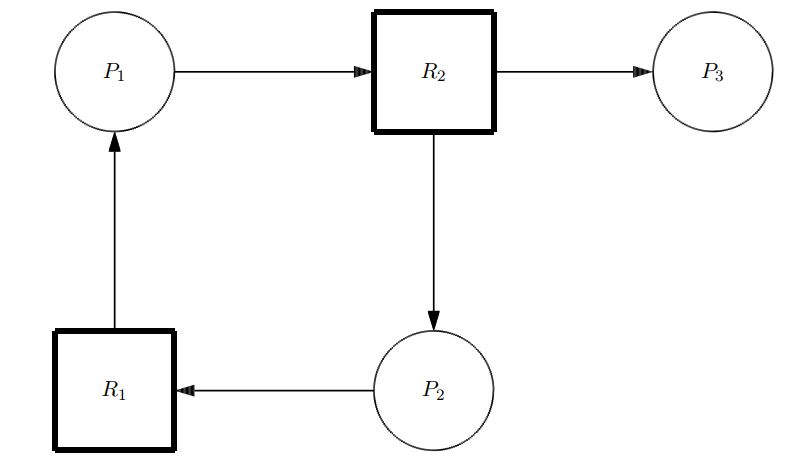
1. **(10 points)**

**The following figure shows a resource graph for a system with consumable resources only. A resource is represented by a rectangle with thick lines and labeled as Ri . A process is represented by a circle, labeled Pi .**

**(a) Is the graph a claim-limited graph? Why?**

**(b) Is the graph reducible? Why?**



Answer:

1. Yes, the graph is a claim limited graph because it represents a consumable resource system and shows that each resource has 0 available units. It also has the relation were a request edge (pi, Rj) if and only if Pi is a consumer of Rj.
2. Yes, the graph is reducible. It is because of P3 which is a producer of R2 and is not blocked. This means that it can produce R2 items to reduce the edge of (P1, R2) and subsequently other edges.
3. **( 10 points )**

**Assume a system has P processes and R identical units of a reusable resource. If each process can claim at most N units of the resource, determine whether each of the following is true or false and prove your claim:**

1. **If the system is deadlock free, then R ≥ P(N − 1) + 1.**
2. **If R ≥ P(N − 1) + 1 then the system is deadlock free.**

Answer:

1. Given that the system is deadlock free, prove that R>= P(N-1) + 1.

Proof:

Assume for the sake of contradiction that R < P(N-1) +1 is true. This means at best we can have R = P(N-1). If each process P holds (N-1) units and requests an additional unit, the system will not have any extra units as it only has R <= P(N-1) units, meaning that it cannot grant any request to the process. Thus, we would result in a deadlock, which leads us to a contradiction. So, the assumption of R < P(N-1) + 1 is wrong and R >= P(N-1) + 1 is correct.

1. Given that R >= P(N-1) + 1 so that the system is deadlock free

Proof:

Since each processor P can claim at most N units, the worst-case scenario is when each process holds N-1 units. Since the system has P(N-1)+1 units, if a process request an additional unit, the system can allocated the available unit to one of the processes to finish its task. Once the process terminates the held resources will be returned to the system. This process can continue for other processes, which keeps the system deadlock free.

