سیستم عامل

Operating Systems

فرجیان

فصل ششم

همزماني: بن بست و گرسنگي

مروري بر عناوين مطالب

- **مساله بن بست**
- مشخصه هاي بن بست
- روش هاي برخورد با بن بست
- پيشگيري از بن بست
- اجتناب از بن بست
- كشف بن بست
- ترميم از بن بست
- روش هاي تركيبي براي برخورد با بن بست

مروري بر اهداف فصل

- **آشنايي با مفهوم بن بست كه مانع از انجام كار يک يا چندين فرآيند مي شود**
- آشنايي با روش هاي مختلف مقابله با بن بست

مساله بن بست

- یک مجموعه از فرآیند ها در حالت بن بست هستند اگر هر یک از فرآیند ها منبعي را در اختیار داشته باشند و منتظر منبع دیگري باشند که در اختیار فرآیند اي دیگر در همین مجموعه است.

مثالی از بن بست در فرایند ها:

- ساده ترین نوع بن بست زمانی اتفاق می افتد که ۲ فرایند مجزا و همزمان منتظر بدست آوردن منبعی هستند که در اختیار دیگریست و خود نیز منابعشان را تا اتمام فرایند آزاد نمیکنند.

Process P :

```
Get A
Get B
Release A
Release B
.
.
```

Process Q :

```
Get B
Get A
Release B
Release A
.
.
```

انواع منابع:

- منابع نقش تعیین کننده ای در بن بست دارند.
- منابع به دو دسته تقسیم میشوند:
  - منابع قابل استفاده مجدد
  - منابع مصرف شدنی یا غیر قابل استفاده مجدد

منابع قابل استفاده مجدد:

- منابعی هستند که در هر لحظه از زمان تنها توسط یک فرایند قابل استفاده اند و استفاده از آنها موجب به پایان رسیدن آنها نمیشود(بدون آسیب دیدن آزاد میشوند)
- فرایند ها منابع را بدست می آورند و سپس آنها را برای استفاده مجدد توسط سایر فرایند ها آزاد میکنند.
- پردازنده، کانالهای I/O، حافظه اصلی و ثانوی، دستگاه ها و ساختمان داده هایی مثل پرونده ها، پایگاه های داده و ... از این دسته اند.
- بن بست زمانی رخ میدهد که یک فرایند منابع را نگه دارد و منبع دیگری درخواست کند.

مثالی از بن بست منابع قابل استفاده مجدد:

- فرض کنید فضای حافظه که میتواند به فرایند ها تخصیص یابد ۲۰۰ KB باشد.در این صورت ترتیب زیر موجب بروز بن بست میشود.

P1

. . .

Request 80 Kbytes;

. . .

Request 60 Kbytes;

P2

. . .

Request 70 Kbytes;

. . .

Request 80 Kbytes;

منابع مصرف شدنی:

- این منابع تولید میشوند و از بین میروند
- وقفه ها، علامتها، پیامها و اطلاعات بافر I/O از این نمونه اند.
- کشف بن بستهای حاصل از این منابع بسیار مشکل است و ممکن است ترکیب نادری از حوادث آنها را ایجاد کند.

مثالی از بن بست منابع قابل استفاده مجدد:

- در صورتیکه عمل Receive مسدود کننده باشد بن بست اتفاق می افتد.

P1

```
...
Receive(P2);
...
Send(P2, M1);
```

P2

```
...
Receive(P1);
...
Send(P1, M2);
```

سیستم عامل برای مدیریت هر نوع منبعی به 4 عمل نیاز دارد :

1. (Resource Status):در هر لحظه باید وضعیت منابع یعنی آزاد یا در حال استفاده بودن آنها توسط تکنیک های مشخص می باشد .

2. (Schedulingزمانبندی) : باید منابع به درستی توسط سیستم عامل زمانبندی می شوند یعنی سیستم عامل باید بداند در چه زمانی ، چه منبعی را به کدام پرداش بدهد .

3. Allocation: یعنی تخصیص واقعی منبع به پردازش در زمان معین

4. Release: یعنی باز پس گیری منبع از پردازش در زمان معین

مدل سیستم

- منابع سیستم از انواع مختلف $R_1, R_2, \ldots, R_m$ هستند.
  - قطعه هاي زماني پردازنده، فضاي حافظه، ابزارهاي خواندن / نوشتن، ...
- از هر منبع R_i ، تعداد W_i تا داريم.
- هر فرآيند از هر منبع با رعايت توالي اعمال زير بهره برداري مي كند:
  - درخواست
  - استفاده
  - رهاسازي

مروري بر عناوين مطالب

- مساله بن بست
- **مشخصه هاي بن بست**
- روش هاي برخورد با بن بست
- پيشگيري از بن بست
- اجتناب از بن بست
- كشف بن بست
- ترميم از بن بست
- روش هاي تركيبي براي برخورد با بن بست

مشخصه هاي بن بست

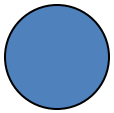
- براي رخ دادن بن بست برقراري همزمان چهار شرط زير الزامي است:
  - ممانعت دوجانبه (Mutual Exclusion): در هر زمان تنها يک فرآيند مي تواند از يک منبع استفاده کند.
  - نگهدار و منتظر بمان (Hold and Wait): فرآيند اي که حداقل يک منبع در اختيار دارد، منتظر است تا منابع ديگري که در اختيار فرآيند هاي ديگر است نيز بگيرد.
  - انحصاری بودن (No Preemption): هر منبعي تنها پس از پايان کار فرآيند اي که آن را در اختيار گرفته است و به صورت داوطلبانه توسط همان فرآيند قابل رهاسازي است.
  - انتظار حلقوي (Circular Wait): مجموعه اي مانند $\{P_0, P_1, \ldots, P_n\}$ وجود دارد به گونه اي که P_0 منتظر منبعي است که در اختيار P_1 قرار دارد، P_1 منتظر منبعي است که در اختيار P_2 قرار دارد، ... ، و P_n منتظر منبعي است که در اختيار P_0 قرار دارد.

