فصل پنجم

بن‌بست

۵ ـ ۱ ـ تعریف بن‌بست (DeadLock)

در یک محیط چند برنامگی، این امکان وجود دارد که فرآیندهای متفاوتی برای به‌دست آوردن تعداد محدودی منبع با یکدیگر رقابت کنند. یک فرآیند، منابعی را درخواست می‌کند و درصورتی که این منابع در آن لحظه در دسترس نباشند، فرآیند به وضعیت انتظار وارد می‌گردد. ممکن است حالتی رخ دهد که فرآیندهای در حال انتظار هرگز از وضعیت انتظار خارج نگردند، به این دلیل که منابع مورد درخواست آن‌ها، در اختیار فرآیندهای منتظر دیگر قرار دارند، این وضعیت را **بن‌بست** گویند.

یک سیستم از تعداد محدودی منابع تشکیل شده است که این منابع بین تعدادی فرآیند رقیب، توزیع می‌شوند. منابع به انواع مختلفی قابل تقسیم هستند که هر یک از این انواع خود نمونه‌های چندی دارند مانند چرخه پردازنده، حافظه، فایل‌ها، دستگاه های ورودی/خروجی نمونه‌هایی از منابع هستند. اگر سیستمی دارای دو پردازنده باشد، منابع با نوع پردازنده دارای دو نمونه (Instance) است. در صورتی که یک فرآیند یک نوع منبع را درخواست کند، اختصاص هر یک از نمونه‌های آن منبع، تقاضای فرآیند را برآورده خواهد کرد. یک فرآیند باید قبل از استفاده از یک منبع درخواست استفاده از منبع را داشته باشد و بعد از استفاده از منبع باید آن را آزاد سازد. تعداد منابع درخواستی نباید از تعداد کل منابع موجود در سیستم تجاوز کند. یک فرآیند فقط در طی مراحل زیر می‌تواند یک منبع را به کار گیرد:

۱ـ درخواست (Request): اگر درخواست بلافاصله نتواند برآورده شود. فرآیند درخواست کننده باید تا زمانی که منبع درخواستی خود را به دست آورد، در انتظار به سر ببرد.

۲ـ استفاده (Use): فرآیند می‌تواند روی منبعی که در اختیار دارد کار کند.

۳ـ آزاد سازی (Release): فرآیند منبع را رها می‌کند.

هرگاه هر فرآیندی در مجموعه‌ای از فرآیندها در انتظار رویدادی باشند که تنها به وسیله فرآیندی دیگر از آن مجموعه قابل وقوع است، این مجموعه از فرآیندها در وضعیت بن‌بست قرار دارند. این منابع می‌توانند فیزیکی (چاپگرها و حافظه، ....) و یا منطقی باشند مانند فایل‌ها، سمافورها و مونیتورها.

بن‌بست ها ممکن است منابع با انواع مختلف را نیز در بر گیرند؛ به عنوان مثال، سیستمی با یک چاپگر و یک دستگاه نوارخوان را در نظر بگیرید. فرض کنید فرآیند Pi دستگاه نوارخوان را در اختیار داشته باشد، و فرآیند Pj چاپگر را تصرف کرده باشد. اگرPiدر این وضعیت نیاز به چاپگر و Pj نیاز به نوارخوان داشته باشد، بن‌بست رخ خواهد داد.

۵-۲- شرایط وقوع بن‌بست

وضعیت بن‌بست در یک سیستم فقط هنگامی می‌تواند رخ دهد که **۴ شرط کافمن به طور هم‌زمان در سیستم** برقرار گردند.

۱ـ حداقل یک منبع باید به صورت غیراشتراکی کنترل گردد. به این معنی که فقط یک فرآیند در یک لحظه بتواند منبع را مورد استفاده قراردهد. در صورتی که فرآیند دیگری آن منبع را تقاضا نماید. فرآیند درخواست کننده باید تا زمانی که منبع مورد بحث رها نشده است، در انتظار بماند. (**Mutal Exclusion**)

۲ـ باید فرآیندی موجود باشد که حداقل یک منبع را در اختیار داشته باشد و در حال انتظار برای دریافت منبع دیگر به سر برد. این منبع جدید خود در تصرف فرآیندی دیگر است. (**Hold and wait**)

۳ـ یک منبع فقط توسط فرآیندی که آن را در اختیار دارد می‌تواند رها گردد. این رهاسازی بر حسب اراده آن فرآیند و بعد از اتمام کار فرآیندداست. **(Non Preemption)**

۴ـ انتظار حلقوی ـ مجموعه‌ای از فرآیندهای در حال انتظار {Po,P1,...Pn} باید وجود داشته باشند، به طوری که Po در حال انتظار برای منبعی باشد که به‌وسیله P1 نگه‌داشته شده است. P1 در حال انتظار برای منبعی باشد که به‌وسیله P2 نگهداشته شده است. $P_{n-1,\ldots\ldots}$ در حال انتظار برای منبعی باشد که به‌وسیله Pn نگه‌داشته شده است و بالاخره Pn در حال انتظار برای منبعی باشد که Po آن را در اختیار دارد. **(CircularWait)**

۵-۳ - گراف تخصیص منابع

بن‌بست‌ها را می‌توان با استفاده از گراف‌های جهت دار با جامعیت بیش‌تری تشریح کرد. این گونه گراف‌ها را **گراف تخصیص منابع سیستمی‌گویند.** این گراف متشکل است از مجموعه‌ای از رأس‌های V و مجموعه‌ای از لبه‌های E. مجموعه رأس‌های V به دو گونه یا نوع قابل تعریف می‌باشد:

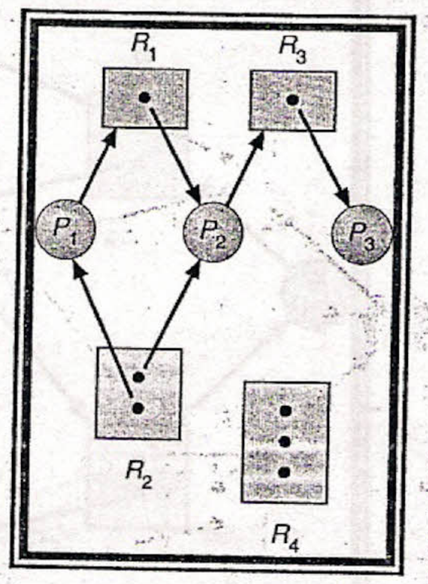
P={P1 ,P2,...,Pn} مجموعه تمام فرآیندهای موجود در سیستم

R={R1,R2,...Rm} مجموعه تمام منابع در سیستم

یک لبه جهت دار به صورت $Pi \rightarrow Rj$ به این مفهوم است که فرآیند Pi نمونه‌ای از منبع R_j را درخواست کرده است و در حال حاضر منتظر به دست آوردن آن منبع می‌باشد. چنین لبه‌ای را Request Edge (لبه تقاضا) می‌نامند.

یک لبه جهت دار به صورت $Rj \rightarrow Pi$ به این مفهوم است که منبع Rj به فرآیند Pi اختصاص یافته است. چنین لبه‌ای را Assignmet Edge (لبه تعلق) می‌نامند.

هر فرآیند Pi را با یک دایره و هر منبع Rj را با یک مربع نشان می‌دهیم. از آن‌جایی که هر منبع می‌تواند دارای نمونه‌های متعدد باشد، هر نمونه را در داخل مربع منبع با یک نشان می‌دهیم. گراف تخصیص منابع زیر را در نظر بگیرید:

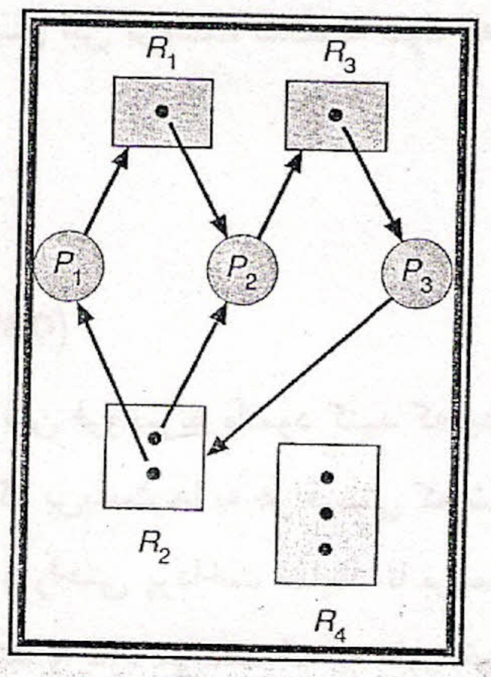


گراف تخصیص منابع

با در نظر گرفتن گراف تخصیص منابع می‌توان به راحتی نشان داد که در صورتی که گراف فاقد حلقه باشد، هیچ فرآیندی در سیستم دچار بن‌بست نشده است. اگر **هر نوع منبع دقیقاً دارای یک نمونه** باشد، **یک حلقه** بیان می‌کند که **بن‌بستی به وقوع پیوسته** است. هرفرآیندی که در حلقه موجود باشد، در بن‌بست قرار دارد. در این حالت حلقه در گراف شرط لازم و کافی برای به وقوع پیوستن بن‌بست خواهد بود.

گراف مثال قبل را در نظر بگیرید، فرض کنید فرآیند P3 نمونه‌ای از منبع نوع R2 را تقاضا کند؛ از آن‌جایی که هیچ نمونه‌ای از این منبع در دسترس نیست، یک لبه تقاضا به گراف افزوده خواهد شد

(لبه P3 → R2)



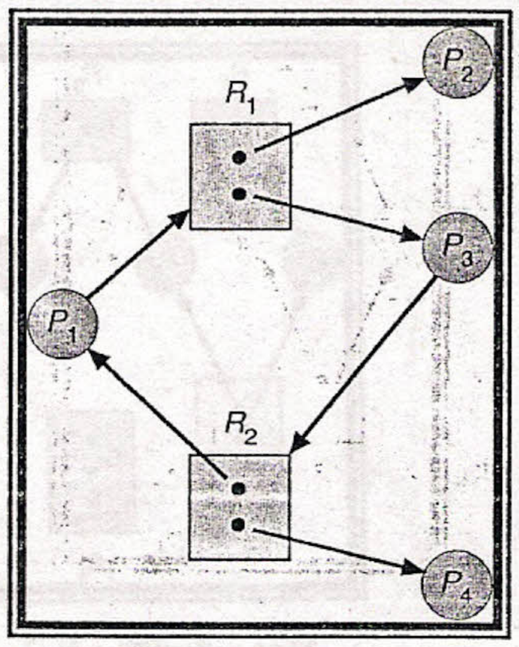
در این لحظه ۲ حلقه در سیستم خواهیم داشت.

P1 → R1 → P2 → R3 → P3 → R2 → P1

P2 → R3 → P3 → R2 → P2

فرآیند **P3,P2,P1 بن‌بست** هستند.

گراف زیر را در نظر بگیرید. در این گراف هم حلقه داریم.



با این حال **بن‌بستی وجود ندارد.** مشاهده کنید که فرآیند P4 ممکن است منبع نوع R2 خود را رها نماید؛ بدین ترتیب این منبع به P3 اختصاص می‌یابد و حلقهٔ شکسته خواهد شد.

۵-۴ - روش‌های کنترل بن‌بست ها

برای برخورد با مساله بن‌بست ۳ راه وجود دارد.

۱ـ مساله را به طور کامل نادیده بگیریم.

۲ـ استفاده از قرار دادهایی که تضمین می کنند سیستم هرگز وارد وضعیت بن‌بست نخواهد شد.

۳ـ اجازه داده شود که سیستم وارد بن‌بست گردد، سپس این بن‌بست شکسته شود که این مورد را بازیافت یا ترمیم (Recovery) گویند.

هریک از روش ها را در بخش‌های ذیل توضیح می‌دهیم.

۵-۴-۱- • روش یا الگوریتم استریخ (ostrich)

ساده‌ترین روش، الگوریتم استریخ است : سر خود را در شن فرو ببرید وانمود کنید که ابداً هیچ مشکلی وجود ندارد. ریاضی دانان این روش را غیر قابل قبول تشخیص می دهند و می‌گویند که بن‌بست ها به هر قیمتی که شده باید جلوگیری شوند. بسیاری از مهندسین مایل نیستند که جریمه زیادی را از نقطه نظر کارایی و راحتی پرداخت نمایند تا بن‌بست را حذف نمایند. سیستم عامل Unix، به بن‌بست تن در می‌دهد و حتی آن‌را کشف هم نمی‌کند واجازه می‌دهد که به طور خودکار منجر به شکست گردد. با فرض این‌که بیش‌تر کاربران ترجیح می دهند که گاه گاه بن‌بست را تحمل نمایند، اما مجبور نباشند که قوانین محدود کننده دیگری را رعایت نمایند، سیستم عامل Unix کل مساله بن‌بست را نادیده می گیرد.