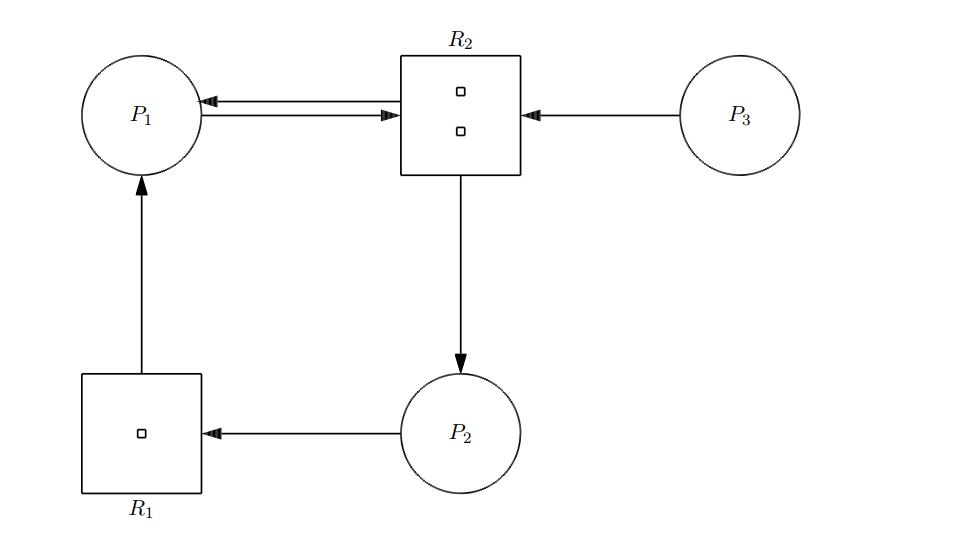
1. **(10 points)**

**The following figure shows a resource graph for a system with reusable resources only. A resource is represented by a rectangle, in which a small square indicates a unit of the resource.**

**(a) Is the graph expedient? Why?**

**(b) Is there any knot in the graph? Why?**

**(c) Is there any deadlock in the system? Why?**



Answer:

1. Yes the graph is expedient. This is because all processes (p1,p2,p3) have outstanding request that are blocked.
2. Yes the subgraph{p1,p2,r1,r2} is a knot. This is because a node in the subgraph is reachable from any other node and no other node in the graph can be read from a node of the subgraph
3. Yes, the system is deadlocked. This is because a knot is sufficient of a condition to conclude deadlock
4. **(10 points)**

**In this problem you are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 15 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming that the data needed are in a cache in main memory. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during which time the thread sleeps. How many requests/sec can the server handle if it is single threaded? If it is multithreaded?**

Answer:

* In a single threaded server:

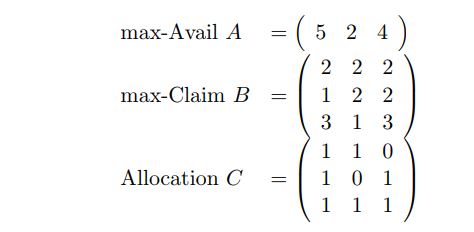
It takes 15mSec to request for work, for the data in cache in main memory. If additional disk operation needed, additional 75mSec required. It forms a total of 75+15=90mSec. Total time required for reading a file using single threaded file server =1/3\*(90) + 2/3\*(15)=40mSec Total number of requests/sec that server can handle in a single threaded server is 1000/40=25requests/sec.

* In a Multi-Threaded server:

The waiting time for the disc is overlapped. So, it takes 15mSec to read a file. Total number of requests/sec that server can handle in a multi-threaded server is 1000/15=66(2/3) requests/sec.

1. **(10 points)**

**Consider the state of a system with processes P1, P2, and P3, defined by the following matrices:**

****

**(a) Find the available matrix D and the need matrix E in this state.**

**(b) Suppose now process P1 makes a request with F1 = ( 0 0 1 ) If the request were granted, what would be D, C, and E in the resulted state?**

**(c) To ensure the system be safe, should the request be granted? Why? Give your reasons in detail.**

1. Available matrix D = A – C

= -

=

Need Matrix E = Max claim – Allocation

= - =

1. Suppose process P1 makes a request with F1 =

D = - =

C = - =

Need Matrix E = Max Claim – Allocation

= - =

1. To ensure the system is safe, should the request be granted? Why? Give your reasons in detail

* The request is granted, if and only if it is a safe state. There are three unsatisfied processes P1, P2, and P3.
* Using the safe-state check algorithm:
* P1 = <= is false
* P2 = <= is false
* P3 = <= is false
* Thus, this does not satisfy the condition E<= D and the process will remain unfinished. Thus, this is not a safe state.