گراف تخصیص منابع

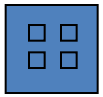
- گراف تخصیص منابع به صورت مجموعه رئوس V و مجموعه يال هاي جهت دار E تعريف مي شود.
- دو نوع راس: فرآيند ها و نوع منابع
  - مجموعه همه فرآيند هاي سيستم $P = \{P_1, P_2, ..., P_n\}$
  - مجموعه همه انواع منابع سيستم $R = \{R_1, R_2, ..., R_m\}$
- هر يال جهت دار نشانگر درخواست يا اختصاص منبع
  - درخواست منبع: $P_1 \rightarrow R_j$
  - اختصاص منبع: $R_j \rightarrow P_i$

گراف تخصیص منابع (ادامه)

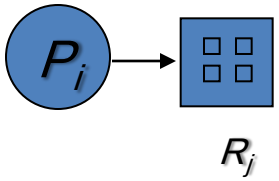
- فرآیند:



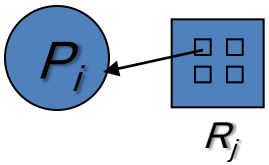
- نوع منبع با چهار نمونه:



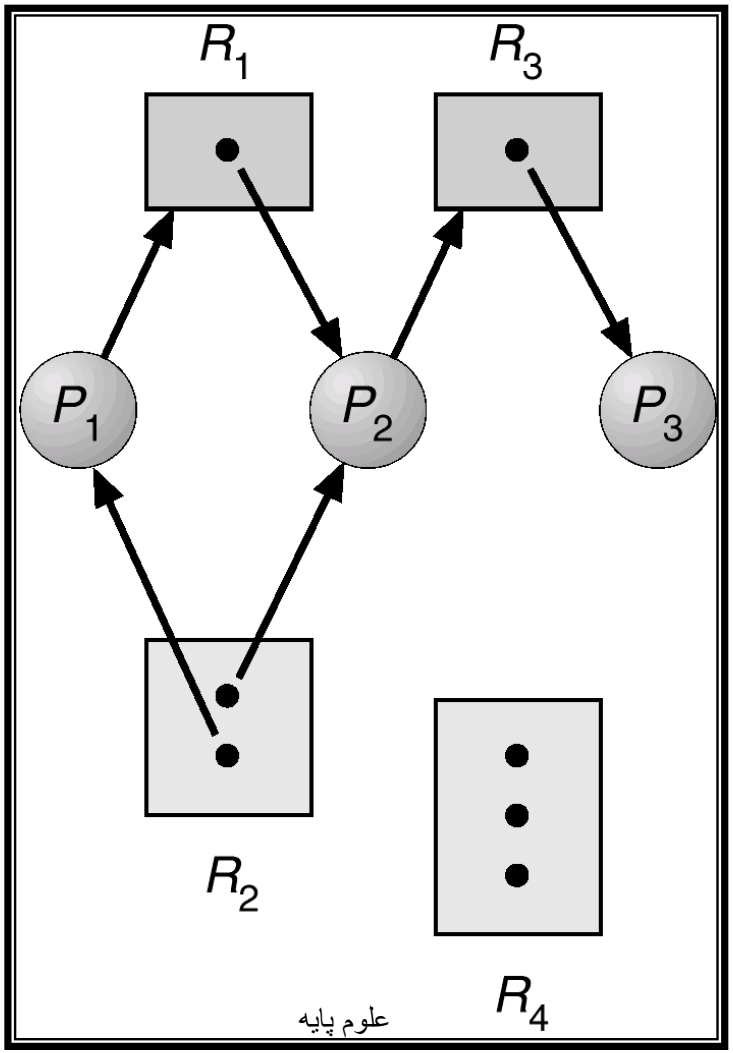
- P_i یک نمونه از R_j را تقاضا مي کند:



- P_i یک نمونه از R_j را در اختیار دارد:

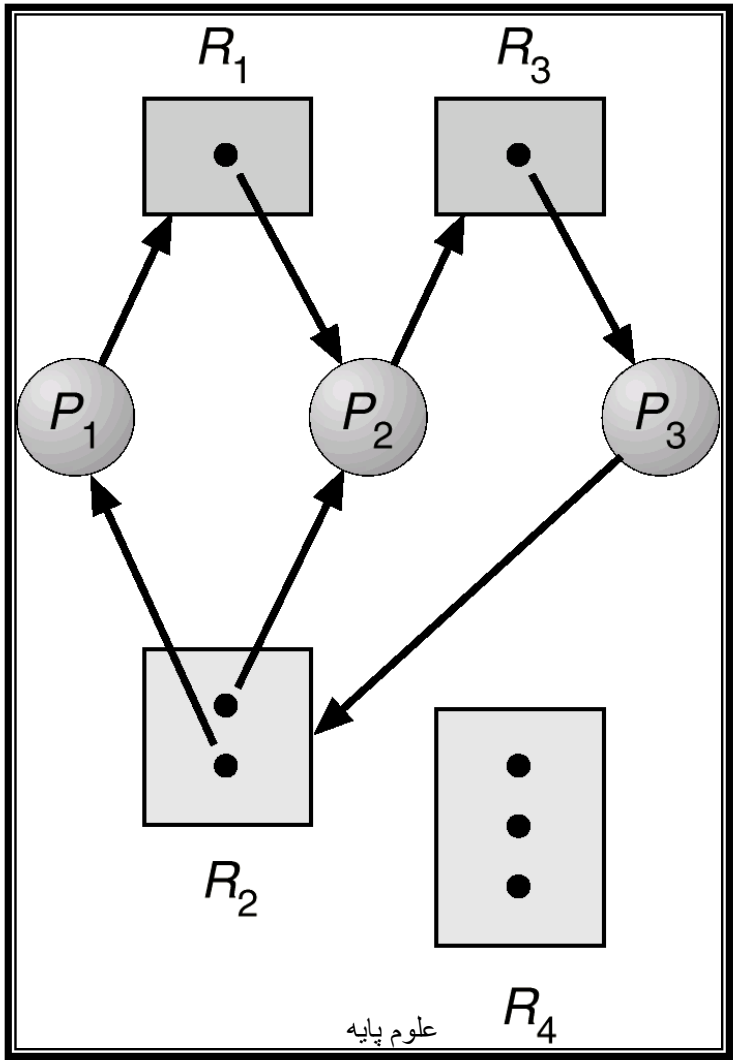


یک نمونه از گراف تخصیص منابع

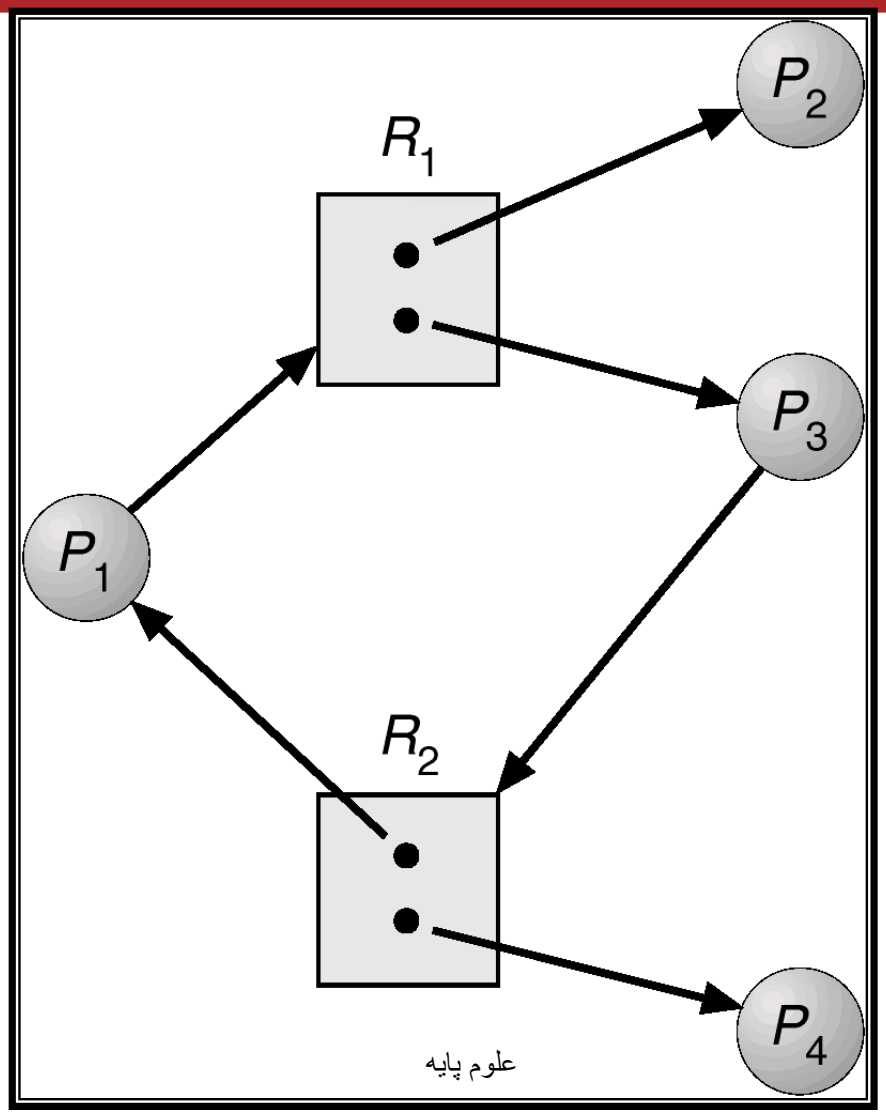


گراف تخصیص منابع

(با حلقه و بن بست)



گراف تخصیص منابع



(با حلقه و بدون بن بست)

معیار تشخیص اولیه

- اگر گراف تخصیص منابع حلقه ندارد ...
  - سیستم بن بست ندارد.
- اگر گراف تخصیص منابع حلقه دارد ...
  - اگر از هر نوع منبع درگیر در حلقه فقط یک نمونه داریم، یک بن بست رخ داده است.
  - اگر از هر نوع منبع درگیر در حلقه چندین نمونه داریم، ممکن است بن بست رخ داده باشد.

مروري بر عناوين مطالب

- مساله بن بست
- مشخصه هاي بن بست
- **روش هاي برخورد با بن بست**
- پيشگيري از بن بست
- اجتناب از بن بست
- كشف بن بست
- ترميم از بن بست
- روش هاي تركيبي براي برخورد با بن بست

روش هاي برخورد با بن بست

- در حالت كلي به چهار گونه مي توان با مساله بن بست برخورد كرد: پیشگیری از بن بست، اجتناب از بن بست، ترميم بن بست و صرف نظر كردن از بن بست
- براي تضمين اينكه هيچ گاه بن بست در سيستم رخ نمي دهد...
  - بايد مكانيزمي وجود داشته باشد كه قبل از تخصيص منابع به فرآيند ها صحت شروط تضمين عدم امكان وقوع بن بست را ارزيابي كند.
  - پيشگيري از بن بست و اجتناب از بن بست هر دو در اين دسته جاي مي گيرند.

روش هاي برخورد با بن بست (ادامه)

- ترميم بن بست
  - در اين رهيافت به سيستم اجازه داده مي شود وارد حالت بن بست شود و سپس تلاش مي شود بن بست رفع شود.
  - نياز به مكانيزم هايي است كه وقوع بن بست را تشخيص دهند و سپس آن را برطرف كنند.
- صرف نظر كردن از بن بست
  - در بسياري از سيستم ها بن بست به ندرت رخ مي دهد، بنابراين به جاي استفاده از راه حل هاي گران قبلي به سادگي فرض مي شود هيچ گاه بن بست رخ نمي دهد.
  - راه حل مورد استفاده در UNIX

مروري بر عناوين مطالب

- مساله بن بست
- مشخصه هاي بن بست
- روش هاي برخورد با بن بست
- **پيشگيري از بن بست**
- اجتناب از بن بست
- كشف بن بست
- ترميم از بن بست
- روش هاي تركيبي براي برخورد با بن بست

پيشگيري از بن بست

- براي پيشگيري از بن بست، بايد يكي از شروط چهارگانه را نقض كرد.

1. ممانعت دوجانبه(انحصار متقابل)
   - اگر منبع قابل اشتراک گذاري باشد (مانند يک پرونده فقط خواندني) مي توان در مورد آن از شرط ممانعت دوجانبه صرف نظر كرد. (مانند فایل های فقط خواندنی)
   - در حالت كلي اين شرط قابل نقض نيست، چون ماهيت بسياري از منابع (مانند چاپگر) وجود اين شرط را الزامي مي كند.

پيشگيري از بن بست (ادامه)

2. گرفتن و منتظر ماندن

– براي نقض اين شرط هر فرآيند تنها بايد در صورتي تقاضاي يک منبع را بکند که هيچ منبع ديگري در اختيار نداشته باشد.

– فرآيند دو راه براي درخواست منبع دارد.

- يا بايد همه منابع مورد نياز خود را در ابتداي اجراي خود به صورت يکجا درخواست کند
- يا بايد همه منابع در اختيار را آزاد کرده و سپس تقاضاي منبع جديد کند.

– بهره وري منابع در اين روش پايين است

– احتمال قحطي نيز وجود دارد.

پيشگيري از بن بست (ادامه)

3.انحصاری بودن

- براي نقض اين شرط مي توان به سيستم عامل اجازه داد تا در صورت لزوم پيشدستي كرده و خود منابع در اختيار فرآيند را آزاد كند.
- در صورتي كه فرآيندي تقاضاي منبع كرد و اختصاص منبع امكان پذير نبود، سيستم عامل مي تواند تمام منابع در اختيار فرآيند متقاضي را آزاد كند.
- منابع آزادشده به ليست منابع مورد تقاضاي فرآيند افزوده مي شوند.
- اجراي فرآيند وقتي از سر گرفته مي شود كه فرآيند بتواند منبع مورد نياز و تمام منابع آزاد شده قبلي را در اختيار بگيرد.
- برای منابعی مثل حافظه میشود اما برای چاپگر ؟

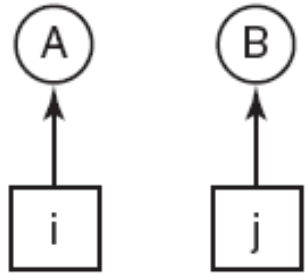
پيشگيري از بن بست (ادامه)

4.انتظار حلقوي

- براي نقض اين شرط مي توان يک ترتيب کلي روي همه منابع موجود در سيستم اعمال کرد و سپس از هر فرآيند خواست منابع مورد نياز خود را فقط در يک ترتيب افزايشي درخواست کند.

