# Operating System: Active Learning Spring 2022

**Instructor: Jungchan Cho (thinkai@gachon.ac.kr),**

# Deadlock Detection – Several Instance of a Resource Type

The wait-for graph scheme is not applicable to a resource-allocation system with multiple instances of each resource type. We need a deadlock detection algorithm that is applicable to such a system. The algorithm employs several time-varying data structure that are similar to those used in the banker’s algorithm.

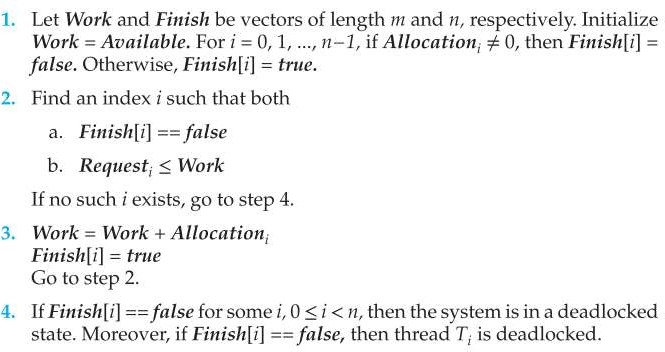
* **Available**. A vector of length 𝑚 indicates the number of available resources of each type.
* **Allocation**: An 𝑛 × 𝑚 matrix defines the number of resources of each type currently allocated to each thread
* **Request**: An 𝑛 × 𝑚 matrix indicates the current request of each thread. If Request[i][ j] equals 𝑘, then thread 𝑇i is requesting 𝑘 more instances of resource type 𝑅j

The ≤ relation between two vectors denotes as follows:

Let 𝑋 and 𝑌 be vectors of length 𝑛. We say that 𝑋 ≤ 𝑌 if and only if 𝑋[𝑖] ≤ 𝑌[𝑖] for all 𝑖 = 1,2, . . , 𝑛. For example, if 𝑋 = (1,7,3,2) and 𝑌 = (0,3,2,1), then 𝑌 ≤ 𝑋. In addition, 𝑌 < 𝑋 if 𝑌 ≤ 𝑋 and 𝑌 ≠ 𝑋.

We can treat each row in the matrices 𝑨𝒍𝒍𝒐𝒄𝒂𝒕𝒊𝒐𝒏 and 𝑹𝒆𝒒𝒖𝒆𝒔𝒕 as vectors and refer to them as 𝑨𝒍𝒍𝒐𝒄𝒂𝒕𝒐𝒊𝒏𝒊 and 𝑹𝒆𝒒𝒖𝒆𝒔𝒕𝒊. The vector 𝑨𝒍𝒍𝒐𝒄𝒂𝒕𝒊𝒐𝒏𝒊 specifies the resources currently allocated to thread 𝑇i ; the vector 𝑹𝒆𝒒𝒖𝒆𝒔𝒕𝒊 specifies the resources requested by thread 𝑇i .

This detection algorithm simply investigates every possible allocation sequence for the threads that remain to be completed.



# [Problem]

Consider a system with five threads 𝑇0 through 𝑇4 and three resource types A, B, and C. Resource type A has seven instances, resource type B has two instances, and resource type C has six instances.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 𝑨𝒍𝒍𝒐𝒄𝒂𝒕𝒊𝒐𝒏 | 𝑹𝒆𝒒𝒖𝒆𝒔𝒕 | 𝑨𝒗𝒂𝒊𝒍𝒂𝒃𝒍𝒆 |
|  | A B C | A B C | A B C |
| 𝑇0 | 0 1 0 | 0 0 0 | 0 0 0 |
| 𝑇1 | 2 0 0 | 2 0 2 | 0 1 0 |
| 𝑇2 | 3 0 3 | 0 0 0 | 3 1 3 |
| 𝑇3 | 2 1 1 | 1 0 0 | 5 2 4 |
| 𝑇4 | 0 0 2 | 0 0 2 | 5 2 6 |

1. Answer whether the system below is in the deadlock state. If there does not exist a deadlock state, solve the problem using the algorithm presented above and write down the process in detail and find the sequence in which the system works without a deadlock. If there exists a deadlock, write down all the threads, consisting of the deadlock.
2. it initialized work = available = {0, 0, 0}, for I = 0, 1, …4, and then finish[i] = false for all i.
3. found an index I such that both finish[i] == false and request i <=work

and it’s 0 Since finish[0] = false and {0,0,0}<={0,0,0}

1. work = {0,0,0} + {0, 1, 0} now work = {0, 1, 0} and finish[0] = true now
2. found an index I such that both finish[i] == false and request I <=work

and it’s 2 since finish[2] = false and {0,0,0}<={0,1,0}

1. work = {0,1,0} + {3,0,3} = {3,1,3} and finish[2] = true now
2. found an index I such that both finish[i] == false and request I <=work

and it’s 1 since finish[3] = false and {1,0,0}<={3,1,3}

1. work = {3,1,3} + {2,1,1} = {5,2,4} and finish[1] = true now
2. found an index I such that both finish[i] == false and request I <=work

and it’s 3 since finish[1] = false and {2,0,0}<={5,2,4} and finish[1] = false

1. work = {5,2,4} + {2,0,0} = {7,2,4} and finish[3] = true now
2. found an index I such that both finish[i] == false and request I <=work and it’s 4 since finish[4] = false and {0,0,2}<={7,2,4} and finish[4] = true now
3. work = {7,2,4} + {0,0,2} = {7,2,6} and finish[i] = true for all i (0<=i<=4). this means the system is not deadlocked

Sequence <T0, T2, T3, T1, T4> is not deadlocked which means Finish[i] = true for all i (0<=i<=4)

1. Suppose now that 𝑇2 makes one additional request for an instance of type C. That is, the 𝑹𝒆𝒒𝒖𝒆𝒔𝒕 matrix is modified as follows:

|  |  |
| --- | --- |
|  | 𝑹𝒆𝒒𝒖𝒆𝒔𝒕 |
|  | A B C |
| 𝑇0 | 0 0 0 |
| 𝑇1 | 2 0 2 |
| 𝑇2 | **0 0 1** |
| 𝑇3 | 1 0 0 |
| 𝑇4 | 0 0 2 |

Answer whether the system below is in the deadlock state. If there does not exist a deadlock state, solve the problem using the algorithm presented above and write down the process in detail and find the sequence in which the system works without a deadlock. If there exists a deadlock, write down all the threads, consisting of the deadlock.

1. it initialized work = available = {0, 0, 0}, for I = 0, 1, …4, and then finish[i] = false for all i.
2. found an index I such that both finish[i] == false and request I <=work

and it’s 0 Since finish[0] = false and {0,0,0}<={0,0,0}

1. work = {0,0,0} + {0, 1, 0} now work = {0, 1, 0} and finish[0] = true.
2. Find an index I such that both finish[i] == false and request i <= work. If no such i exists, go to step 4.
3. If finish[i] == false for some i (0<=i<=4), then the system is in a deadlocked state.

T1, T2, T3, and T4 consists of deadlock