۵-۴-۲-• روشی که تضمین می‌کند، سیستم هرگز وارد وضعیت بن‌بست نخواهد شد

در این زمینه ۲ شیوه وجود دارد:

۱ـ پیشگیری از بن‌بست (Dead Lock Prevention)

۲ـ اجتناب از بن‌بست (Dead Lock Avoidance)

۵-۴-۲-۱ • پیشگیری از بن‌بست

اگر مطمئن شویم که حداقل یکی از ۴ شرط کافمن هرگز بر آورده نمی‌شود، آن‌گاه بن‌بست غیرممکن خواهدشد.

شرط اول: شرط انحصار متقابل می‌باشد. اگر هیچ منبعی تا بحال به یک فرآیند منفرد به صورت انحصاری اختصاص نیافته باشد، هرگز بن‌بست نداشتیم. کاملاً واضح است که اجازه دادن به دو فرآیند برای نوشتن هم‌زمان بر روی چاپگر، نتیجه چاپ را خراب می‌کند. فقط توسط spool کردن خروجی چاپگر، چندین فرآیند می‌توانند در یک زمان خروجی‌های خود را تولید کنند. اما متاسفانه تمام دستگاه‌ها نمی‌توانند spool شوند، بنابراین این شرط را در **کل نمی‌توان تکذیب** کرد.

شرط دوم: نگهداشتن و انتظار می‌باشد. با ملزم کردن فرآیند به درخواست یک‌باره تمام منابع مورد نیاز (TOTAL ALLOCATION) و مسدود کردن آن تا موقعی که تمام منابع در اختیارش گذاشته شود، می‌توان از بروز شرط نگهداشتن و انتظار پیشگیری کرد. این روش از دو جهت نا کار آمد می‌باشد:

۱ـ ممکن است فرآیندی برای مدت طولانی در انتظار تکمیل تمام منابع مورد درخواستش باقی بماند (Starvation) که قحطی‌زدگی را باعث می‌شود.

۲ـ ممکن است برای مدت قابل ملاحظه‌ای منابعی که به یک فرآیند تخصیص داده شده است بی‌استفاده بماند که باعث به هدر رفتن یا کاهش بهره‌وری منبع می‌گردد.

شرط سوم: غیرقابل بازپس گرفتن منبع می‌باشد. راه حلی که می‌توان از بروز این شرط پیشگیری کرد این است که اگر فرآیندی منبعی را درخواست نماید که فرآیند دیگری آن‌را نگه‌داشته است سیستم عامل آن منبع را از فرآیند باز پس گیرد (Preemption) و به فرآیند درخواست کننده اعطا نماید. در این حالت فرآیند از اولویت یکسان برخوردار نمی‌باشند. این رویکرد وقتی عملی است که وضعیت منابع به آسانی بتواند ذخیره گردند و دو باره به آن‌ها بازگردند، مانند ثبات‌ها، پردازنده و حافظه.

شرط چهارم: انتظار چرخشی یا حلقوی می‌باشد. با تعریف یک ترتیب خطی از انواع منابع می‌توان از بروز این شرط پیشگیری کرد. تابع یک به یکی به صورت $F:R \rightarrow N$ تعریف می‌کنیم که در آن N مجموعه اعداد طبیعی و $R=\{R1,R2,...Rm\}$ مجموعه انواع منابع می‌باشد. به هر نوع یک عدد صحیح منحصر به فرد اختصاص می‌دهیم که این اجازه را می‌دهد که دو منبع را با هم مقایسه کنیم. این مقایسه به منظور پی بردن به این نکته است که آیا یکی از آن‌ها از لحاظ ترتیب بر دیگری اولویت دارد یا خیر.

اکنون می‌توانیم پروتکل زیر را برای ممانعت از وقوع بن‌بست دنبال کنیم:

درخواست هر فرآیند برای منابع باید به صورت یک ترتیب صعودی صورت گیرد، به این معنی که در آغاز یک فرآیند می‌تواند هر تعداد نمونه از یک نوع منبع Ri را درخواست نماید. بعد از آن، فرآیند در صورتی می‌تواند تقاضای منبع Rj را داشته باشد که $F(Ri)<F(Rj)$ برقرار باشد.

هر زمانی فرآیندی، نمونه‌ای از منبع نوع Rj را درخواست کند، باید منبع Ri را که در آن رابطه $F(Ri)>=F(Rj)$ برقرار است، رها کرده باشد.

توجه به این نکته حایز اهمیت است که تابع F باید برحسب نیاز متعارف به منابع تعریف شود، برای مثال چون نوارخوان معمولاً قبل از چاپگر استفاده می‌شود، مطلوبست که (چاپگر) F > (نوار خوان) F باشد.

رعایت پروتکل فوق الذکر، ممکن است باعث به هدر رفتن منابع و کاهش بهره‌وری آن‌ها و **نیاز به پیش‌بینی آینده** گردد.

۵-۴-۲-۲ • اجتناب از بن‌بست ها

همان‌طور که توضیح داده شد الگوریتم‌های پیشگیری از بن‌بست (Dead Lock Prevention) مانع بن‌بست می‌شود و این کار را با محدود کردن چگونگی انجام درخواست‌ها انجام می‌دهند. این محدودیت‌ها، مطمئن می‌سازد که حداقل یکی از شرایط لازم برای وقوع بن‌بست هرگز به وقوع نپیوندند، بنابراین بن‌بست نمی‌شود.

تاثیرات جانبی این الگوریتم عبارت‌اند از **استفاده کم از دستگاه ها و کاهش توان عملیاتی سیستم.**

روش دیگر ممانعت به عمل آوردن از وقوع بن‌بست این است که اطلاعات بیش‌تری درباره چگونگی درخواست منابع مهیا گردد. به عنوان مثال در یک سیستم با یک نوارخوان و یک چاپگر ممکن است گفته شود که فرآیند P ابتدا نوارخوان را مورد درخواست قرار خواهد داد و آن‌گاه چاپگر را تقاضا می‌کند. قبل از آزاد سازی هر دو منبع مذکور، ابتدا فرآیند Q از سوی دیگر، چاپگر را درخواست کرده و به دنبال این درخواست تقاضای نوارخوان خواهد نمود. با این علم از دنباله کامل تقاضاها و رهاسازی ها برای هر فرآیند، می‌توان برای هر تقاضا تصمیم گرفت که آیا فرآیند تقاضا کننده برای آن درخواست به وضعیت انتظار درخواهد آمد یا خیر.

هر تقاضایی نیازمند این است که سیستم منابع موجود فعلی را مورد توجه قراردهد. ضمناً منابعی که در حال حاضر به هر فرآیند اختصاص دارند و تقاضاهای آینده و رها سازی های هر یک از فرآیندها باید ملاحظه گردد و سیستم در طی آن تصمیم بگیرد که آیا تقاضای رسیده فعلی می‌تواند بر آورده گردد یا این‌که برای اجتناب از یک بن‌بست احتمالی در آینده این فرآیند باید در حال انتظار بماند.

الگوریتم‌های متفاوت در میزان و نوع اطلاعات مورد نیاز با یکدیگر متفاوتند. ساده‌ترین و مفیدترین مدل نیازمند این است که هر فرآیند حداکثر تعداد منابع از هر نوع را که ممکن است درخواست کند، مشخص نماید. با این اطلاعات قبلی، برای هر فرآیند (ماکسیمم تعداد منابع از هر نوعی که ممکن است توسط فرآیندها درخواست گردد.) این احتمال وجود دارد که الگوریتمی ایجاد شود که اطمینان دهد سیستم هرگز وارد وضعیت بن‌بست نخواهد شد.

این **الگوریتم اجتناب از بن‌بست** (DeadLock –Avoidance)نام دارد. الگوریتم پرهیز از بن‌بست به طور پویا وضعیت تخصیص منابع را بررسی می‌کند و از این نظر اطمینان حاصل می‌شود که شرط انتظار حلقوی هرگز به وقوع نخواهد پیوست.

وضعیت (State) تخصیص منابع به وسیله تعداد منابع موجود و منابع اختصاص داده شده و هم‌چنین حداکثر تقاضای فرآیندها تعریف می‌شود. **حالت مطمئن یا امن (Safe)** حالتی است که در ان سیستم بتواند منابع را به هر یک از فرآیندها (تا حداکثر نیاز آن‌ها) به ترتیبی اختصاص دهد که درضمن آن از وقوع بن‌بست نیز جلوگیری گردد. به عبارت دیگر سیستم زمانی در وضعیت امن قرار دارد که **دنباله امنی** (Safe Sequence) موجود باشد.

دنباله‌ای از فرآیندها <P1,P2,.... . Pn> را دنباله امن می‌نامند، اگر برای هر یک از Pi ها، منابعی که Pi می‌تواند هنوز درخواست کند، به‌وسیله منابع آزاد فعلی به علاوه منابع اختصاص داده شده به تمام Pj ها (j<i) تامین گردد. در این شرایط، اگر منابعی که فرآیند Pi نیاز دارد بلافاصله در دسترس آن قرار نگیرد، Pi می‌تواند تا زمانی که Pj پایان یابد در انتظار بماند. هنگامی که Pj پایان بپذیرد Pi می‌تواند تمام منابع مورد نیازش را به دست آورد و عمل مورد نیازش را به پایان برساند. هنگامی که Pi خاتمه بپذیرد، Pi+1 منابع مورد درخواست خود را به‌دست می‌آورد و این ترتیب ادامه خواهد یافت. در صورتی که چنین دنباله‌ای موجود نباشد می‌گویند سیستم در **وضعیت ناامن** (Unsafe) واقع است.

در یک **وضعیت امن بن‌بست رخ نمی‌دهد.** وضعیت بن‌بست یک وضعیت ناامن است. **تمام وضعیت‌های ناامن، بن‌بست نیستند.** با این وجود هر وضعیت ناامن می‌تواند به یک بن‌بست ختم گردد.

سیستمی با ۱۲ نوارخوان مغناطیسی و سه فرآیند را در نظر بگیرید: P2,P1,P0 فرض کنید در لحظه t0 وضعیت این سیستم مطابق جدول زیر باشد:

فرآیند	حداکثر نیاز	منابع تخصیص یافته	نیاز آتی
P0	10	5	5
P1	4	2	2
P2	9	2	7

در لحظه t0 سیستم در وضعیت امن قرار دارد. دنباله <P1,P0,P2> وضعیت امن را نشان می‌دهد، چون با وجود 3 نوار خوان فعلی بعد از تخصیص فرآیندP1 می‌تواند بلافاصله تمام نوارخوان‌های مورد نیازش را به خود اختصاص دهد و سپس آن‌ها را به سیستم بازگرداند(در این صورت سیستم دارای ۵ نوارخوان آزاد خواهد بود). فرآیند P0 خواهد توانست تمام نـــوارخوان های مورد نیاز خود را دریافت کند و سپس آن‌ها را آزاد کند(سیستم در این حالت ۱۰ نوارخوان آزاد خواهد داشت) و نهایتاً فرآیند P2 خواهد توانست تمام نوارخوان‌های مورد نظر خود را به خود اختصاص دهد و بعد از پایان کارش آن‌ها را به سیستم بازگرداند که به این ترتیب ۱۲ نوارخوان آزاد در سیستم خواهیم داشت.

این احتمال وجود دارد که از یک حالت امن به یک وضعیت ناامن تغییر حالت یا وضعیت داده شود؛ فرض کنید در لحظه t1 فرآیند P2 تقاضایی را صادر کند و یکی از نوار خوان ها به آن اختصاص داده شود. در این صورت سیستم در **وضعیت امن قرار ندارد.** در این لحظه تنها فرآیندP1 می‌تواند تمام نوارخوان‌های مورد نیازش را دریافت کند و بعد از آزاد کردن آن‌ها سیستم فقط دارای ۴ نوارخوان آزاد خواهد بود. چون فرآیندP0، ۵ نوارخوان در اختیار دارد. اما ۱۰ نوارخوان مورد احتیاج آن است، در صورت تقاضای P0 برای ۵ نوارخوان دیگر، این فرآیند به حالت انتظار درخواهد آمد. به طور مشابه فرآیند P2 ممکن است تقاضای ۶ نوار خوان دیگر نماید و در نتیجه باید در حال انتظار باقی بماند و **لذا بن‌بستی به وقوع خواهد** پیوست.

اشتباه در این‌جا موافقت کردن با تقاضای فرآیند P2 برای در اختیار گرفتن یک نوارخوان دیگر است. در صورتی که P2 را تا زمانی که یکی از فرآیندهای دیگر به پایان برسد و منابع خود را آزاد کند، در حال انتظار قرار می‌دادیم، از وقوع بن‌بست جلوگیری می‌شد.

بنابراین هر زمان فرآیندی منبعی تقاضا کند و منبع در ان زمان در دسترس باشد، سیستم باید تصمیم بگیرد که آیا فرآینـد مـی‌توانـد بلافاصله منبع را در اختیار داشته باشد یاخیر و باید در حال انتظار بماند. زمانی که با اختصاص منبع موافقت می‌شود که این تخصیص، سیستم را در وضعیت امن نگه‌دارد.