1. Imagesetter
2. Scanner
3. Plotter
4. Tape drive
5. CD-ROM drive

(a)



(b)

Approaches to Deadlock Prevention

Condition	Approach
Mutual exclusion	Spool everything
Hold and wait	Request all resources initially
No preemption	Take resources away
Circular wait	Order resources numerically

- Figure 6-14. Summary of approaches to deadlock prevention.

مروري بر عناوين مطالب

- مساله بن بست
- مشخصه هاي بن بست
- روش هاي برخورد با بن بست
- پيشگيري از بن بست
- **اجتناب از بن بست**
- كشف بن بست
- ترميم از بن بست
- روش هاي تركيبي براي برخورد با بن بست

اجتناب از بن بست

- در اين رهيافت سيستم تلاش مي كند تا با استفاده از يک *دانش قبلي* درباره رفتار فرآيند ها مانع از وقوع بن بست شود.
- رهيافت «پيشگيري از بن بست» بر نقض شرايط پيش نيازي وقوع بن بست و رهيافت «اجتناب از بن بست» بر نظارت مداوم و بررسي امكان يا عدم امكان وقوع بن بست تمركز دارند.
- در ساده ترين و مفيدترين روش موجود هر فرآيند قبل از شروع اجرا بايد نوع و حداكثر تعداد منابع مورد نياز از هر نوع را اعلان كند.

حالت تخصيص منابع و حالت امن

- الگوريتم اجتناب از بن بست به صورت پويا *حالت تخصيص منابع* را بررسي مي كند تا تضمين كند هيچ گاه انتظار حلقوي اتفاق نمي افتد.
- *حالت تخصيص منابع* از منابع موجود و تخصيص داده شده و همچنين حداكثر تقاضاي فرآيند ها براي هر نوع منبع تشكيل شده است.
- وقتي فرآيند اي تقاضاي اختصاص يک منبع موجود را مي كند، سيستم بايد تصميم بگيرد كه آيا اختصاص فوري منبع به فرآيند سيستم را در *حالت امن* نگاه مي دارد يا خير؟

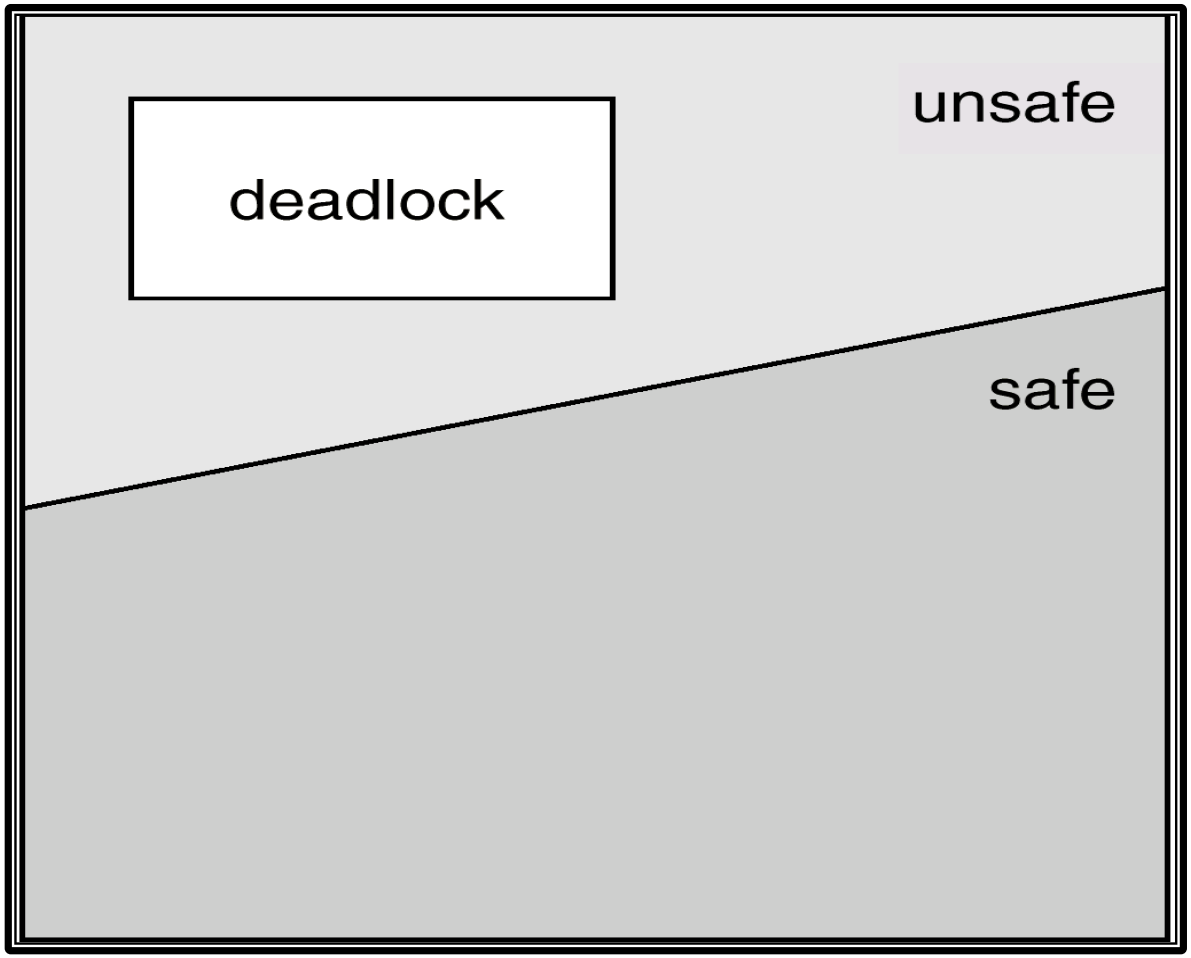
حالت امن

- سیستم در *حالت امن* است در صورتي كه يک *ترتيب امن* از فرآيند ها وجود داشته باشد.
- ترتيب $<P_1, P_2, \ldots, P_n>$ يک ترتيب امن است در صورتي كه براي هر P_i، منابعي كه P_i مي تواند تقاضا كند زيرمجموعه اي از منابع موجود و منابع در اختيار P_j ها باشد $(\forall j < i)$.
  - اگر منابع مورد نياز P_i همه فراهم نيستند، P_i مي تواند تا پايان تمام P_j ها كه منابع مورد نياز را در اختيار دارند صبر كند.
  - وقتي منابع آماده شد، P_i مي تواند آنها را در اختيار بگيرد، اجرا شود و سپس منابع در اختيار را آزاد كند.
  - وقتي P_i پايان يافت، P_{i+1} مي تواند اجرا شود و به همين صورت اجرا ادامه مي يابد.

معیار تشخیص اولیه

- اگر سیستم در یک حالت امن است ...
  - سیستم بن بست ندارد.
- اگر سیستم در یک حالت ناامن است ...
  - ممکن است بن بست رخ داده باشد.
- اجتناب از بن بست تضمین مي کند سیستم هیچ گاه وارد یک حالت ناامن نشود.

