

Ex. No.: 9

Date: 3/4/25

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>
#define MAX-process 5
#define MAX-resources 3
bool isSafe(int p[], int avail[], int m[])
{
    int need[m][p];
    bool finish[p] = false;
    int work[MAX-resources];
    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            n[i][j] = max[i][j] - allot[i][j];
        }
    }
    for (int i=0; i<n; i++)
        w[i] = avail[i];
    int safe sequence [mp]; int c=0;
```



```
while (c < n) {
```

```
    bool found = false;
```

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

```
        if (!finished[i]) {
```

```
            int j;
```

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

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

```
                    break;
```

```
            }
```

```
            if (j == m) {
```

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

```
                    work[k] += allot[i][k]
```

```
                }
```

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

```
                safe_sequence[c++] = 1;
```

```
                found = true;
```

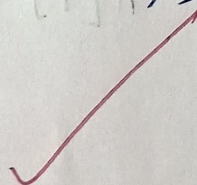
```
                break;
```

```
            }
```

```
        }
```

```
    }
```

```
    printf("safe sequence : ");
```




```

for (int i=0; i<n; i++) {
    printf (" %d " safe sequence [i]);
}
printf ("\n");
return true;
}

```

```

int main () {
    int processes [Max-processes];
    int avail [Max-resources];
    printf ("Enter available resources (A B C) : ");
    for (int i=0; i<Max-resources; i++)
        scanf ("%d", &avail[i]);
    int max [mp] [mr];
    printf ("Enter max demand matrix : \n");
    for (int j=0; j<mr; j++)
        scanf ("%d", &max[i][j]);
}

```

```

int allot [M-P] [M-R];
printf ("Enter allocation matrix : \n");
for (int i=0; i<Max-processes; i++) {
    printf ("Enter allocation for process p%d", i);
    for (int j=0; j<Max-resources; j++)
        scanf ("%d", &allot[i][j]);
}

```

3


```

int n = MAX_processes, m = max_resources;
if ( ! is_safe (processes, avail, max, allot, n, m)
    return 0;
}

```

Output:

Enter available resource (ABC) : 3 3 2

Enter max demand matrix :

$$P_0 = 7 \ 5 \ 3$$

$$P_1 = 3 \ 2 \ 2$$

$$P_2 = 9 \ 0 \ 2$$

$$P_3 = 2 \ 2 \ 2$$

$$P_4 = 4 \ 3 \ 3$$

Enter allocation matrix :

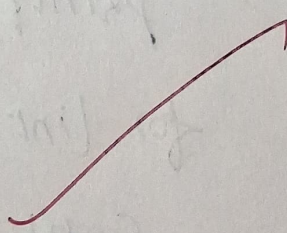
$$P_0 = 0 \ 1 \ 0$$

$$P_1 = 2 \ 0 \ 0$$

$$P_3 = 3 \ 0 \ 2$$

$$P_4 = 0 \ 0 \ 2$$

Safe sequence : $P_1 \ P_3 \ P_4 \ P_0 \ P_2$



Sample Output:

The SAFE Sequence is
P1 → P3 → P4 → P0 → P2

Result:

A C program for finding out safe
sequence is done using Banker's algorithm
Successfully

OK