هنگامی که یک فرآیند جدید وارد سیستم می‌گردد. باید حداکثر تعداد نمونه‌هایی را که از هر نوع منبع نیازخواهد داشت، را مشخص کند. بدیهی است که این تعداد نباید از حداکثر تعداد منابع موجود در سیستم تجاوز کند. هنگامی که کاربردی مجموعه‌ای از منابع را درخواست می‌کند، سیستم باید تعیین کند آیا اختصاص این منابع سبب خارج شدن سیستم از حالت امن خواهد شد یا خیر.

برای پیاده سازی این الگوریتم که بنام **الگوریتم بانکداری** معروف است، نیاز به ساختارهای داده‌ای چندی داریم. این ساختارهای داده‌ای وضعیت تخصیص منابع سیستم را نشان می دهند.

فرض کنید n تعداد فرآیندهای سیستم باشد و m تعداد منابع موجود در سیستم را نشان دهد.

ساختمان داده‌های زیر مورد نیاز هستند:

۱ـ Available آرایه ای به طول m که تعداد منابع موجود برای هر نوع منبع را مشخص می‌کند عبارت available[j] =K یعنی ازمنبع نوع Rjتعداد K نمونه در دسترس می‌باشد.

۲- Max یک ماتریس n*m که حداکثر نیاز هر فرآیند را معین می‌کند. اگر max[i,j]=k باشد، فرآیند Pi ممکن است حداکثر K نمونه از منبع نوع R_j را درخواست کند.

۳- Allocation یک ماتریس n*m تعریف کننده تعداد منابعی که در حال حاضر به هر فرآیند اختصاص یافته است. Allocation[i,j]=K یعنی فرآیند Pi در حال حاضر K نمونه از منبع نوع Rj را در اختیار دارد.

۴- need یک ماتریس n*m است که منابع باقیمانده را که یک فرآیند ممکن است درخواست نماید، نشان می‌دهد. need[i,j]=K به این مفهوم است که فرآیند Pi ممکن است برای تکمیل کردن کارش به K نمونه از منبع Rj، علاوه برآن‌چه در حال حاضر در اختیار دارد، نیاز داشته باشد.

توجه کنیدکه :

need[i,j] =max [i,j] – Allocation[i,j]

برای ساده کردن نمایش الگوریتم، نمادهایی را معرفی می نماییم. فرض کنید Y,X آرایه هایی با طول n باشند. آن‌گاه :

$X \le Y$ IF $X[i] \le Y[i]$ For i = 1,2,3,. . ,n

الگوریتم بانکداری

فرض کنید Request (i) یک بردار تقاضا برای فرآیند Pi باشد، اگرRequest(i) [j]=K، آن‌گاه فرآیند Pi به K نمونه از منبع نوع Rj نیاز دارد. هنگامی که به وسیله فرآیند Pi تقاضایی برای منابع رخ دهد، مراحل زیر صورت می‌پذیرد.

۱- اگر need(i)≥Request(i) به مرحله ۲ برو. در غیر این‌صورت یک وضعیت خطا رخ داده است.

۲- اگرRequest(i) ≤ Available به مرحله ۳ برو. در غیراین صورت Pi باید در حال انتظار باقی بماند، چراکه منابع آزاد نیستند.

۳- سیستم منابع درخواست شده را به فرآیند Pi اختصاص می‌دهد و اصلاحاتی به شرح زیر انجام می‌شود.

Available(i) =Available(i) –Request(i)

Allocation(i)=Allocation(i)+ Request(i)

Need(i) = Max (i) –Allocation (i)

درصورتی که وضعیت تخصیص منابع، امن باشد، انتقال کامل می‌شود و فرآیند Pi منابع‌اش را در اختیار می‌گیرد. با این حال، اگر وضعیت جدید ناامن باشد، Pi باید برای Request(i) در انتظار بماند و وضعیت تخصیص منابع پیشین حفظ خواهد شد.

الگوریتم ایمن بودن:

الگوریتم برای پیدا کردن این‌که آیا سیستم در وضعیت امن قرار دارد یا خیر، مطابق روش زیر عمل می‌نماید:

۱- فرض کنید Work و Finish ارایه هایی به طول n,m باشند. موارد زیر را به عنوان مقادیر اولیه قرار دهید:

Work= Available

Finish[i] = False For i=1,2,…n

۲- i ای را پیدا کنید که شرایط زیر برای آن برقرار باشند.

Finish[i] = False

Need(i) ≤ work

۳ـ در صورتی که چنین i ای موجود نباشد. به مرحله ۴ برو

Work=Work+ Allocation
Finish(i) = True

انتقال به مرحله ۲

۴ـ اگر برای تمام مقادیر i, Finish[i] = True سیستم در وضعیت امن قرار دارد.

این الگوریتم برای تصمیم‌گیری حالت امن، به m x (n**2) عمل نیازمند است.

این نکته قابل ذکر است که الگوریتم بانکداری از نظر **تئوری بسیار جالب** می‌باشد، اما **در عمل به طور موثر استفاده نمی‌شود**، چرا که به ندرت اتفاق می افتد که فرآیند ها پیشاپیش از حداکثر نیاز نهایی خود به منابع آگاهی داشته باشند. علاوه بر آن، تعداد فرآیندها ثابت نمی‌باشند، بلکه به صورت پویا با داخل و خارج شدن کاربران تغییر می‌نماید. یا حتی منابعی که گمان می‌رود که در اختیار سیستم قرار دارند، ممکن است به طور اتفاقی به‌علت خرابی یا هر حادثه غیرقابل پیش بینی بلااستفاده شوند.

مثال: سیستمی با پنج فرآیند P0 تا P4 و سه منبع A .B .C را در نظر بگیرید. منبع نوع A دارای ۱۰ نمونه، نوع B دارای ۵ نمونه و منبع نوع C دارای ۷ نمونه است. فرض کنید که در زمان T0 وضعیت سیستم مطابق جدول زیر باشد.

فرآیند	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

محتویات ماتریس Need نیز به صورت زیر است : Max - Allocation

فرآیند	Need		
	A	B	C
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

دنباله <P1,P3,P4,P2,P0> یک وضعیت امن را معین می‌کند.

فرض کنید اکنون فرآیند P1 یک نمونه دیگر از منبع نوع A و دو منبع نوع C را درخواست کند؛ بنابراین، Request (1) =(1,0,2).

برای تصمیم‌گیری درباره این‌که این تقاضا بتواند بلافاصله برآورده شود، ابتدا بررسی می‌کنیم که Available ≥ Request(1) مشاهده می‌شود که این شرط برقرار است.

بعد از برآورده شدن تقاضای فوق، وضعیت جدید مطابق جدول زیر خواهد شد:

فرآیند	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	4	3	2	3	0
P1	3	0	2	0	2	0			
P2	3	0	2	6	0	0			
P3	2	1	1	0	1	1			
P4	0	0	2	4	3	1			

در این‌جا باید معین گردد که آیا این وضعیت جدید سیستم امن است یا خیر. برای انجام این بررسی از الگوریتم ایمن بودن استفاده می‌شود. در می یابیم<P1,P3,P4,P0,P2> وضعیت امن را نشان می‌دهد. بنابراین می‌توانیم بلافاصله با درخواست فرآیندP1 موافقت نماییم.

با درخواست (۳،۳،۰) که توسط فرآیند P4 انجام می‌گیرد، موافقت نمی‌شود؛ چرا که منابع مورد نیاز در دسترس نیستند. درخواست(۰،۲،۰) که به‌وسیله P0 انجام می‌گیرد، حتی با وجودی‌که منابع از نظر تعداد در دسترس هستند، نمی‌تواند برآورده شود. چراکه وضعیت حاصله یک وضعیت ناامن است.

اگر چه الگوریتم بانکداری کاملاً کلی است و برای هر سیستم تخصیص منابع کار می‌کند، اما در صورتی که سیستم تخصیص منابعی داشته باشیم که در آن برای **هر نوع منبعی تنها یک نمونه** موجود باشد، یک الگوریتم کاراتر می‌تواند به کار گرفته شود. این الگوریتم از یک گراف تخصیص منابع استفاده می‌کند که در آن علاوه بر لبه‌های تقاضا، انتصاب یک لبه با نوع جدید تعریف می‌گردد که آن را **لبه مطالبه یا ادعا (Claim)** می‌نامند.

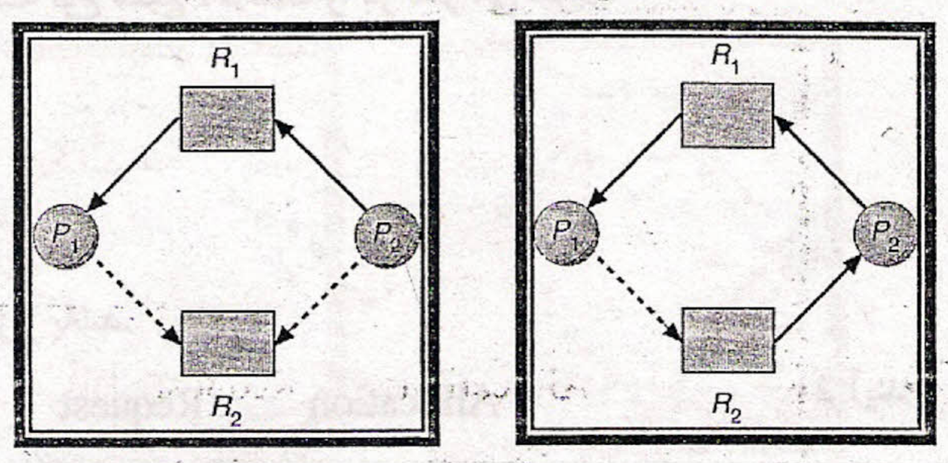
یک لبه مطالبه به شکل $Pi \rightarrow Rj$ بیان می‌کند که فرآیند Pi ممکن است منبع Rj را در آینده در اختیار بگیرد. این لبه‌ها با خط چین نشان داده می‌شوند. هنگامی که فرآیند Pi منبع Rj را درخواست نماید، لبه مطالبه $Pi \rightarrow Rj$ به یک لبه تقاضا تبدیل می‌گردد. و هنگامی که یک منبع Rj به وسیله فرآیند Pi آزاد گردد، لبه تعلق $Rj \rightarrow Pi$ دوباره به لبه مطالبه $Pi \rightarrow Rj$ تبدیل می‌شود.

قبلاً ذکر شدکه منابع باید پیش تر در سیستم درخواست و مطالبه شوند؛ به این معنی که قبل از این‌که اجرای فرآیند Pi شروع گردد، تمام لبه‌های مطالبه‌این فرآیند باید ظاهر شده باشند.

فرض کنید که فرآیند Pi منبع Rj را تقاضا کند. این تقاضا فقط هنگامی می‌تواند برآورده شود که تبدیل لبه مطالبه$Pi \rightarrow Rj$ به یک لبه تعلق $Rj \rightarrow Pi$ نوعی حلقه را در گراف تخصیص منابع نتیجه ندهد. توجه کنید که حالت امن به‌وسیله یک الگوریتم تشخیص حلقه بررسی خواهد شد.

در صورتی که حلقه ای موجود نباشد، اختصاص منبع سبب می‌گردد که سیستم در وضعیت امن باقی بماند و اگر حلقه ای ایجاد شود سیستم در اثر این تخصیص وارد وضعیت ناامن می‌گردد. بنابراین فرآیندPi باید در حال انتظار بماند.

گراف تخصیص منابع شکل زیر را در نظر بگیرید. فرض کنید که P2 منبع R2 را درخواست می‌کند. اگر چه R2 در حال حاضر آزاد است. اما نمی‌تواند به P2 اختصاص یابد، چرا که این عمل چرخه‌ای در گراف ایجاد خواهد کرد. این چرخه نشان می‌دهد که سیستم در یک حالت نا امن قرار گرفته است. در صورتی که P1 منبع R2 را تقاضا کند، یک بن‌بست به وقوع می پیوندد.



یک وضعیت ناامن در گراف تخصیص منابع

۵-۴-۳ – تشخیص بن‌بست (Deadlock Dectection)

اگر در سیستم از الگوریتم Deadlock - Prevention و یا Deadlock – Avoidance استفاده نگردد. یک حالت بن‌بست ممکن است به وقوع بپیوندد. در این محیط، سیستم باید:

- الگوریتمی در نظر بگیرد که وضعیت سیستم را بررسی کند و معین نماید که آیا بن‌بستی رخ داده یا خیر.
- الگوریتمی برای بازیابی از وضعیت بن‌بست مهیا نماید.

الگوریتم تشخیص بن‌بست از ساختمان داده‌های زیر استفاده می‌کند:

Available ـ ارایه ای از طول m که تعداد منابع در دسترس از هر نوعی را نشان می‌دهد.