امنیت – عدم امنیت و بن بست



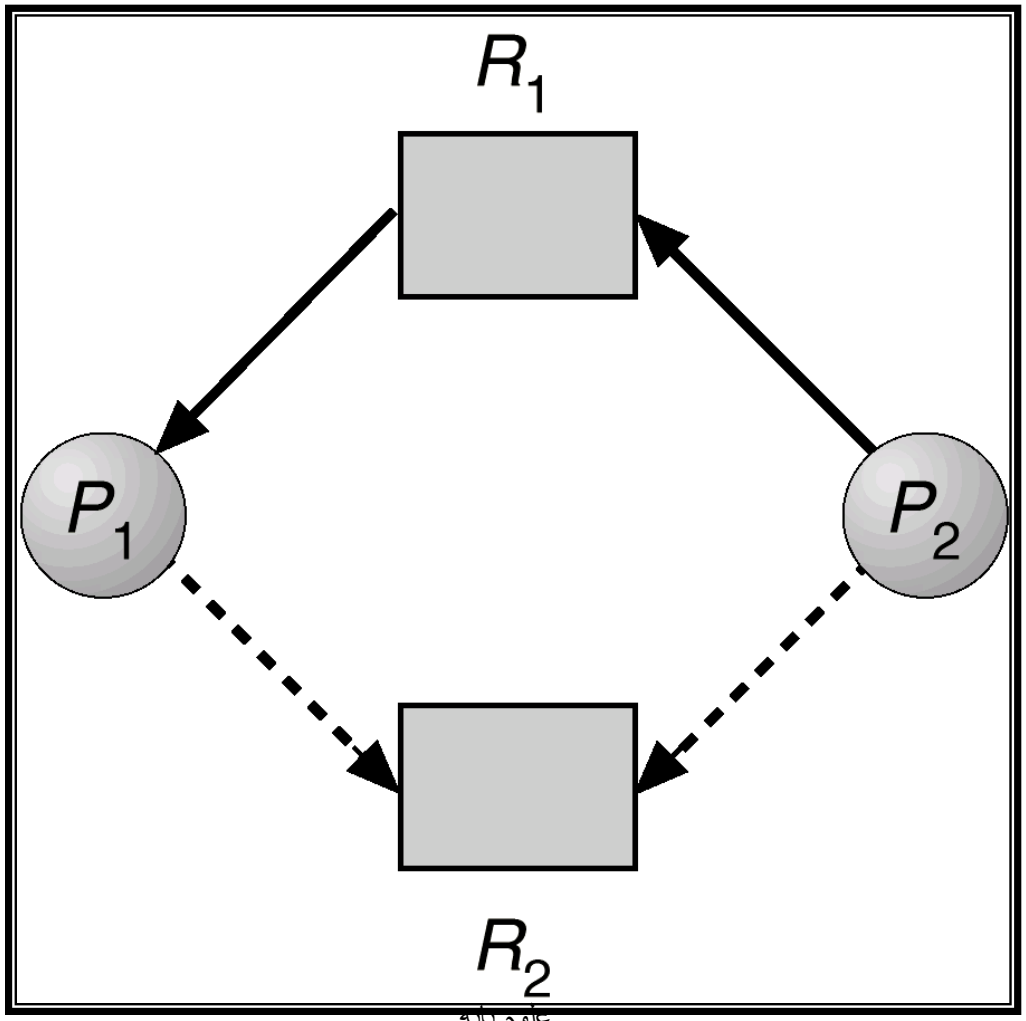
الگوريتم هاي جلوگيري از بن بست

- هنگامي كه از هر منبع تنها يک نمونه موجود است . از الگوريتم گراف تخصيص منابع استفاده مي كنيم .
- هنگامي كه بيش از يک نمونه ازمنبع موجود است . از الگوريتم بانكدار استفاده مي كنيم .

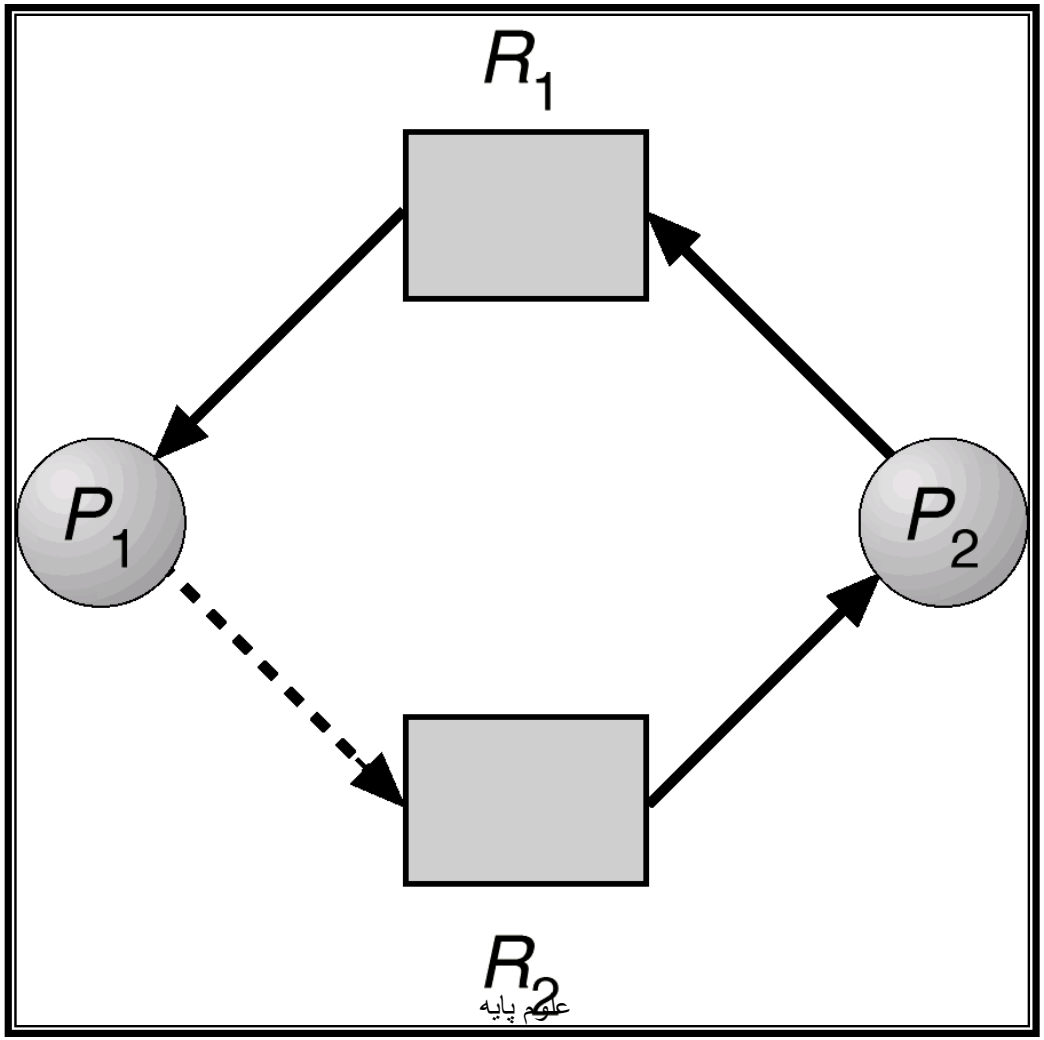
الگوريتم گراف تخصيص منابع

- اجزاي الگوريتم:
  - وقتي ممكن است فرآيند P_i منبع R_j را تقاضا كند، يک *يال خواسته* *(claim edge)* به صورت خط چين بين رئوس مربوطه مي كشيم.
  - وقتي فرآيند منبع را درخواست مي كند، يال خواسته به يال تقاضاي منبع تبديل مي شود.
  - وقتي فرآيند منبع را آزاد مي كند، يال تخصيص به يال خواسته تبديل مي شود.
  - منابع مورد نياز بايد از قبل از سيستم *خواسته* *شوند*.

