Banker's Algorithm

Session: 2021 – 2025

Submitted by:

Muhammad Yaqoob 2021-CS-118

Supervised by:

Ms. Abqa Javed

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

0.1 Banker's Algorithm

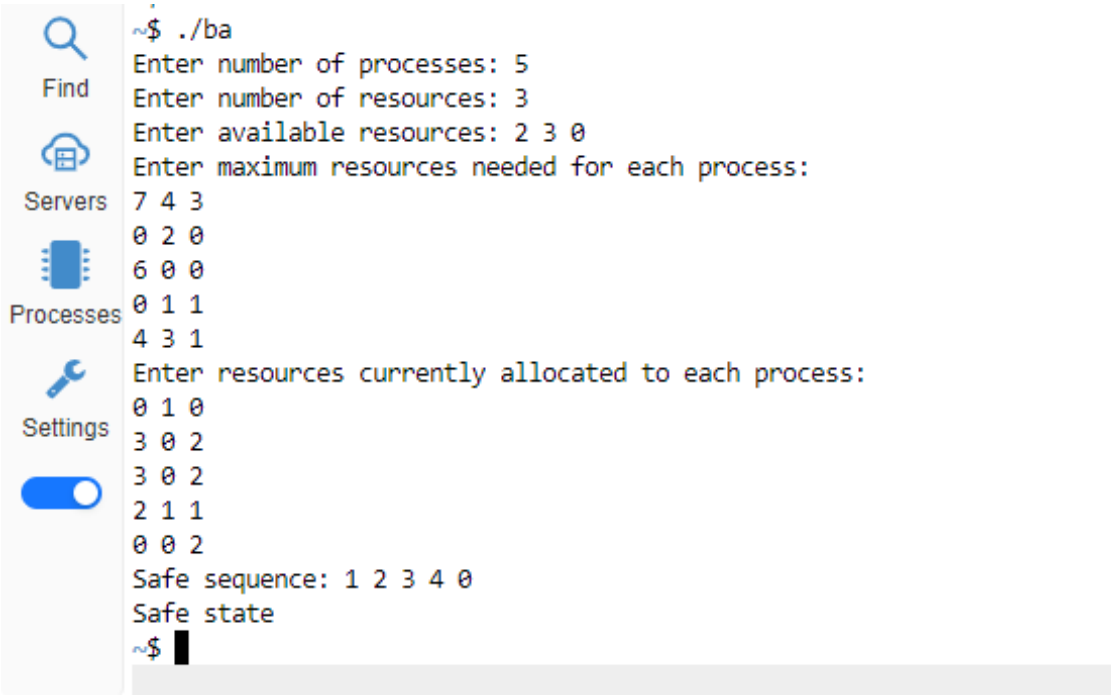

FIGURE 1: Banker's Algorithm to determine the safe state

```
#include <stdio.h>

// Define maximum number of processes and resources
#define MAX_PROCESSES 10
#define MAX_RESOURCES 10

// Function to find the safe state
int bankerAlgorithm(int available[], int max[][MAX_RESOURCES], int allocation[][
    MAX_RESOURCES], int nProcesses, int nResources)
{
    int i, j, k;
    int work[MAX_RESOURCES];
    int finish[MAX_PROCESSES] = {0};
    int safeSequence[MAX_PROCESSES];
    int need[MAX_PROCESSES][MAX_RESOURCES];

    // Initialize work array with available resources
    for (i = 0; i < nResources; i++)
        work[i] = available[i];

    // Calculate the need matrix
    for (i = 0; i < nProcesses; i++)
        for (j = 0; j < nResources; j++)
            need[i][j] = max[i][j] - allocation[i][j];

    int count = 0;
    while (count < nProcesses)
```

```
    {
        int found = 0;
        for (i = 0; i < nProcesses; i++)
        {
            if (!finish[i])
            {
                int j;
                for (j = 0; j < nResources; j++)
                {
                    if (need[i][j] > work[j])
                        break;
                }

                if (j == nResources)
                {
                    for (k = 0; k < nResources; k++)
                        work[k] += allocation[i][k];
                    safeSequence[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }

        if (!found)
        {
            // No process found, indicating unsafe state
            return -1;
        }
    }

    printf("Safe sequence: ");
    for (i = 0; i < nProcesses; i++)
        printf("%d ", safeSequence[i]);
    printf("\n");

    return 0;
}

int main()
{
    int nProcesses, nResources;
    printf("Enter number of processes: ");
    scanf("%d", &nProcesses);
    printf("Enter number of resources: ");
    scanf("%d", &nResources);

    int available[MAX_RESOURCES];
    printf("Enter available resources: ");
    for (int i = 0; i < nResources; i++)
        scanf("%d", &available[i]);

    int max[MAX_PROCESSES][MAX_RESOURCES];
    printf("Enter maximum resources needed for each process:\n");
    for (int i = 0; i < nProcesses; i++)
```

```
        for (int j = 0; j < nResources; j++)
            scanf("%d", &max[i][j]);

    int allocation[MAX_PROCESSES][MAX_RESOURCES];
    printf("Enter resources currently allocated to each process:\n");
    for (int i = 0; i < nProcesses; i++)
        for (int j = 0; j < nResources; j++)
            scanf("%d", &allocation[i][j]);

    int result = bankerAlgorithm(available, max, allocation, nProcesses,
    nResources);

    if (result == -1)
        printf("Unsafe state\n");
    else
        printf("Safe state\n");

    return 0;
}
```