Allocation ـ یک ماتریس n*m است و تعریف کننده تعداد منابع از هر نوعی است که در حال حاضر به هر فرآیند اختصاص داده شده است.

Request ـ یک ماتریس n*m است که تقاضای فعلی هر فرآیند را بیان می‌کند. اگر Request [i,j] =K (فرآیند Pi، متقاضی K نمونه از منبع نوع R_j است.)

۱- فرض کنید Finish , Work آرایه ای با طول n,m باشند. مقادیر آغازی عبارت‌انداز:

```
Work= Available
For i=1 to n
If Allocation(i) <> 0 Then Finish [i] = False
Else Finish [i] = True
```

۲ ـ شاخص i را به گونه‌ای بیابید که

```
Finish [i] =  False
Request [i] < = Work
```

۳ـ درصورتی که چنین i ای وجود نداشته باشد، به مرحله ۴ بروید.

```
Work= Work+ Allocation(i)
Finish[i] = True
```

به مرحله ۲ بروید.

۴ـ اگر Finish [i] = False برای مقادیری از i برقرار باشد (1<=i<= n)، سیستم در وضعیت بن‌بست واقع است. به علاوه اگر Finish [i] = False فرآیند Pi بن‌بست است.

به عنوان مثال سیستم با ۵ فرآیند P0 تا P4 و سه نوع منبع C,B,A را در نظر می‌گیریم.

منبع A ← ۷ نمونه

منبع B ← ۲ نمونه

منبع C ← ۶ نمونه

فرض کنید که در لحظه t0 سیستم در وضعیت زیر باشد.

فرآیند	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0	0	0	0
P1	2	0	0	2	0	2			
P2	3	0	3	0	0	0			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	2			

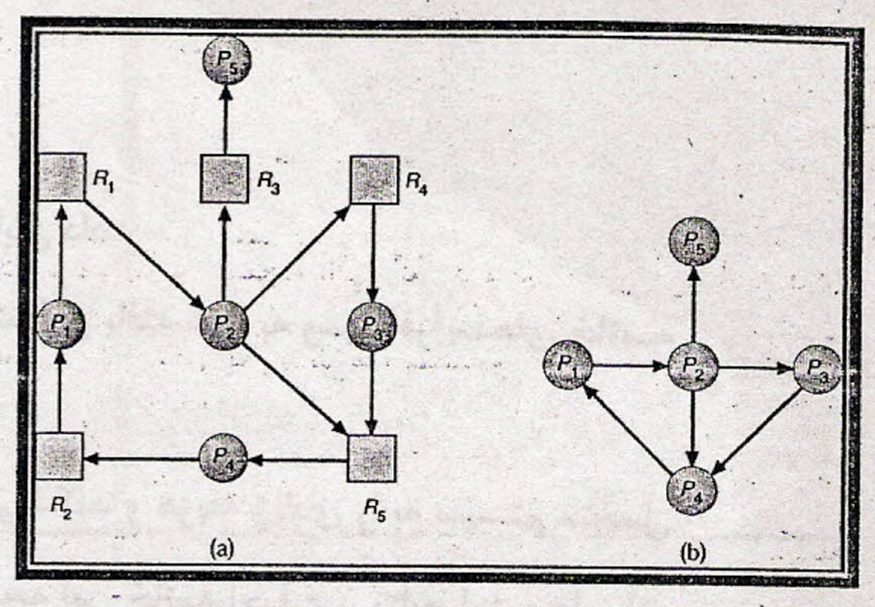
مدعی هستیم که سیستم در **وضعیت بن‌بست نیست**. برای دنباله <P0,P2,P3,P1,P4> برای تمام مقادیر i رابطه Finish [i] = True برقرار است.

فرض کنید در یک لحظه فرآیند P2 یک تقاضای دیگر برای دست‌یابی به نمونه‌ای از منبع C داشته باشد. ماتریس Request به صورت زیر خواهد بود.

	Request		
	A	B	C
P0	0	0	0
P1	2	0	2
P2	0	0	1
P3	1	0	0
P4	0	0	2

در این شرایط ادعا می‌کنیم که سیستم در حالت بن‌بست است، فرآیندهای P4,P3,P2,P1 در بن‌بست هستند.

الگوریتم تشخیص بن‌بست در بخش قبل معرفی شد. در صورتی که تمامی منابع فقط دارای یک نمونه باشد این الگوریتم می‌تواند با الگوریتم سریع‌تری جایگزین گردد. در این الگوریتم از نوعی گراف تخصیص منبع به نام **گراف انتظار** استفاده می‌شود. این گراف از گراف تخصیص منبع به دست می‌آیدکه در آن گره‌های متناظر با منابع و لبه‌های نظیر آن‌ها حذف می‌گردد. به عبارت دقیق تر، یک لبه از Pi به Pj در یک گراف انتظار به این مفهوم است که فرآیند Pi در انتظار فرآیند Pj است، تا این‌که Pj منبعی را که مورد نیاز Pi است، رها نماید. لبه Pi → Pj در یک گراف انتظار وجود دارد اگر و فقط اگر گراف تخصیص منابع متناظرش شامل دو لبه Pi → Rq و Rq → Pj باشد که در آن Rq نوعی منبع است. به عنوان مثال در شکل زیر یک گراف تخصیص منبع و یک گراف انتظار معادلش نشان داده شده است.



گراف تخصیص منابع　　گراف انتظار

د د د اگر و فقط اگر گراف انتظار یک چرخه یا حلقه داشته باشد. برای تشخیص بن‌بست‌ها، سیستم باید نهداری کند و به طور دوره ای الگوریتمی را که یک حلقه در گراف را جستجو می‌کند، فرا بخواند.

ن آ الگوریتم تشخیص چه زمانی فراخوانی می‌گردد؟ جواب به دو عامل بستگی دارد:

لر احتمال دارد که بن‌بستی رخ دهد؟

نگا قوع بن‌بست متاثر خواهند شد؟

ها رخ دهد، الگوریتم تشخیص نیز ناچار باید بارها فراخوانی گردد.

فرآ دهای بن‌بست شده، تا زمانی که بن‌بست شکسته شود، بیکار خواهند بود؛ به علاوه تعداد فرآیندهای ممکن است افزایش یابد.

د ه آید که فرآیندی تقاضایی صادر کند که بلافاصله قابل جوابگویی نباشد. این احتمال وجود دارد که این شد ه زنجیره فرآیندهای منتظر را کامل می‌کند. بنابراین، می‌توان الگوریتم تشخیص بن‌بست را، هر زمان که یک منبع بلافاصله نتواند برآورده شود، فراخوانی کرد. در این حالت، نه تنها قادر خواهیم بود مجموعه ش اند، تشخیص دهیم، بلکه این امکان نیز وجود دارد که فرآیند مسبب نیز شناسایی گردد.

رآیندهای بن‌بست شده پیوندی است در چرخه گراف انتظار، بنابراین، تمام آن‌ها سبب بن‌بست هستند). در قاو منبع موجود باشد، یک تقاضا ممکن است چرخه های بسیاری را در گراف منابع ایجاد کند. هر چرخه رسیده کامل می‌گردد.

نش ص بن‌بست برای هر تقاضا، ممکن است سربار قابل توجهی در زمان محاسبه ایجاد کند. شیوه بسیار کم ن این است که این الگوریتم در فاصله های زمانی طولانی تر فراخوانی گردند و یا این‌که هر زمان کارایی رص رسید، الگوریتم فراخوانی شوند، ممکن است چرخه های بسیاری در گراف منابع ایجاد گردد، و معمولاً یک از چندین فرآیند بن‌بست شده سبب ایجاد بن‌بست بوده اند.

(Recovery From Deadlock)

تش یص بن‌بست تعیین کند که بن‌بستی رخ داده است، تصمیمات متفاوتی می‌توان اتخاذ کرد. یک راه، آن را از وقوع بن‌بست با خبر کنیم و او به **صورت دستی** مشکل را حل نماید. راه حل دیگر ان است که خود از بست خارج گردد و به عبارتی دیگر بازگشت نماید. برای شکستن یک بن‌بست دو شیوه وجود دارد :

راه حل اول این است که به سادگی یک یا چند فرا یند را مختل سازد، به نحوی که انتظار حـ
که چند منبع را از یک یا چند فرآیند بن‌بست بازپس گیرد.

خاتمه دادن فرآیند

به منظور حذف بن‌بست‌ها می‌توان فرآیندی را بی نتیجه باقی گذارد و به اجرای آن پایان داد.
دو شیوه می‌تواند در این راستا به کار آید. در هر دوی این روش‌ها، سیستم تمام منابع اختصاص یافته شده به وسیله فرآیندهای خاتمه یافته را باز پس می‌گیرد.

۱ـ خاتمه دادن فرآیندهای بن‌بست شده: این شیوه به وضوح چرخه بن‌بست را می‌شکند و هزینه زیادی را به سیستم متحمل خواهد کرد. این فرآیندها ممکن است مدت زمان زیادی محاسبه داشته باشند و در نتیجه این خاتمه اجباری، نتایج این محاسبات ناتمام باید از بین برود و احتمالاً در زمانی دیگر دوباره محاسبه گردد.

۲ـ خاتمه فرآیندها به صورت منفرد تا زمانی که چرخه بن‌بست حذف گردد: این شیوه، سربار قابل توجهی را نتیجه می‌دهد. چون، بعد از این‌که هر یک از فرآیندها خاتمه یابد. باید یک‌بار الگوریتم تشخیص بن‌بست فراخوانی شود تا این‌که معین گردد، ایا هیچ فرآیندی هنوز در بن‌بست قرار دارد یاخیر.

توجه به این نکته حایز اهمیت است که خاتمه و مختل کردن یک فرآیند ممکن است آسان نباشد. اگر فرآیندی در میانه به هنگام سازی یک فایل باشد، خاتمه دادن به آن در این لحظه سبب خواهد شد تا فایل با یک وضعیت غلط رها گردد و یا اگر فرآیندی در اواسط چاپ داده‌ها روی یک چاپگر باشد، سیستم باید قبل از اقدام به چاپ کردن کاربعدی، وضعیت چاپگر را به یک وضعیت مناسب بازگرداند.

درصورتی که از شیوه دوم استفاده گردد، باید معین شود که کدام فرآیند (یا فرآیندها) باید برای شکستن بن‌بست خاتمه بپذیرد، که این مساله شبیه مساله زمان‌بندی پردازنده می‌باشد. پاسخ این است: آن فرآیندهایی که خاتمه آن‌ها حداقل هزینه را بر سیستم تحمیل می‌کند فرآیندهای انتخابی خواهند بود.

متاسفانه عبارت حداقل هزینه یک ملاک جامع و دقیق نیست، عوامل چندی برای تعیین فرآیند انتخابی وجود دارند که عبارت‌انداز:

۱. حق تقدم فرآیند
۲. مدت زمانی که از محاسبه فرآیندی می‌گذرد و این‌که فرآیند چه مدت زمانی قبل از کامل شدن کار معین شده اش باید محاسبه گردد.
۳. این فرآیند منابع را به چه تعداد و از چه نوعی به کار گرفته است (به عنوان مثال آیا این منابع ساده اند و یا غیرانحصاری).
۴. فرآیند برای کامل شدن چه تعداد منابع دیگری را لازم دارد؟
۴. لازم است فرآیندها تا چه تعداد خاتمه یابد؟
۵. آیا فرآیند از نوع محاوره‌ای است یا دسته‌ای؟

بازپس گیری منابع

برای حذف بن‌بست از طریق باز پس گیری منابع، منابع چندی از فرآیندها بازپس گرفته می‌شود و این منابع به فرآیندهای دیگر اختصاص می یابند، تا این‌که چرخه بن‌بست شکسته شود.

در صورتی که بازپس گیری در برخورد با مساله بن‌بست ها مورد نیاز باشد مراحل کار به صورت زیر است:

۱ـ کدام منابع و کدام فرآیندها باید باز پس گرفته شوند؟ در بازپس گیری منابع مانند حالت خاتمه دادن به فرآیند، باید برای خروج از بن‌بست فاکتورهایی را که درهزینه دخیل هستند، مد نظر قرار دهیم.

۲ـ در صورتی که منبعی از فرآیندی بازپس گرفته شود. تکلیف آن فرآیند چه خواهد شد؟ واضح است که ان فرآیند نمی‌تواند به اجرای طبیعی خود ادامه دهد، چرا که برخی از منابع مورد نیازش را از دست داده است. بنابراین باید فرآیند مذکور را به وضعیت امنی بازگردانیم. آن‌گاه در وضعیت جدید، آن فر انید را مجدداً اغاز کنیم. از آن‌جایی که در کل تعیین این‌که وضعیت امن حاصل شده است یا خیر، کار مشکلی است. ساده‌ترین راه حل بازگشت کلی فرآیند است؛ یعنی فرآیند به طور کلی مختل شود و آن‌گاه مجدداً اجرای آن آغاز گردد. اگر چه بازگرداندن فرآیندها فقط به اندازه مورد نیاز برای شکستن بن‌بست بسیار کاراتر است، از سوی دیگر، لازمه این شیوه آن است که اطلاعات بیش‌تر درباره وضعیت تمام فرآیندهای درحال اجرا نگهداری گردد.