گراف تخصيص منابع براي اجتناب از بن بست



حالت ناامن در گراف تخصیص منابع



الگوریتم گراف تخصیص منابع

- فرض کنیم فرآیند P_i منبع R_j را تقاضا کند
- درخواست تنها هنگامي پاسخ داده مي شود که تبدیل یال تقاضا به یال تخصیص باعث ایجاد یک حلقه در گراف نشود

الگوريتم بانكدارها

- اين الگوريتم از حالتي كه از هر نوع منبع چندين نمونه داشته باشيم پشتيباني مي كند.
  – الگوريتم گراف تخصيص منابع براي اين حالت جواب نمي دهد.
- هر فرآيند بايد از پيش حداكثر تعداد منابع مورد نياز خود را اعلام كند.
- وقتي فرآيندي تقاضاي منبعي را مي كند، ممكن است مجبور شود منتظر بماند.
- وقتي فرآيند اي تمام منابع مورد نياز را مي گيرد، بايد همه را در زمان محدود بازگرداند.

ساختار داده الگوریتم بانکدار ها

- فرض کنید n تعداد فرآیند ها و m تعداد نوع منابع باشد(ارایه)

int available [m];

available [j] = k یعني k نمونه از منبع R_j موجود هستند.

int max [n][m];

max [i][j] = k یعني فرآیند P_i حداکثر ممکن است k نمونه از منبع R_j را درخواست کند.

int allocation [n][m];

allocation [i][j] = k یعني فرآیند P_i، k نمونه از منبع R_j را در اختیار دارد.

int need [n][m];

need [i][j] = k یعني فرآیند P_i به k نمونه دیگر از منبع R_j نیاز دارد تا کارش را تمام کند.

Need [i,j] = Max[i,j] – Allocation [i,j]

الگوريتم ايمني – آيا سيستم در حالت امن است؟

1. فرض كن *Work* و *Finish* دو بردار به طول m و n باشند. اين بردارها را به اين صورت مقداردهي اوليه كن:

Finish = {False}; Work = Available;

2. انديس i را به گونه اي پيدا كن كه:

Finish [i] = False; and $Need_i \leq Work$;

اگر چنين i پيدا نكردي به گام 4 برو.

3. **$Work = Work + Allocation_i$; Finish [i] = True;**

به گام 2 برو.

4. اگر $\forall i$: *Finish [i] == True* آنگاه سيستم در يک حالت امن است.

الگوريتم درخواست منبع براي فرآيند P_i

- بردار $Request_i$ را به عنوان بردار نياز فرآيند P_i تعريف مي كنيم.
  - $Request_i [j] == k$ يعني فرآيند P_i به k نمونه از منبع R_j نياز دارد.

1. اگر $Request_i \leq Need_i$ به گام 2 برو، وگرنه اعلام خطا.
2. اگر $Request_i \leq Available$ به گام 3 برو، وگرنه P_i بايد منتظر بماند تا منابع مورد نياز آزاد شوند.
3. فرض كن منابع مورد نياز P_i را اختصاص داده اي. حالت تخصيص منابع را به صورت زير به روز كن:

$Available = Available – Request_i$

$Allocation_i = Allocation_i + Request_i;$

$Need_i = Need_i – Request_i;$

اگر حالت سيستم امن بود $\Leftarrow$ منابع به P_i اختصاص يافته اند.

اگر حالت سيستم ناامن بود $\Leftarrow$ P_i بايد منتظر بماند، حالت قبلي سيستم را بازيابي كن.

الگوریتم بانکداران:

- با توجه به جداول زیر بگویید آیا حالت امن است؟
  - این جدول بیان میکند تنها یک نوع منبع داریم.
  - فرایند A به ۶ منبع نیاز دارد و در حال حاضر ۱ منبع به آن اختصاص یافته است.(برای فرایندB نیز به همین صورت)
  - در حال حاظر از کل تعداد منابع تنها ۲ منبع آزاد است.(Available=2)

Availavle=2		
فرایند	Allocation	Max
A	1	6
B	1	5
M	2	4
S	4	7

Press Enter

الگوریتم بانکداران:

- محاسبه Need :

(Need=Max-Allocation)

Availavle=2			
فرایند	Allocation	Max	Need
A	1	6	5 1
B	1	5	4 1
M	2	4	2 2
S	4	7	3 4

Press Enter

الگوریتم بانکداران:

پیدا کردن کوچکترین مقدارNeed:

Availavle= 4			
فرایند	Allocation	Max	Need
A	1	6	5
B	1	5	4
M	2	4	2
S	4	7	3

2<2(Available)

Available=Available + 2

Press Enter

الگوریتم بانکداران:

پیدا کردن کوچکترین مقدارNeed:

Availavle= 8			
فرایند	Allocation	Max	Need
A	1	6	5
B	1	5	4
M	2	4	2
S	4	7	3

3<4(Available)

Available=Available + 4

Press Enter

الگوریتم بانکداران:

پیدا کردن کوچکترین مقدارNeed:

Availavle= 9			
فرایند	Allocation	Max	Need
A	1	6	5
B	1	5	4
M	2	4	2
S	4	7	3

4<8(Available)

Available=Available + 1

Press Enter

الگوریتم بانکداران

پیدا کردن کوچکترین مقدارNeed:

Availavle= 10			
فرایند	Allocation	Max	Need
A	1	6	5
B	1	5	4
M	2	4	2
S	4	7	3

5<9(Available)

Available=Available + 1

Press Enter

الگوریتم بانکداران

- چون توانستیم تمام فرایند ها را حذف کنیم بنابراین حالت داده شده در صورت مسأله امن است.
- توجه داشته باشید که در این مسأله فرض شد که تنها یک نوع منبع وجود دارد . در مثال بعدی مسأله را با انواع مختلف منابع بررسی میکنیم.

