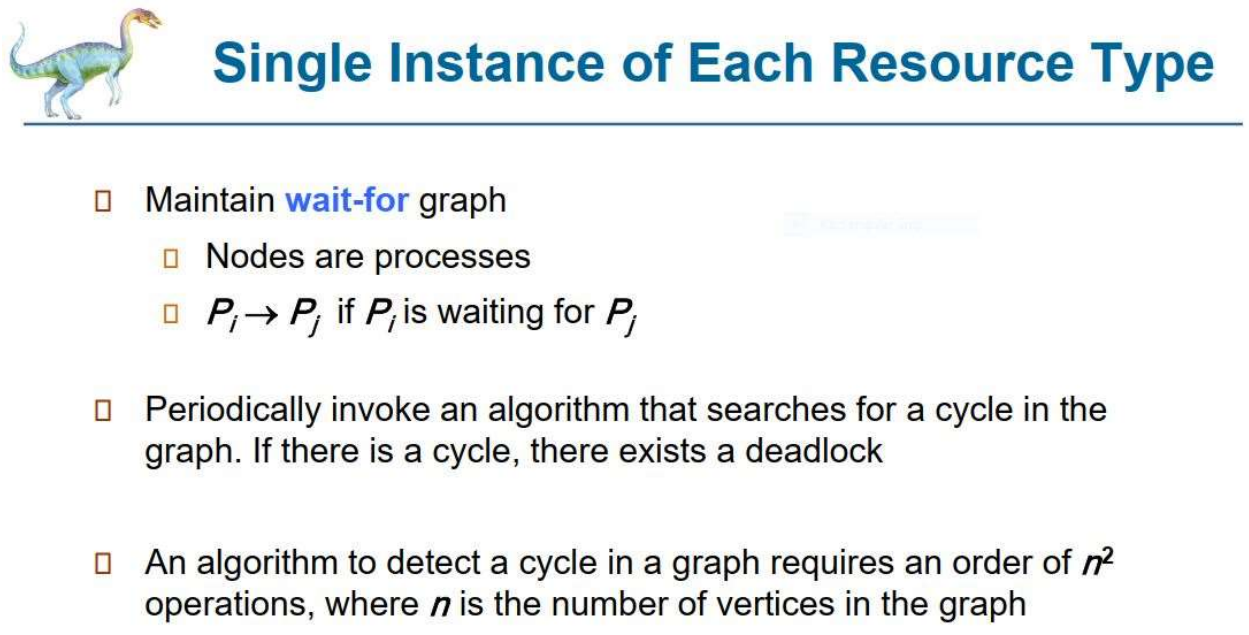
b) Available = [۱, ۰, ۰, ۲]   init: Work = [۱, ۰, ۰, ۲]
Finish = [F, F, F, F, F]

~~Need~~ $Need_1 \leq work$ Finish[1] = true work = [۳, ۲, ۱, ۲] → $Need_2 \leq work$ Finish[۲] = T work = [۶, ۳, ۳, ۳] → $Need_3 \leq work$ Finish[۳] = T work = [۶, ۸, ۴, ۳] → $Need_0 \leq work$ Finish[۰] = T Work = [۹, ۸, ۵, ۷]

→ $Need_4 \leq work$ work = [۱۳, ۱۰, ۶, ۹] → دیگر فرایندی نمانده و همگی اجرا شده‌اند. → Finish = [T, T, T, T, T]

این حالت safe یا امن است که چرا که دنباله‌ی $\langle T_1, T_2, T_3, T_0, T_4 \rangle$ شرایط را ارضا می‌کند و قابل اجرا است.

۴. از الگوریتم های زیر برای تشخیص بن بست استفاده می کنیم.



Single Instance of Each Resource Type

- Maintain **wait-for** graph
  - Nodes are processes
  - $P_i \rightarrow P_j$ if P_i is waiting for P_j
- Periodically invoke an algorithm that searches for a cycle in the graph. If there is a cycle, there exists a deadlock
- An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph

Detection Algorithm

1. Let *Work* and *Finish* be vectors of length *m* and *n*, respectively Initialize:

   (a) *Work* = *Available*

   (b) For $i = 1, 2, \ldots, n$, if $Allocation_i \neq \mathbf{0}$, then *Finish*[i] = *false*; otherwise, *Finish*[i] = *true*

2. Find an index *i* such that both:

   (a) *Finish*[*i*] == *false*

   (b) $Request_i \leq Work$

   If no such *i* exists, go to step 4

Detection Algorithm (Cont.)

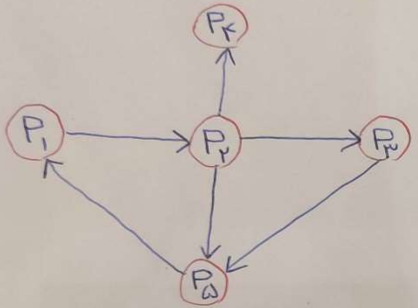
3. $Work = Work + Allocation_i$
   Finish[*i*] = *true*
   go to step 2

4. If *Finish[i] == false*, for some *i*, $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if *Finish*[*i*] == *false*, then P_i is deadlocked

(A

	allocation R۱	R۲	R۳	R۴	R۵	requested R۱	R۲	R۳	R۴	R۵	available R۱	R۲	R۳	R۴	R۵
P۱	۰	۰	۰	۱	۰	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰
P۲	۱	۰	۰	۰	۰	۰	۰	۱	۰	۱					
P۳	۰	۰	۱	۰	۰	۰	۰	۰	۰	۱					
P۴	۰	۱	۰	۰	۰	۰	۰	۰	۰	۰					
P۵	۰	۰	۰	۰	۱	۰	۰	۰	۱	۰					

باتوجه به اینکه منابع single instance هستند گراف wait-for رامی کشیم و می بینیم که دارای cycle هست یا خیر.



~~دو حلقه‌ی cycle در گراف وجود دارد.~~

P۱ → P۲ → P۳ → P۵ → P۱

P۱ → P۲ → P۵ → P۱

گراف دارای دو حلقه است.

بله بن بست (deadlock) وجود دارد و فرایند های داخل حلقه یعنی P۱, P۲, P۳, P۵ درگیر deadlock (بن بست) هستند.

(B

	allocation R۱	R۲	R۳	R۴	requested R۱	R۲	R۳	R۴	available R۱	R۲	R۳	R۴
P۱	۱	۰	۰	۰	۰	۱	۰	۰	۲	۰	۰	۰
P۲	۰	۱	۰	۰	۰	۰	۱	۰				
P۳	۰	۰	۱	۰	۰	۰	۰	۱				
P۴	۰	۱	۰	۱	۱	۰	۰	۰				
P۵	۰	۰	۰	۱	۰	۰	۰	۰				

init : Work = available = [۲, ۰, ۰, ۰]   Finish = [F, F, F, F, F]

Reqest4 ≤ Work, Finish[۴] = T, work = [۲,۱,۰,۱] → Reqest1 ≤ Work, Finish[۱] = T, work = [۳,۱,۰,۱] → Request3 ≤ Work, Finish[۳] = T, work = [۳,۱,۱,۱] → Reqest5 ≤ Work, Finish[۵] = T, work = [۳,۱,۱,۲] → Request2 ≤ Work, Finish[۲] = T, work = [۳,۲,۱,۲]

تمامی فرایندها قابل اجرا هستند و دنباله‌ی <P۴, P۱, P۳, P۵, P۲> شرایط را ارضای کند پس بن بست یا deadlock وجود ندارد و حالت ایمن است.