

Ex. No.: 9

Date: 4/1/25

DEADLOCK AVOIDANCE

Aim:

To find out a safe sequence using Banker's algorithm for deadlock avoidance.

Algorithm:

1. Initialize work=available and finish[i]=false for all values of i
2. Find an i such that both:
finish[i]=false and Need[i] ≤ work
3. If no such i exists go to step 6
4. Compute work=work+allocation[i]
5. Assign finish[i] to true and go to step 2
6. If finish[i]=true for all i, then print safe sequence
7. Else print there is no safe sequence

Program Code:

```
#include <stdio.h>
#include <stdbool.h>

int main() {
    int n, m;
    printf("Enter number of procs:");
    scanf("%d", &n);
    printf("Enter number of resources:");
    scanf("%d", &m);
    int max[n][m];
    printf("Enter the value for max array:");
    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            scanf("%d", &max[i][j]);
        }
    }
    int allocate[n][m];
    printf("Enter values for allocate array:");
```

```

for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++) {
        scanf ("%d", &allocate [i] [j]);
    }
}

int avail [m];
for (int i=0; i<m; i++)
{
    printf ("Enter Avail [%d]", i);
    scanf ("%d", &avail [i]);
}

int Need [n] [m];
for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++)
    {
        Need [i] [j] = max [i] [j] - allocate [i] [j];
    }
}

int work [m];
boolean finish [n];
for (int i=0; i<m; i++)
{
    work [i] = avail [i];
}

for (int i=0; i<n; i++)
    finish [i] = false;

int set [n];
int flag ind = 0;
while (ind != n) {

```

Max Array

$$\begin{bmatrix} 7 & 5 & 3 \\ 3 & 2 & 2 \\ 9 & 0 & 2 \\ 2 & 2 & 2 \\ 4 & 3 & 3 \end{bmatrix}$$

Allocation Array

$$\begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 6 & 2 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Available Array

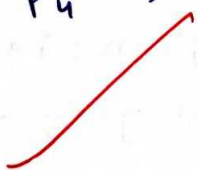
$$\begin{bmatrix} 3 & 2 & 2 \end{bmatrix}$$

Need Array

$$\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

Safe sequence :

$P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$



```
for (int i=0 ; i<n ; i++) {
```

```
    flag = 1;
```

```
    if ( finished [i] == false ) {
```

```
        for (int j=0 ; j<m ; j++) {
```

```
            if ( Need [i][j] > work [j])
```

```
                flag = 0;
```

```
        }
```

```
    } if ( flag == 1 ) {
```

```
        for (int j=0 ; j<m ; j++) {
```

```
            finish [i] = true;
```

```
            work [j] += allocate [i][j];
```

```
        }
```

```
        seq [ind++] = i;
```

```
    }
```

```
}
```

```
}
```

```
}
```

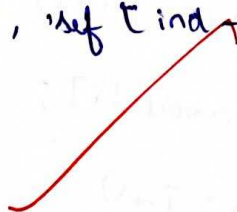
```
printf ("The SAFE SEQUENCE is\n");
```

```
for (int i=0 ; i<n-1 ; i++)
```

```
    printf ("P%d → ", seq[i]);
```

```
    printf ("P%d", seq[ind-1]);
```

```
}
```



Sample Output:

The SAFE Sequence is

P1 → P3 → P4 → P0 → P2

Result:

Program to find out a safe sequence using banker's algorithm for deadlock was written & executed successfully.

Shit