Press Enter

الگوریتم بانکداران:

- در این حالت نیز همانند قبل عمل میکنیم. با این تفاوت که کمترین تعداد نیاز یک سطر است.
- بنابراین ابتدا مقدار نیاز را محاسبه میکنیم.
- تعداد نیاز های هر فرایند را جمع میزنیم و کمترین را انتخاب میکنیم.
- اگر سطر مربوط به کمترین مقدار از مقادیر منبع باقیمانده بیشتر بود، نتیجه میگیریم که حالت ناامن است. در غیر این صورت
- فرایند با کمترین نیاز را حذف میکنیم و تعداد منابع تخصیص یافته آنرا را آزاد میکنیم و مراحل را دوباره تکرار میکنیم.

الگوریتم بانکداران:

- مثال: آیا حالت زیر یک حالت امن است ؟

Press Enter

Allocation				
فرایند	R	S	T	U
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

MAX				
فرایند	R	S	T	U
A	4	1	1	1
B	0	2	1	2
C	4	2	1	0
D	1	1	1	1
E	2	1	1	0

Resource	
نوع	تعداد
R	6
S	3
T	4
U	2

الگوریتم بانکداران:

- مرحله اول :جدول Need را محاسبه میکنیم.

Press Enter

Allocation					Max				Need=max-Alloc				
فرایند	R	S	T	U	R	S	T	U	R	S	T	U	+
A	3	0	1	1	4	1	1	1	1	1	0	0	2
B	0	1	0	0	0	2	1	2	0	1	1	2	4
C	1	1	1	0	4	2	1	0	3	1	0	0	4
D	1	1	0	1	1	1	1	1	0	0	1	0	1
E	0	0	0	0	2	1	1	0	2	1	1	0	4

الگوریتم بانکداران:

- جدول Available را محاسبه میکنیم.

Press Enter

Allocation				
فرایند	R	S	T	U
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0
+	5	3	2	2

Resource	
نوع	تعداد
R	6
S	3
T	4
U	2

Available= Res[i] - A[+][i]	
نوع	تعداد
R	6-5=1
S	3-3=0
T	4-2=2
U	2-2=0

الگوریتم بانکداران:

- Press Enter

Allocation					Need				
فرایند	R	S	T	U	R	S	T	U	+
A	3	0	1	1	1	1	0	0	2
B	0	1	0	0	0	1	1	2	4
C	1	1	1	0	3	1	0	0	4
D	1	1	0	1	0	0	1	0	1
E	0	0	0	0	2	1	1	0	4

Available	
نوع	تعداد
R	1
S	0
T	2
U	0

مثالي از الگوريتم بانكدار ها

- پنج فرآيند P_0 تا P_4 و سه منبع A (10 نمونه)، B (5 نمونه) و C (7 نمونه).
- حالت تخصيص منابع در T_0:

	Allocation	*Max*	*Available*
	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	

مثالي از الگوريتم بانكدارها (ادامه)

- مقادير ماتريس Need را محاسبه مي كنيم:

Need

A B C

7 4 3 P_0

1 2 2 P_1

6 0 0 P_2

0 1 1 P_3

4 3 1 P_4

- سيستم در يک حالت امن است چون ترتيب $< P_1, P_3, P_4, P_2, P_0>$ شرط امنيت را تامين مي كند و يک ترتيب امن است.

مثال : P_1 براي (1,0,2) درخواست مي دهد

- ببين آيا Request ≤ Available است؟ يعني :
  - (1,0,2) ≤ (3,3,2) ⇒ true

Available	Need	Allocation	
A B C	A B C	A B C	
2 3 0	7 4 3	0 1 0	P_0
	0 2 0	3 0 2	P_1
	6 0 0	3 0 1	P_2
	0 1 1	2 1 1	P_3
	4 3 1	0 0 2	P_4

- اجراي الگوريتم ايمني نشان مي دهد رشته ي $<P_1, P_3, P_4, P_0, P_2>$ شرايط امنيت را برقرار مي كند.
- آيا مي توان درخواست (3,3,0) را براي P_4 اجابت كرد؟
- آيا مي توان درخواست (0,2,0) را براي P_0 اجابت كرد؟

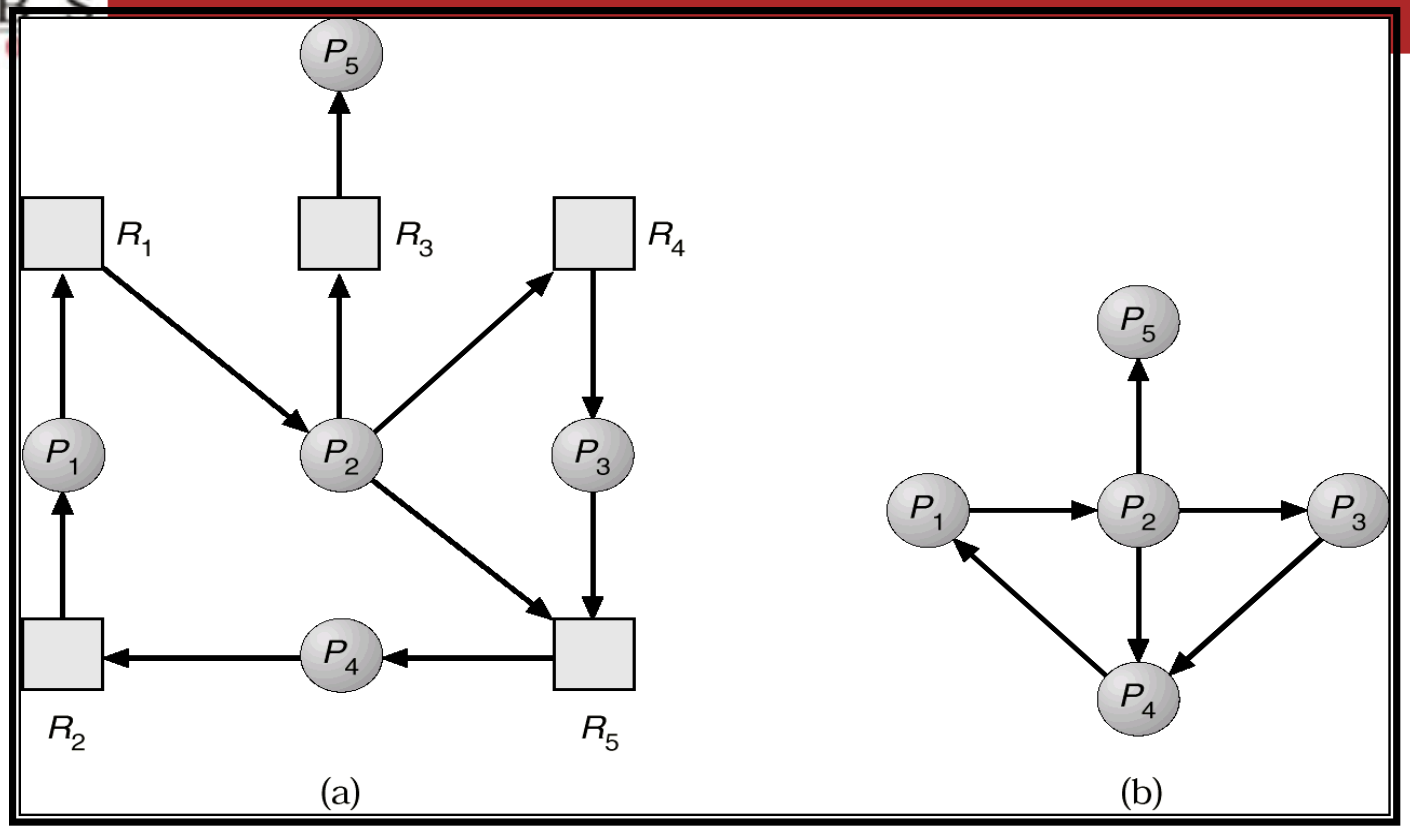
مروري بر عناوين مطالب

- مساله بن بست
- مشخصه هاي بن بست
- روش هاي برخورد با بن بست
- پيشگيري از بن بست
- اجتناب از بن بست
- **كشف بن بست**
- ترميم از بن بست
- روش هاي تركيبي براي برخورد با بن بست