۳ـ چگونه می‌توان مطمئن بود که قحطی منابع یا کمبودِ منابع به وقوع نخواهد پیوست ؟ به این معنی که چگونه می‌توان تضمین نمود که منابع همیشه از فرآیند یکسانی بازپس گرفته نمی‌شوند؟ در سیستمی که انتخاب فرآیند قربانی (جهت بازپس گیری منابع از آن) تنها مبتنی بر عوامل هزینه است، این امکان وجود دارد که یک فرآیند یکسان همیشه به عنوان قربانی انتخاب گردد. در نتیجه این فرآیند هرگز قادر نخواهد بود که کار خود را به پایان برساند. واضح است که باید اطمینان حاصل گردد که یک فرآیند فقط در تعداد محدودی به عنوان فرآیند قربانی انتخاب می‌شود که متعارف ترین راه حل آن، وارد کردن تعداد بازگشت‌های یک فرآیند، به عنوان یکی از فاکتورهای هزینه ای است.

روش تشخیص و بازگشت از بن‌بست، روشی است که اغلب روی کامپیوترهای بزرگ استفاده می‌شود. به‌خصوص سیستم‌های دسته‌ای که در آن‌ها خاتمه دادن به یک فرآیند و آغاز مجدد آن معمولاً امکان پذیر است.

۵-۵- یک راهبرد مجتمع برای بن‌بست‌ها

هر یک از راهبردهای برخورد با بن‌بست نقاط قوت و ضعف خاص خودرا دارند. شاید استفاده از راهبردهای متفاوت در شرایط متفاوت، کارآمدتر باشد. رویکرد ذیل توسط "Silberchatz" پیشنهاد شده است.

ـ تقسیم بندی منابع در تعدای از گروه‌های مختلف.

ـ استفاده از راهبرد مرتب سازی خطی بین گروه‌های منابع.

ـ در داخل یک گروه از منابع، استفاده از مناسب‌ترین الگوریتم برای آن گروه.

گروه‌های ذیل از منابع را در نظر بگیرد:

ـ **فضای قابل مبادله**: بلوک‌های از حافظه ثانوی برای استفاده در مبادله فرآیندها.

ـ **منابع فرآیند**: دستگاه‌های قابل تخصیص مانند گرداننده‌های نوار.

ـ**حافظه اصلی**: حافظه قابل تخصیص به فرآیندها به صورت صفحه‌ها یا قطعه‌ها.

ـ **منابع داخلی**: مثل کانال‌های ورودی/خروجی، بلوک کنترلی فرآیند.

در داخل گروه نیز از راهبردهای زیر می‌توان استفاده کرد:

ـ **فضای قابل مبادله**: با تخصیص یک‌بار منبع مورد نیاز، ازروش پیشگیری از بن‌بست استفاده گردد.

ـ **منابع فرآیند**: غالباً اجتناب از بن‌بست موثر است، به دلیل این‌که انتظار از فرآیندها برای معرفی قبلی منابعی که از این گروه نیاز دارند، معقول می‌باشد.

ـ **حافظه اصلی**: با بازپس گرفتن منبع، پیشگیری از بن‌بست در این نوع منابع امکان پذیر است.

ـ **منابع داخلی**: با مرتب کردن منابع، پیشگیری از بن‌بست برای این منابع مناسب می‌باشد.

تست‌های مربوط به مبحث بن‌بست ها

۱ـ سیستمی شامل ۴ پردازه هم‌زمان و ۲ منبع قابل استفاده مجدد را در نظر بگیرید. به شرط این‌که هر پردازه حداکثر به ۲ منبع نیاز داشته باشد، تعداد وضعیت‌های بن‌بست در این سیستم حداکثر چند حالت می‌باشد.

الف) ۳ ب) ۵ ج) ۶ د) ۸

۲ـ الگوریتم BANKER در رابطه با کدام‌یک از موارد زیر به‌کار می‌رود؟

الف) کشف DeadLock در سیستم. ب) پیشگیری ازDeadLock .

ج) اجتناب از DeadLock. د) هیچ‌کدام.

۳ـ در یک کامپیوتر که دارای شش نوارگردان(Tape Drivers) می‌باشد، n پروسس برای در انحصار گرفتن این نوار گردان‌ها رقابت می کنند. برای چه مقدار n سیستم با DeadLock مواجه نخواهد شد؟

الف) n=۶

ب) n می‌تواند هر مقدار بین ۵ و بی‌نهایت باشد به شرط آن‌که در ابتدا فقط ۵ پروسس را به حالت Ready ببریم و به هر کدام فقط یک نوارگردان اختصاص دهیم.

ج) با اطلاعات داده شده نمی‌توان مقدار n را دقیقاً مشخص کرد.

د) با هر مقدار n که بیش از ۶ باشد، سیستم به هر حال با DeadLock مواجه خواهد شد.

۴ـ به یک سیستم اشتراک زمانی که هم‌زمان حداکثر هشت پردازش را فراهم می‌کند ۲۰ عدد نوار گردان متصل است. تعداد نا محدودی کار به سیستم ارجاع می‌شود که هر یک برای تکمیل عملیات خود به حداکثر چهار نوار گردان نیاز دارد. هر کارمی‌تواند با سه نوار گردان شروع به کار کرده و برای مدت طولانی ادامه دهد و در اواخر عملیات نوار گردان چهارم را درخواست نماید. اگر زمان‌بندی مورد استفاده در این سیستم اشتراک زمانی تا زمانی که چهار گرداننده نوار در دسترس نباشد کاری را شروع نکند و اگر کاری شروع شد، بلافاصله چهار گرداننده به آن اختصاص داده شود و تا پایان عملیات نوار گردان‌ها رها نگردند، (a) حداکثر تعداد کارهایی که می‌توانند به صورت هم‌زمان در حال پیشرفت باشند، (b) حداکثر نوار گردان‌هایی که بی‌کار می‌مانند و (c) حداقل نوار گردان‌هایی که بیکار می‌مانند چه تعداد می‌باشند؟

الف) (a) ۸ (b) ۴ (c) ۱ ب) (a) ۶ (b) ۲ (c) ۲

ج) (a) ۷ (b) ۱ (c) ۱ د) (a) ۵ (b) ۵ (c) ۰

۵ـ سیستمی دارای پنج پردازه (P0-P4) و چهار منبع (resource)، در حالت (state) زیر قرار دارد.

تعداد کل منابع اولیه

R0	R1	R2	R3
۶	۳	٤	۲

منابعی که هنوز مورد نیاز است

R0	R1	R2	R3
۱	۱	۰	۰
۰	۱	۱	۲
۳	۱	۰	۰
۰	۰	۱	۰
۲	۱	۱	۰

منابع تخصیص یافته

R0	R1	R2	R3
۳	۰	۱	۱
۰	۱	۰	۰
۱	۱	۱	۰
۱	۱	۰	۱
۰	۰	۰	۰

در چه صورتی احتمال وقوع بن‌بست وجود دارد؟

الف) پردازه P0 یک واحد از منبع R2 را در خواست کند.

ب) پردازه P1 یک واحد ازمنبع R2 را درخواست کند و پردازه P3 آخرین واحد باقیمانده R2 را درخواست کند.

ج) پردازه P1 یک واحد از منبع R2 را درخواست کند و پردازه P4 آخرین واحد باقیمانده R2 را درخواست کند.

د) پردازه P3 یک واحدازمنبع R2 رادرخواست کند و پردازه P4 کلیه منابع مورد نیازش را درخواست کند.

۶ـ یک سیستم عامل در حال اجرا متشکل از چند پردازه همروند را در نظر بگیرید که در حالت بن‌بست نمی‌باشند. اگر یکی از این پردازه ها در وضعیت بلوکه قرار گیرد، آیا سیستم عامل می‌تواند تشخیص دهد که این پردازه در انتظار رویدادی است که هرگز اتفاق نخواهد افتاد؟

الف) خیر

ب) در سیستم‌های تک کاربره بله، ولی در سیستم‌های چند کاربره خیر.

ج) در سیستم‌های تک پردازنده بله، ولی در سیستم‌های چند پردازنده خیر.

د) سیستم عامل با کنترل زمان سنج قادر به تشخیص است.

۷ـ سیستمی متشکل از چهار پردازه P1 و P2 وP3 و P4 و سه نوع منبع قابل استفاده مجدد S1 وS2 و S3 را در نظر بگیرید. تعداد واحدهای هر منبع به ترتیب t1=3 و t2=2 و t3=2 است. P1 یک واحد از منبع S1 را در اختیاردارد و تقاضای یک واحد از S2 را کرده است. P2 دو واحد از منبع S2 را در اختیاردارد وتقاضای یک واحد از S1 را کرده است. P3 یک واحد از منبع S1 را در اختیار دارد و تقاضای یک واحد از S2 را کرده است. P4 نیزدو واحد از منبع S3 را در اختیاردارد و تقاضای یک واحد از S1 را کرده است. در وضعیت موجود کدام‌یک از پردازه ها در بن‌بست قرار ندارند؟

الف) P2 و P1 ب) P3 وP1 ج) P4 و P2 د) P4 وP3 وP2 و P1

۸ـ n پردازه ازm منبع به طور اشتراکی استفاده می کنند. گرفتن و آزاد کردن این منابع یکی یکی صورت می‌گیرد و حداکثر نیاز هر پردازه به منابع از m تجاوز نمی‌کند و کل نیاز پردازه‌ها نیز کمتر از m+n است. در مورد این حالت کدام گزینه صحیح است؟

الف) اگر n<=m باشد، امکان وجود بن‌بست وجود دارد.

ب) اگر m<=n باشد، امکان وجود بن‌بست وجود دارد.

ج) در این سیستم هیچ‌گاه بن‌بست اتفاق نمی افتد.

د) وقوع بن‌بست به رابطه بین n وm بستگی ندارد و به هر حال باید برای جلوگیری از بن‌بست در این سیستم آزاد کردن و گرفتن منابع را به صورت ناحیه‌ای بحرانی پیاده سازی کرد.

۹ـ شبکه‌ای از صندوق پستی (Mailbox) و دو دستور send و Receive برای تبدل پیغام بین فرآیندها استفاده می‌کند. دستور Receive فرآیندی را که باید پیغام آن دریافت شود مشخص می‌کند، و در صورت عدم وجود پیغام ازآن فرآیند، بلوکه می‌شود. منابع مشترک وجود ندارد، ولی فرآیندها نیازمند تماس با یکدیگر جهت انجام برخی از امور هستند. آیا امکان بروز بن‌بست در این شبکه وجود دارد؟

الف) خیر، چون دستور send و Receive به سنکرون کار می کنند.

ب) بلی، به‌دلیل آن‌که انحصار متقابل (mutual Exclusion) می‌تواند وجود نداشته باشد.

ج) خیر، چون منابع مشترک توسط فرآیند استفاده نمی‌شود.

د) بلی، در صورتی که سیکل انتظار بین چند فرآیند وجود داشته باشد.

۱۰ ـ یک سیستم کامپیوتری دارای ۶ عدد Tape Drive است که n پردازه برای دست‌یابی به آن‌ها رقابت می‌کنند هر پردازه بـه ۲ درایو نیاز دارد. این سیستم به ازای حداکثر چه ارزش‌هایی از n فاقد بن‌بست است؟

الف) n<۲ ب) n<=۳ ج) n<۶ د) n<=۶

۱۱ ـ در سیستمی با ۵ پردازه (Process) وسه نوع منبع (resource)وضعیت تخصیص منابع به شکل زیر است:

	تخصیص یافته			حداکثرمورد نیاز			موجودی اولیه		
	A	B	C	A	B	C	A	B	C
P0	0	1	2	3	6	8	8	6	10
P1	2	0	3	7	3	6			
P2	3	2	0	5	3	3			
P3	1	0	2	4	5	9			
P4	1	1	0	2	3	3			

اگر در این وضعیت درخواستی برای یک واحد دیگر از منبع A توسط پردازه P3 صادر شود:

الف) پس از انجام درخواست فوق وقوع بن‌بست قطعی است.

ب ـ پس از انجام درخواست فوق احتمال وقوع بن‌بست وجود دارد.

ج) قبل از انجام درخواست فوق احتمال وقوع بن‌بست وجود دارد.

د) قبل از انجام درخواست فوق وقوع بن‌بست قطعی است.

۱۲ ـ یک کامپیوتر دارای ۶ دستگاه نوارخوان است و n فرآیند برای استفاده از آن‌ها رقابت می کنند. هر فرآیند حـداکثر بـه ۳ دستگاه نوارخوان نیاز دارد. برای چه مقادیری از n سیستم در حالت امن قرار دارد؟

الف) n>۲ ب) n<=۶

ج) n<=۵ د) امن بودن سیستم به مقدار n بستگی ندارد.

