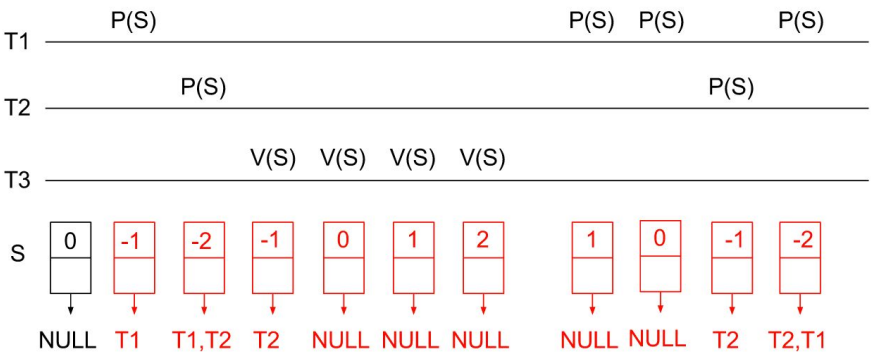
Esercizio 1

Considerare tre processi che si sincronizzano tramite il semaforo S inizializzato a 0.



Indicare nello schema sopra lo stato del semaforo S (valore e coda) in corrispondenza di ogni P e V effettuata dai thread.

Breve spiegazione (indicare se i thread sono bloccati): Il semaforo è rosso e le prime due P sono bloccanti e mettono i thread T1 e T2 in coda. Le successive due V sbloccano prima T1 e poi T2 portando il valore nuovamente a 0. Le successive V portano il valore del semaforo a 2 e le 2 P eseguite da T1 non sono bloccanti, ma riportano il valore a 0. L'ultima P di T2 è bloccante e lo stesso vale per l'ultima P di T1. I due thread vengono messi in coda.

Esercizio 2

Scrivere un Monitor `risorse` che gestisca le richieste di allocazione di N risorse da parte di M thread. I thread richiedono le risorse invocando il metodo `risorse.richiedi(risorsa,thread)` e le rilasciano invocando `risorse.rilascia(risorsa,thread)`. Il monitor deve prevenire lo stallo implementando l'allocazione gerarchica secondo l'ordine naturale delle risorse 0<1<2< … < N-1. Il metodo `richiedi` deve ritornare `false` se è stata violata la gerarchia e `true` altrimenti. Se non ci sono violazioni deve essere bloccante nel caso la risorsa non sia disponibile.

Codice Monitor:

```
Monitor risorse {
  boolean allocata[N]={false, … , false}; // allocazione
  int thread[N]; // a quale thread la risorsa è allocata

boolean richiedi(int r, int t) {
  // controlla se una risorsa >= r è allocata a t
  for(i=r;i<N;i++) {
    if (allocata[i] && thread[i]==t)
      return false; // violazione!
  }
  // attende la risorsa
  while (allocata[r]) wait();

  // assegna la risorsa a t
  allocata[r] = true;
  thread[r] = t;
  return true;
}

void rilascia(int r, int t) {
  // rilascia la risorsa
  // non è necessario modificare thread[t]
  allocata[r] = false;
  notifyAll();
}
```

Breve spiegazione della soluzione proposta: memorizziamo in `thread` l'assegnamento in modo da controllare che quando un thread chiede una risorsa non ne abbia di già assegnate maggiori o uguali a quella che richiede. In tal caso ritorniamo false (violazione dell'allocazione gerarchica). Se non c'è violazione attendiamo che la risorsa sia disponibile e in tal caso la allochiamo. Il metodo `rilascia` rende nuovamente la risorsa disponibile e notifica tutti i thread in attesa.