کشف بن بست

- به سیستم اجازه داده مي شود وارد حالت بن بست شود و سپس تلاش مي شود بن بست کشف شده و ترميم شود.
  - مکانيزم کشف بن بست ؟
  - مکانيزم ترميم بن بست ؟
- در حالتي که از هر نوع منبع يکي موجود باشد، مي توان از *گراف انتظار* *(wait-for graph)* استفاده کرد.
  - هر فرآيند يک گره.
  - $P_i \rightarrow P_j$ يعني P_i منتظر P_j است.
  - به صورت دوره اي گراف انتظار براي يافتن حلقه جستحو مي شود. پيچيدگي زماني اين الگوريتم $O(n^2)$ است (n تعداد فرآيند ها).

گراف تخصیص منابع و گراف انتظار



گراف تخصیص منابع

گراف انتظار

الگوریتم کشف بن بست در حالت وجود چند نمونه از هر منبع

- ساختار داده الگوریتم:
  - m تعداد نوع منابع و n تعداد فرآیند ها

```
int available [m];
int allocation [n][m];
int Request [n][m];
```

- الگوریتم:

1. فرض کن *Work* و *Finish* دو بردار به طول m و n باشند. این بردارها را به این صورت مقداردهي اوليه کن:

```
Work = Available;
if (Allocation_i ≠ 0) Finish_i = False;
else Finish_i = True;
```

الگوريتم کشف بن بست در حالت وجود چند نمونه از هر منبع (ادامه)

- ادامه الگوريتم

2. انديس i را به گونه اي پيدا کن که:

$Request_i \leq Work$; **Finish [i] = False;**

اگر چنين i پيدا نکردي به گام 4 برو.

3. **$Work = Work + Allocation_i$;** **Finish [i] = True;**

به گام 2 برو.

4. اگر $\exists$ i: Finish [i] == False آنگاه سيستم در حالت بن بست است و P_i نيز در بن بست قرار دارد.

- پيچيدگي زماني اين الگوريتم $O(m \times n^2)$ است.

نمونه اي از كاربرد الگوريتم كشف بن بست

- پنج فرآيند P_0 تا P_4 و سه منبع A (7 نمونه)، B (2 نمونه) و C (6 نمونه). حالت تخصيص منابع در T_0:

	Allocation	*Request*	*Available*
	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	

n با در نظر گرفتن ترتيب $<P_0, P_2, P_3, P_1, P_4>$ داريم:

$\forall$ i: Finish [i] == True

نمونه اي از كاربرد الگوريتم كشف بن بست (ادامه)

- P_2 يک نمونه ديگر از c درخواست مي كند.

Request	
A B C	
0 0 0	P_0
2 0 1	P_1
0 0 1	P_2
1 0 0	P_3
0 0 2	P_4

- حالت سيستم:
  - سيستم تنها مي تواند پاسخگوي باقيمانده منابع مورد نياز P_0 باشد.
  - بن بست وجود دارد. فرآيند هاي P_1 تا P_4 در بن بست قرار دارند.

استفاده از کشف بن بست

- فواصل زماني و چگونگي فراخواني الگوريتم به موارد زير بستگي دارد:
  - معمولا هر چند مدت یکبار در سیستم بن بست رخ مي دهد؟
  - چه تعداد فرآیند باید عقب کشیده شوند؟
    - در هر حلقه بن بست حداقل يكي از فرآیند ها باید عقب کشیده شود.
- اگر الگوريتم در زمان دلخواه و بدون در نظر گرفتن معيارهاي بالا فراخواني شود، ممكن است ما با تعداد زيادي حلقه در گراف منابع و فرآیند گرفتارشده در بن بست مواجه شويم و نتوانيم تشخيص دهيم كدام فرآیند باعث ايجاد زنجير بن بست شده است.

مروري بر عناوين مطالب

- مساله بن بست
- مشخصه هاي بن بست
- روش هاي برخورد با بن بست
- پيشگيري از بن بست
- اجتناب از بن بست
- كشف بن بست
- **ترميم از بن بست**
- روش هاي تركيبي براي برخورد با بن بست

ترميم از بن بست

- مي توان بن بست را با پايان دهي به تعدادي فرآيند و رهاسازي منابع در اختيار آنها ترميم كرد.
  - پايان دهي به همه فرآيند هاي درگير
  - پايان دهي به يكي از فرآيند هاي درگير
- در انتخاب فرآيند قرباني بايد هزينه پرداختي پايان دهي به فرآيند را مينيمم كرد.
- پس از پايان دهي به فرآيند قرباني، سيستم به يک حالت امن عقبگرد مي كند

ترميم از بن بست (ادامه)

- براي انتخاب فرآيند قرباني مي توان معيارهايي در نظر گرفت...
  - اولويت فرآيند ها
  - زمان پردازنده مصرف شده، و تخمين موجود از زمان پردازنده مورد نياز براي تكميل كار فرآيند
  - منابع مصرف شده و منابع مورد نياز فرآيند براي تكميل كار
  - تعداد فرآيند هايي كه براي رفع همه بن بست ها بايد به آنها پايان داده شود.
  - تعاملي يا دسته اي بودن فرآيند.
  - كمترين خروجی توليد شده
- خطر قحطي: يک فرآيند هميشه قرباني شود
  - مي توان تعداد عقبگرد هاي هر فرآيند را نيز يک معيار در نظر گرفت.

مروري بر عناوين مطالب

- مساله بن بست
- مشخصه هاي بن بست
- روش هاي برخورد با بن بست
- پيشگيري از بن بست
- اجتناب از بن بست
- كشف بن بست
- ترميم از بن بست
- **روش هاي تركيبي براي برخورد با بن بست**

روش هاي تركيبي براي برخورد با بن بست

- مي توان پيشگيري، اجتناب و كشف بن بست را با هم تركيب كرد و براي هر منبع از روشي كه براي آن منبع مناسب تر است استفاده كرد.
- براي اين كار مي توان منابع را كلاس بندي كرد و سپس براي برخورد با بن بست در هر يک از كلاس هاي منابع از مناسبترين روش موجود استفاده كرد.