۱۳ ـ سیستمی با ۵پردازش P0 تا P4 و سه نوع منبع A و B و C را در نظرمی‌گیریم. منبع A دارای ۷ نمونه و منبع نوع B دارای ۲ نمونه و منبع نوع C دارای ۶ نمونه است. فرض کنید که در زمان t0 حالت تخصیص منبع زیر را داریم:

Allocation			Request			Avalable		
A	B	C	A	B	C	A	B	C
0	1	0	0	0	0	0	0	0
2	0	0	2	0	2			
3	0	3	0	0	0			
2	1	1	1	0	0			
0	0	2	0	0	2			

کدام‌یک از گزینه‌های زیر صحیح می‌باشد؟

الف) سیستم در حالت بن‌بست می‌باشد.

ب) چنان‌چه پردازش P2 یک درخواست اضافی برای یک نمونه نوع C صادر نماید، پردازش‌های P0 تا P4 در بن‌بست خواهند بود.

ج) چنان‌چه پردازش P2 یک درخواست اضافی برای یک نمونه نوع C صادر نماید، پردازش‌های P1 تا P4 در بن‌بست خواهند بود.

د) سیستم در حالت بن‌بست نمی‌باشد و چنان‌چه پردازش P2 یک درخواست اضافی برای یک نمونه نوع C صـادر نمایـد بـاز هـم بن‌بست اتفاق نمی افتد.

۱۴ ـ در مورد بن‌بست کدام یک از عبارات زیر صحیح است؟

الف) SPOOLING مشکل بن‌بست را در مورد منابع انحصاری از بین می برد.

ب) اگر هر فرآیند دارای پردازنده خاص خود باشد مشکل بن‌بست پیش نخواهد آمد.

ج) از طریق استفاده از الگوریتم‌های مستقل از بن‌بست برای منابع مختلف می‌توان از بروز بن‌بست در سیستم جلوگیری کرد.

د) مشکل بن‌بست با هیچ‌کدام از موارد فوق برنامه طرف نمی‌گردد.

۱۵ ـ کدام‌یک از گزینه‌ها یکی از ۴ شرط لازم برای وقوع بن‌بست نیست؟

الف) دوبدو ناسازگار　　ب) انتظار دورانی

ج) تخصیص ناقص　　د) Non Preemptive Scheduling of Processes

۱۶ ـ سیستمی دارای 4 فرآیند P_1 تا P_4 و 3 منبع R_1 (3 واحد) و R_2 (دو واحد) و R_3 (2 واحد) است. فرآیند P_1 یک واحد R_1 را در اختیار دارد و تقاضای یک واحد R_2 دارد. فرآیند P_2، 2 واحد R_2 در اختیار دارد و تقاضای یک واحد R_1 و یک واحد R_3 دارد. فرآیند P_3 یک واحد R_1 در اختیار دارد و تقاضای یک واحد R_2 دارد. فرآیند P_4، دو واحد R_3 را در اختیار دارد و تقاضای یک واحد R_1 دارد. کدام عبارت در مورد این سیستم اشتباه است؟

الف) سیستم در حالت امن است.

ب) تخصیص یک واحد از R_1 به فرآیند P_4 باعث می‌شود که همه فرآیندها منابع درخواستی شان را دریافت کنند و خاتمه یابند.

ج) تخصیص یک واحد از R_1 به فرآیند P_2 باعث بن‌بست می‌شود.

د) سیستم در حالت بن‌بست است.

۱۷ ـ برای وقوع بن‌بست باید هم‌زمان 4 شرط اتفاق افتاده باشد، کدام گزینه مشخص کننده این چهار شرط می‌باشد؟

الف) Partial Allocation, Circular wait, Preemption, Mutualexclusion

ب) Holdand wait, Circular wait, Nonpreemption, Mutual exclusion

ج) Total allocation, Circuler wait, Non Preemption, Mutual exclusion

د) Partial Allocation, Circular wait, Nonpreemption Nonmutualexclusion

۱۸ ـ برای پیش‌گیری از بن‌بست از طریق شرط انتظار چرخشی، روشی مبتنی بر شماره‌گذاری همه منابع و الزام فرآیندها به درخواست منابع به ترتیب عددی (صعودی)، پیشنهاد شده است. در پیاده‌سازی آن با چه مشکلی روبرو خواهیم شد؟

الف) منابع سیستم را هدر می‌دهد.

ب) نیاز به پیش‌بینی اینده دارد.

ج) همه منابع قابل Spool نیست و خودفضای Spool بر روی دیسک نیز موجب بن‌بست می‌شود.

د) هر دو مورد اول و دوم صحیح است.

۱۹ ـ سیستمی شامل 5 فرآیند همروند و 2 منبع یکسان استفاده مجدد، دارای انحصار متقابل و Preemptive را در نظر بگیرید. به شرط آن‌که دو فرآیند حداکثر به 2 منبع نیاز داشته باشد، تعداد وضعیت‌های بن‌بست در این سیستم به خاطر منابع مذکور حداکثر چند حالت می‌باشد؟

الف) 20 حالت　　ب) 10 حالت　　ج) 5 حالت　　د) صفر حالت

۲۰ـ تحت سیستم عاملی، چهار فرآیند فعال و دو منبع مدیریت می‌شوند. وضعیت سیستم تحت جدول ذیل بیان شده است. حالت سیستم چیست؟

فرآیند	مقادیر اختصاص داده شده		حداکثر نیاز		منابع در دسترس	
	R_1	R_2	R_1	R_2	R_1	R_2
P_1	7	2	9	5	2	1
P_2	1	3	2	6		
P_3	1	1	2	2		
P_4	3	0	5	0		

الف) امن

ب) ناامن

ج) به طور قاطع نمی‌توان پیش‌بینی کرد.

د) بستگی دارد چه فرآیندی چه فرآیندی چه منبعی را تقاضای کند.

۲۱ـ سیستمی شامل 5 فرآیند P_1 تا P_5 و منابع R_1, R_2 در حالت زیر مفروض می‌باشد. حداقل مقدار x برای این که سیستم امن باشد کدام است؟

فرآیند	منابع اختصاص یافته		حداکثر منابع مورد نیاز	
	R_1	R_2	R_1	R_2
P_1	0	2	5	2
P_2	2	5	2	10
P_3	4	0	4	5
P_4	1	1	1	4
P_5	0	0	9	5

منابع فعلی = $(x \quad 3)$

۲۲ـ سیستمی دارای 5 فرآیند و چهار منبع در حالت زیر به‌سر می‌برد.

تعداد کل منابع اولیه

R_0	R_1	R_2	R_3
6	3	4	2

منابعی که هنوز موردنیاز است

	R_0	R_1	R_2	R_3
P_0	1	1	0	0
P_1	0	1	1	2
P_2	3	1	0	0
P_3	0	0	1	0
P_4	2	1	1	0

منابع تخصیص یافته

	R_0	R_1	R_2	R_3
P_0	3	0	1	1
P_1	0	1	0	0
P_2	1	1	1	0
P_3	1	1	0	1
P_4	0	0	0	0

در چه صورتی احتمالی وقوع بن‌بست وجود دارد؟

الف) فرآیند P_1 یک واحد از منبع R_2 را درخواست نماید.

ب) فرآیند P_1 یک واحد از منبع R_2 را درخواست نماید و فرآیند P_4 آخرین واحد R_2 را درخواست نماید.

ج) فرآیند P_1 یک واحد از منبع R_2 را درخواست نماید و فرآیند P_3 آخرین واحد R_2 را درخواست نماید.

د) فرآیند P_2 یک واحد از منبع R_2 را درخواست نماید و فرآیند P_4 کلیه منابع مورد نیازش را درخواست نماید.

۲۳ ـ در یک مدیریت حافظه اصلی به شکل صفحه‌بندی بدون حافظه مجازی همه فضای موردنیاز فرآیندها در ابتدا اختصاص نمی‌یابد، اما وقتی صفحه‌ای به حافظه اصلی منتقل شد تا انتهای کار فرآیند مربوط، در حافظه باقی می‌ماند. هر فرآیند در نهایت تمام صفحاتش به حافظه اصلی منتقل می‌شود. حافظه اصلی دارای 1200 قاب صفحه است. جدول داده شده فرآیند P_1, P_2, P_3 و حداکثر فضای موردنیاز هر یک و تعداد قاب صفحه واگذار شده به هر یک را در حال حاضر نشان می‌دهد. اگر فرآیند 4 وارد شود و کل فضای موردنیاز آن 600 قاب صفحه باشد، حداکثر چند قاب صفحه در ابتدا می‌توان به آن اختصاص داد که سیستم Safe باشد.

فرآیند	حداکثر فضای موردنیاز	فضای واگذار شده
P_1	950	600
P_2	200	100
P_3	550	250

الف) 50

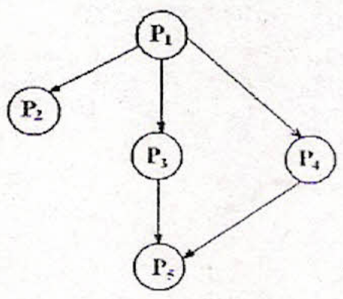
ب) 150

ج) سیستم در وضعیت فعلی امن نیست.

د) در مورد حافظه اصلی صفحه‌بندی شده حالت Holdand wait پیش نمی‌آید، لذا سیستم همیشه امن است.

۲۴ ـ برای 5 فرآیند که گراف تقدم ـ تاخر اجرای آن‌ها در شکل آمده است و همه برای اجرا نیاز به یک منبع واحد دارند. اگر حداکثر نیاز هم‌زمان هر یک طبق جدول مشخص شده باشد. حداقل چند نسخه از این منبع نیاز است که احتمال بن‌بست صفر باشد؟

P_1	P_2	P_3	P_4	P_5
3	3	2	2	8



الف) 14

ب) 10

ج) 9

د) 8

۲۵ ـ سیستمی را در نظر بگیرید که در آن 5 فرآیند و 4 نوع مختلف از منابع وجود دارند. تقاضای تخصیص منابع برای فرآیندها مطابق جدول زیر است. موجودی فعلی منابع (1 5 2 0) می‌باشد. فرض کنید فرآیند P_2 تقاضایی به‌صورت (0 x 0 0) را مطرح می‌نماید، مقدار ماکزیمم X چه می‌تواند باشد به‌طوری که پس از تخصیص X به P_2 سیستم امن باشد.

حداکثر منابع موردنیاز

	A	B	C	D
P_1	0	1	1	2
P_2	1	6	5	0
P_3	2	2	5	6
P_4	0	3	5	3
P_5	0	5	5	6

مناب تخصیص یافته

	A	B	C	D
P_1	0	0	1	2
P_2	0	0	5	0
P_3	1	1	5	4
P_4	0	1	2	0
P_5	0	2	4	2

الف) x=6

ب) x=5

ج) x=3

د) x=4

۲۶ ـ در یک سیستم با وجود 4 پردازه و 4 منبع با جداول تخصیص و درخواست زیر شرایط سیستم چگونه می‌باشد؟

$$All = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad req = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

الف) در این سیستم یک حلقه و بن‌بست وجود دارد.

ب) در این سیستم دو حلقه و بن‌بست وجود دارد.

ج) در این سیستم یک حلقه و بدون بن‌بست است.

۴) در این سیستم حلقه وجود ندارد.

جواب‌های تشریحی تست‌های مربوط به مبحث بن‌بست‌ها

۱ ـ گزینه سوم صحیح است.

۲ ـ گزینه سوم صحیح است.

۳ ـ چنان‌چه n فرآیند از m منبع به طور اشتراکی استفاده نمایند، به طوری که حداکثر نیاز هر فرآیند به منابع از m تجاوز نکند و کل نیاز تمام فرآیندها نیز کمتر از m+n باشد آن‌گاه سیستم هیچ گاه دچار بن‌بست نمی‌گردد.

گزینه سوم صحیح است.

۴ ـ با توجه به الگوریتمی که در این سیستم برای توزیع نوار گردان‌ها بین فرآیندها به‌کار می‌رود، بیش‌تر از ۵ فرآیند به صورت هم‌زمان نمی‌توانند فعالیت نمایند، زیرا هر فرآیند که وارد سیستم می‌شود ۴ نوارگردان به آن اختصاص داده می‌شود، کلاً ۲۰ نوار گردان موجود است پس صفر نوار گردان باقی می‌ماند و فرآیند دیگری نمی‌تواند فعال شود. از طرفی هر فرآیند در شروع کار ۳ نوار گردان را به‌کار می‌گیرد و نوار گردان چهارم ممکن است بیکار بماند و طبق صورت مساله معمولاً نوار گردان چهارم در انتهای کار فرآیند به کار گرفته می‌شود بنابراین در بدترین حالت هر ۵ فرآیند ۳ نوارگردان از ۴ نوارگردان را استفاده می نمایند و کلاً ۵ نوارگردان بیکار می‌ماند و در بهترین حالت تمام ۵ فرآیند تمام نوارگردان‌های اختصاص داده شده را به کار گیرند، آن‌گاه نوارگردان بیکار می‌ماند.

گزینه چهارم صحیح است.

۵ ـ ابتدا موجودی فعلی را محاسبه می نماییم. با توجه به ماتریس تخصیص منابع جمع اعداد مربوط به این ماتریس در هر ستون، کل منابع اختصاص یافته در همان ستون را که مشخص کننده نوع منبع است نشان می‌دهد. بنابراین کل منابع اختصاص یافته بشکل ذیل می‌شود.

$$(5\ 3\ 2\ 2)$$

اگر این بردار را از تعداد کل منابع اولیه کسر نماییم. موجودی فعلی منابع به‌دست می‌آید.

$$AV_0=(1\ 0\ 2\ 0)$$

احتمال وقوع بن‌بست وقتی وجود دارد که با در نظر گرفتن درخواست‌ها در گزینه‌های مساله، سیستم در حالت ناامن قرار گیرد. بنابراین به ترتیب گزینه‌ها را بررسی می نماییم. آن گزینه ای که سیستم را ناامن می‌سازد، گزینه مورد نظر خواهد بود.

در گزینه اول، درخواستی از فرآیند P0 مبنی بر اخذ یک واحد از منبع R2 صادر می‌شود. با توجه به ماتریس Need، چنین درخواستی نباید صادر شود، زیرا نیاز آتی فرآیند P0 از منبع R2، در ماتریس مربوطه صفر است.

در گزینه دوم، درخواستی از فرآیند P1 مبنی بر اخذ یک واحد منبع R2 صادر می‌گردد، در این حالت ماتریس(نیاز آتی) Need، All (تخصیص)و بردار Av (موجودی) را به‌دست می آوریم.

ابتدا وانمود می‌کنیم که یک واحد منبع R2 به فرآیند P1 اعطا شود، آن‌گاه

$$All=\begin{bmatrix}3&0&1&1\\0&1&1&0\\1&1&1&0\\1&1&0&1\\0&0&0&0\end{bmatrix}\quad Need=\begin{bmatrix}1&1&0&0\\0&1&0&2\\3&1&0&0\\0&0&1&0\\2&1&1&0\end{bmatrix}$$

$AV=(1\ 0\ 1\ 0)$

بر مبنای الگوریتم امن و ناامن موجودی را در برابر ماتریس Need قرار می‌دهیم

سطری از ماتریس Need را که در آن شرط AV>=Need برقرار است، حذف می‌نماییم. بنابراین فرآیند P3 حذف می‌گردد و موجودی به $AV_1=(2\ 1\ 1\ 1)$ تغییر می نماید.

در قدم بعدی فرآیند P4 حذف می‌شود و موجودی $AV_2=(2\ 1\ 1\ 1)$ بدون تغیرباقی می‌ماند. سپس فرآیند P0 حذف می‌شود و موجودی را به $AV_3=(5\ 1\ 2\ 2)$ افزایش می‌دهد و بالاخره P1 و در انتهای P2 حذف می‌گردد، بنابراین سیستم امن خواهد بود پس فرآیند P1 یک واحد منبع R2 را دریافت می‌کند، حال باید بررسی کنیم که چنان‌چه فرآیند P3 آخرین واحد باقیمانده R2 را درخواست کند سیستم در چه وضعیتی قرار می‌گیرد. مجدداً الگوریتم امن و ناامن بررسی می‌شود.

فرض می‌کنیم که یک واحد منبع R2 به فرآیند P3 داده شود، آن‌گاه وضعیت سیستم به شکل ذیل خواهد شد.

$$All=\begin{bmatrix}3&0&1&1\\0&1&1&0\\1&1&1&0\\1&1&1&1\\0&0&0&0\end{bmatrix}\quad Need=\begin{bmatrix}1&1&0&0\\0&1&0&2\\3&1&0&0\\0&0&0&0\\2&1&1&0\end{bmatrix}$$

$AV=(1\ 0\ 0\ 0)$

با توجه به مقادیر بردار موجودی و ماتریس Need، کلیه فرآیندها به ترتیب P3، P0، P4، P2 و P1 حذف می‌شوند و سیستم در حالت امن قرار می‌گیرد.

گزینه سوم را بررسی می نماییم. بخش اول آن (یعنی فرآیند P1 یک واحد از منبع R2 را تقاضا می‌کند) پاسخ داده می‌شود، زیرا مشاهده کردیم که سیستم امن باقی می‌ماند. بخش دوم را که فرآیند P4 یک واحد از منبع R2 را درخواست می‌نماید، باید بررسی گردد.

مانند وضعیت قبلی مجدداً الگوریتم امن و ناامن را با فرض $AV=(1\ 0\ 1\ 0)$ و با وانمود کردن این مطلب که یک واحد از منبع R2 به فرآیند P4 اختصاص داده شود، اجرا می نماییم.

$$All=\begin{bmatrix}3&0&1&1\\0&1&1&0\\1&1&1&0\\1&1&0&1\\0&0&1&0\end{bmatrix}\quad Need=\begin{bmatrix}1&1&0&0\\0&1&0&2\\3&1&0&0\\0&0&1&0\\2&1&0&0\end{bmatrix}$$

$AV=(1\ 0\ 0\ 0)$

ملاحظه می‌کنید که موجودی فعلی جوابگوی نیازهای آتی فرآیندها نمی‌باشد، زیرا هیچ یک از سطرهای ماتریس Need کوچک‌تر از بردار AV نمی‌باشد، بنابراین سیستم در حالت ناامن قرار می‌گیرد و احتمال بن‌بست وجود دارد.

گزینه سوم صحیح است.

۶ـ گزینه دوم و سوم هر دو غلط است، زیرا داشتن یک پردازنده و یا چند پردازنده و سیستم تک کاربره و یا چند کاربره ربطی به مساله بن‌بست فرآیندها ندارد.

چنان‌چه در سیستم، چند فرآیند به طور همروند فعال باشند و یکی از فرآیندها به حالت Wait برود، سیستم عامل نمی‌تواند تشخیص دهد که آیا Signal برای خروج این فرآیند خواهد آمد یا خیر! و ضمناً با کنترل زمان سنج این تشخیص غیرممکن است، زیرا زمان را نمی‌توان تعیین نمود.

گزینه اول صحیح است.

۷ـ از فرض مساله، ماتریس تخصیص منابع(All) و درخواست منابع (Req) و بردار موجودی فعلی AV به شکل ذیل به‌دست می‌آید.

$$All = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \qquad Request = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

موجودی فعلی=(۱ ۰ ۱) ⇐ بعد از اختصاص منابع (3 2 3)=AV موجودی اولیه

موجودی فعلی را در مقابل ماتریس Req قرار دهید، مشاهده می‌شود که تمام فرآیندها جواب داده می‌شوند، زیرا ابتدا به فرآیند P2 جواب می‌دهیم. بعد از مدتی این فرآیند کارش به اتمام می‌رسد، منابع دریافتی را بر می‌گرداند. موجودی به (1 2 1)=AV تبدیل می‌شود.

سپس سیستم به فرآیند P1 پاسخ می‌دهد موجودی به (2 2 1)=AV تبدیل می‌گردد. متعاقباً فرآیندهای P3 و P4 منابع مورد درخواستشان را دریافت می‌نمایند. پس ملاحظه می‌کنید که هیچ فرآیندی در بن‌بست قرار نمی‌گیرند.

گزینه چهارم صحیح است.

۸ـ چنان‌چه در سیستمی n فرآیند، m منبع را به طور اشتراکی استفاده نمایند و گرفتن و آزاد کردن منابع یک به یک صورت پذیرد و جمع کل درخواست‌ها کمتر از m+n و حداکثر منابع مورد نیاز هر فرآیند m باشد، آن‌گاه این سیستم همیشه عاری از بن‌بست است.

گزینه سوم صحیح می‌باشد.

۹ـ در این نوع سیستم‌ها، فرآیندها به کمک دستور Send (به معنی ارسال نمودن پیغامی از طرف یک فرآیند به مقصد مشخصی از فرآیندها)، دستور Receive (به‌معنی دریافت پیغام از طرف فرآیند مشخص) با یکدیگر ارتباط برقرار می نمایند. معمولاً دستور Receive فرآیند فراخواننده را به شرط عدم وجود پیغام در انتظار نگه می‌دارد(چنان‌چه پیغامی نرسد در انتظار همیشگی باقی خواهد ماند). فرض کنید فرآیندها دستور Receive را صادر نمایند و برای آن‌ها پیغامی فرستاده نشود و یا پیغام فرستاده شده به علتی گم شود آن‌گاه امکان این وجود دارد که فرآیندها در بن‌بست قرار گیرند.

گزینه اول حالتی از ارتباط بین دو فرآیند می‌باشد که با هم قرار ملاقات دارند، آن‌گاه در مقابل هر Send طرف مقابل دستور Receive را صادر می نمایند و زمان ارسال و دریافت مشخص است.

گزینه چهارم صحیح است.

۱۰ ـ به جواب سوال ۸ بن‌بست مراجعه نمایید.

تعداد پردازش ها + کل منابع موجود > جمع کل درخواست‌ها

$2n< m+n \Rightarrow n< m \Rightarrow n<6$

گزینه سوم صحیح است.

۱۱ ـ ماتریس نیاز آتی را به‌دست می آوریم:

$$\text{All} = \begin{array}{c} \\ P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$\text{Max} = \begin{bmatrix} 3 & 6 & 8 \\ 7 & 3 & 6 \\ 5 & 3 & 3 \\ 4 & 5 & 9 \\ 2 & 3 & 3 \end{bmatrix}$$

$$\text{Need} = \begin{bmatrix} 3 & 5 & 6 \\ 5 & 3 & 3 \\ 2 & 1 & 3 \\ 3 & 5 & 7 \\ 1 & 2 & 3 \end{bmatrix}$$

مقادیر اولیه $AV_0=(8\ \ 6\ \ 10)$

مقادیر فعلی $AV_1=(1\ \ 2\ \ 3)$

امن بودن وضعیت فعلی سیستم به شکل ذیل بررسی می‌شود:

موجودی فعلی نشان می‌دهد که پردازش P_4 می‌تواند به اتمام برسد با اتمام این پردازش منابع $(0\ \ 1\ \ 1)$ آزاد می‌شود و موجودی تبدیل به $AV_2=(2\ \ 3\ \ 3)$ خواهد شد. متعاقباً پردازش P_2 با موجودی به‌دست آمده پاسخ داده خواهد شد و منابع $(3\ \ 2\ \ 0)$ که در اختیار این پردازش بوده است به سیستم بر می‌گردد، بنابراین موجودی سیستم تبدیل به $AV_3=(5\ \ 5\ \ 3)$ می‌شود، به همین ترتیب پردازش‌های P_1 و P_0 ونهایتا P_3 از سیستم حذف می گردند بنابراین به این نتیجه می رسیم که وضعیت فعلی امن است.

چنان‌چه پردازش P_3 تقاضای یک واحد از منبع A را نماید بدین معنی که

$Req_{P3}=(1\ \ 0\ \ 0)$

آن‌گاه مجدداً الگوریتم امن و ناامن را با وانمود کردن این‌که به تقاضای P_3 جواب مثبت دهیم، مورد بررسی قرار می‌دهیم.

ماتریس All و Need را با فرض آن‌که به درخواست P_3 جواب دهیم به‌دست می‌آوریم:

$$\text{All} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 3 & 2 & 0 \\ 2 & 0 & 2 \\ 1 & 1 & 0 \end{bmatrix} \qquad \text{Need} = \begin{bmatrix} 3 & 5 & 6 \\ 5 & 3 & 3 \\ 2 & 1 & 3 \\ 2 & 5 & 7 \\ 1 & 2 & 3 \end{bmatrix}$$

موجودی سیستم بعد از پاسخ به درخواست پردازش P_3، تبدیل به $AV=(0\ \ 2\ \ 3)$ خواهد شد این بردار در مقابل Need به‌دست آمده قرار می‌گیرد، همین طور که ملاحظه می‌کنید هیچ یک از سطرهای ماتریس Need را نمی‌توان حذف کرد، بنابراین به این نتیجه می رسیم که اگر بخواهیم به درخواست P_3 جواب مثبت دهیم سیستم در حالت ناامن قرار می‌گیرد، بدین معنی که احتمال وقوع بن‌بست وجود خواهد داشت.

گزینه دوم صحیح است.

۱۲ ـ حالت امن بودن سیستم به تعداد پردازش‌ها ربطی پیدا نمی‌کند، بلکه بستگی به تعداد منابع موجود فعلی (بردار موجودی منابع) و نیازهای آتی پردازش‌ها از منابع (ماتریس نیازهای آتی) دارد.

گزینه چهارم صحیح است.

۱۳ ـ در وضعیت فعلی هیچ یک از پردازش‌های سیستم در حالت بن‌بست نمی‌باشند، زیرا با توجه به ماتریس تخصیص منابع و درخواست‌ها و بردار موجودی فعلی دو پردازش P0 و P2 که درخواستی ندارند کارشان می‌تواند تمام شود و موجودی به (3 1 3) تغییر نماید منابع موجود به‌دست آمده بقیه پردازش ها را جوابگو است، پس سیستم در حالت بن‌بست نمی‌باشد.

چنان‌چه P2 یک درخواست اضافی از منبع نوع C تقاضا نماید آن‌گاه ماتریس درخواست‌ها به:

$$AV=(0\ 0\ 0) \qquad Req=\begin{bmatrix}0&0&0\\2&0&2\\0&0&1\\1&0&0\\0&0&2\end{bmatrix}$$

تغییر می‌نماید. هم اکنون پردازش P0 می‌تواند کارش تمام شود و یک واحد منبع نوع B را به سیستم برگرداند. بنابراین موجودی (0 1 0) خواهد شد متاسفانه بقیه پردازش‌ها P1، P2، P3 و P4 هیچ کدام از تقاضاهایشان جواب داده نخواهد شد و بن‌بست ایجاد می‌شود.

گزینه سوم صحیح است.

۱۴ ـ گزینه چهارم صحیح می‌باشد.

۱۵ ـ گزینه چهارم صحیح می‌باشد.

۱۶ ـ گزینه چهارم صحیح می‌باشد.

با تشکیل ماتریس‌های ذیل می‌توان وضعیت را بررسی نمود.

$$\text{ماتریس منابع تخصیص یافته} = \begin{array}{c}\begin{array}{ccc}R_1 & R_2 & R_3\end{array}\\ \begin{pmatrix}1&0&0\\0&2&0\\1&0&0\\0&0&2\end{pmatrix}\end{array}$$

$$\text{ماتریس درخواست} = \begin{array}{c}\begin{array}{ccc}R_1 & R_2 & R_3\end{array}\\ \begin{pmatrix}0&1&0\\1&0&1\\0&1&0\\1&0&0\end{pmatrix}\end{array}$$

$$\text{منابع فعلی} = (1\quad 0\quad 0)$$

منابع فعلی را در مقابل ماتریس درخواست قرار می‌دهیم، ابتدا فرآیند P_4 حذف می‌شود.

$(1\quad 0\quad 2)$ = منبع فعلی بعد از حذف P_4، سپس فرآیند P_2 حذف می‌شود و منابع خود را آزاد می‌نماید.

$(1\quad 2\quad 2)$ = منبع فعلی بعد از حذف P_2 نهایتاً دو سطر مربوط به فرآیند اول و سوم هم حذف می‌گردند، بنابراین سیستم در حالت بن‌بست نمی‌باشد.

۱۷ ـ گزینه دوم صحیح می‌باشد.

گزینه اول به‌خاطر Preemption غلط می‌باشد.

گزینه سوم به‌خاطر Total allocation غلط می‌باشد (Hold and wait را نقض می‌نماید)

گزینه چهارم به‌خاطر Non Mutual exclusion غلط می‌باشد.

Partial allocation همان Hold and wait است.

۱۸ ـ گزینه چهارم صحیح می‌باشد.

۱۹ ـ گزینه چهارم صحیح می‌باشد.

به دلیل این‌که منابع خاصیت Preemptive دارند، هیچ‌گاه بن‌بست رخ نمی‌دهد.

۲۰ ـ گزینه دوم صحیح می‌باشد.

$$MAX-ALL=Need=\begin{pmatrix}2 & 3\\ 1 & 3\\ 1 & 1\\ 2 & 0\end{pmatrix}$$

ماتریس Need در مقابل منابع قرار می‌گیرد. $(1 \quad 2)$= منابع در دسترس

سطر سوم و چهارم مربوط به فرآیندهای P_4, P_3 از ماتریس Need حذف می‌شود، اما دو سطر اول و دوم نمی‌تواند حذف گردند.

بنابراین توالی امن یافت نمی‌شود، پس سیستم ناامن است.

۲۱ ـ ماتریس Need را محاسبه نمایید.

$$MAX-ALLOC=Need=\begin{pmatrix}5 & 0\\ 0 & 5\\ 0 & 5\\ 0 & 3\\ 9 & 5\end{pmatrix}$$

$(x \quad 3)$= منابع فعلی

ماتریس Need را در مقابل منابع فعلی قرار می‌دهیم.

فقط سطر چهارم حذف می‌شود منبع فعلی $=(x+1 \quad 4)$ خواهد شد، سپس سطر اول اگر بخواهد حذف شود. می‌بایست $x+1\geq 5$ باشد، پس حداقل $x=4$ می‌شود و بقیه سطرها هم حذف خواهد شد.

۲۲ ـ یک به یک گزینه‌ها را بررسی می‌کنیم تا به گزینه‌ای برخورد کنیم که باعث ناامن شدن سیستم گردد. گزینه اول صحیح نمی‌باشد زیرا فرآیند P_1 یک واحد از منبعی را درخواست کرده که با دریافت آن سیستم در حالت امن باقی می‌ماند، چنان‌چه منبع به فرآیند P_1 تخصیص داده شود ماتریس Need به‌شکل ذیل خواهد شد.

$$Need=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 0 & 2\\ 3 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 2 & 1 & 1 & 0\end{bmatrix} \quad Av=(1 \quad 0 \quad 1 \quad 0)$$

و یک توالی امن $\langle P_3, P_4, P_0, P_2, P_1 \rangle$ خواهد بود بنابراین سیستم امن است.

گزینه دوم صحیح می‌باشد، زیرا با درخواست‌های دو فرآیند P_4, P_1 از منبع R_2 و دریافت آن سیستم ناامن می‌گردد. ماتریس Need و موجودی فعلی را بعد از تخصیص منبع R_2 به فرآیندهای P_4, P_1 به شکل ذیل می‌نویسیم.

$$\text{Need}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix} \qquad Av=(1\ \ 0\ \ 0\ \ 0)$$

با موجودی فعلی سطرهای ماتریس Need حذف نمی‌شود و نمی‌توان توالی امن به‌دست آورد.

گزینه سوم صحیح نمی‌باشد چون فرآیند P_2 نمی‌تواند از منبع R_2 درخواستی داشته باشد، زیرا در ماتریس Need نیاز فرآیند P_2 از منبع R_2 صفر می‌باشد.

گزینه چهارم صحیح نمی‌باشد زیرا منابعی که فرآیند P_4 درخواست می‌نماید بیشتر از موجودی فعلی است و سیستم امن است.

۲۳ـ نیاز آتی فرآیندهای P_3, P_2, P_1 را به شکل ذیل محاسبه می‌نماییم.

$$\text{Need} = \text{MAX} - \text{ALL} = \begin{bmatrix} 350 \\ 100 \\ 300 \end{bmatrix} \qquad Av = 1200 - 950 = 250$$

حداکثر نیاز فرآیند $P_4 = 600$ قاب صفحه است، فرض می‌کنیم x قاب صفحه در ابتدا به فرآیند P_4 اختصاص دهیم. موجودی اولیه و نیاز آتی را برای 4 فرآیند به شکل ذیل می‌نویسیم.

$$\text{Need} = \begin{bmatrix} 350 \\ 100 \\ 300 \\ 600-x \end{bmatrix} \qquad Av = 250 - x$$

حال سعی می‌کنیم که توالی امن به‌دست آوریم. ابتدا سطر دوم یعنی فرآیند P_2 را حذف می‌نماییم، به شرطی این سطر حذف می‌شود که حداکثر مقدار x برابر 150 باشد فرض کنیم که این چنین باشد. آن‌گاه بعد از حذف سطر دوم موجودی 100 واحد افزایش می‌یابد. بدین‌صورت $Av=200$ خواهد شد متاسفانه سطرهای دیگر Need حذف نمی‌شود، نمی‌توان توالی امن به‌دست آورد. حال چنان‌چه مقدار x برابر 50 باشد آن‌گاه سطر دوم که حذف شود موجودی 300 خواهد بود که سطر سوم هم حذف می‌شود و متعاقباً سطر اول و چهارم هم حذف می‌گردد و سیستم امن خواهد شد.

گزینه اول صحیح است.

۲۴ـ همان‌طور که از شکل مساله مشخص است بعد از اتمام فرآیند P_1 سه فرآیند P_4, P_3, P_2 به‌طور همروند اجرا می‌شوند. این سه فرآیند به 7 منبع نیازمندند از آن‌جایی که می‌دانیم چنان‌چه در سیستمی کل نیاز فرآیندها کمتر از مجموع منابع و تعداد فرآیندها باشد $(R<x+n)$ سیستم عاری از بن‌بست خواهد شد، بنابراین $7<m+3 \Leftarrow m>4$ نتیجه می‌گیریم که در این لحظه حداقل منابع باید 5 باشد تا در این سیستم بن‌بست نداشته باشیم. اما بعد از اتمام دو فرآیند P_4, P_3 فرآیند P_5 وارد سیستم می‌شود و هم اکنون دو فرآیند P_5, P_2 به‌صورت همروند در سیستم اجرا می‌شوند، حال باید تعداد حداقل منابع را که سیستم

عاری از بن‌بست باشد محاسبه نماییم. می‌دانیم که تقاضای دو فرآیند مذکور در کل 11 عدد می‌باشد و می‌بایست $11<2+m$ برقرار باشد نتیجه می‌گیریم که حداقل $m=10$ است تا سیستم عاری از بن‌بست گردد.

گزینه دوم صحیح است.

۲۵ ـ گزینه چهارم صحیح می‌باشد.

قبل از درخواست P_2 ,Need را محاسبه می‌نماییم.

$$\begin{array}{c} \quad A \quad B \quad C \quad D \\ \text{Need}=\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 6 & 0 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 2 & 3 & 3 \\ 0 & 3 & 1 & 4 \end{pmatrix} \end{array}$$

در این حالت سیستم امن می‌باشد.

پردازش P_2 از منبع B حداکثر 6 عدد در آینده درخواست خواهد داشت. هم‌اکنون چون موجودی فعلی برای منبع B، 5 می‌باشد. حداکثر 5 منبع را به‌شرطی که سیستم امن بماند می‌تواند درخواست نمایند، اما چون با تخصیص 5 منبع به پردازش P_2 سیستم ناامن می‌گردد، پس حداکثر 4 منبع را به P_2 اختصاص می‌دهیم که در این‌صورت سیستم امن خواهد ماند، زیرا پس از تخصیص 4 منبع، Need به شکل زیر خواهد آمد.

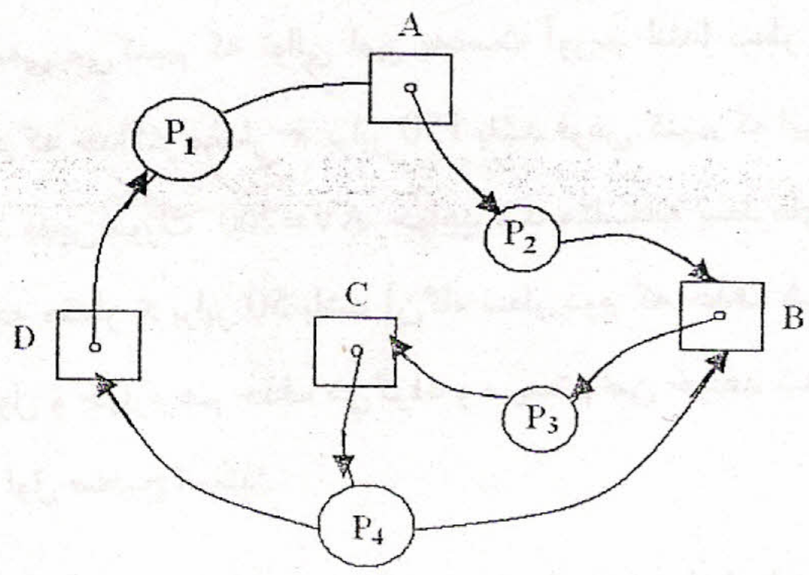
$$\text{Need}=\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 2 & 3 & 3 \\ 0 & 3 & 1 & 4 \end{pmatrix} \qquad \text{ALL}=\begin{pmatrix} 0 & 0 & 1 & 2 \\ 0 & 4 & 5 & 0 \\ 1 & 1 & 5 & 4 \\ 0 & 1 & 2 & 0 \\ 0 & 2 & 4 & 2 \end{pmatrix}$$

۲۶ ـ گزینه دوم صحیح می‌باشد.

گراف تخصیص منابع را به شکل ذیل می‌کشیم.

حلقه اول شامل:

$$P_1 \rightarrow A \rightarrow P_2 \rightarrow B \rightarrow P_3 \rightarrow C \rightarrow P_4 \rightarrow D$$



حلقه دوم شامل:

$$P_3 \rightarrow C \rightarrow P_4 \rightarrow B$$

دو حلقه و بن‌بست وجود دارد.