OPERATING SYSTEM LAB (COM -312)

DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM

At

CSE, MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY (Computer Engineering)

Submitted by

Name: Aditya Vardhan Sharma, Akhilesh Sharma, Khushi, Lakshay Magotra
Roll no: 2021a1r035, 2021a1r031, 2021a1r039 ,2021a1r054
Branch: CSE
Semester: 3rd

TABLE OF CONTENTS

Acknowledgement

Abstract

1. Introduction

2. Objective

3. Requirement Analysis

4. Algorithm

5. Flowchart

6. Implementation
- **a) Coding**
- **b) Output**

7. References

ACKNOWLEDGEMENT

Under CRIE (Center for Research, Innovation and Entrepreneurship) we both worked under the guidance of mentors of CRIE, who guided us through a way leading to professionalism and practical hands on work experience.It is indeed with a great pleasure and immense sense of gratitude that we acknowledge the help of these individuals. We are highly indebted to our Director Ankur Gupta, "MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY", for the facilities provided to accomplish this main project. We would like to thank our Prof.Ashok Kumar , Head of the Department of Computer Science and Engineering , MIET. For this constructive criticism throughout our project .We feel elated in manifesting our sense of gratitude tour internal project guide. Asst. Saurabh Sharma , Department of Computer Science and Engineering ,MIET.He has been a constant source of inspiration for us and we are very deeply thankful to him.

FACULTY AND MEMBERS

Dr. Ankur Gupta - He is the Director at the ModeI Institute of Engineering and Technology, Jammu, India, besides being a Professor in the Department of Computer Science and Engineering. Prior to joining academia, he worked as Technical Team Lead at Hewlett Packard, developing software in the network management and e-Commerce domains. He has 2 patents to his name and 20 patents pending. He obtained his B.E, Hons. He is a senior member of the ACM, senior member IEEE and a life-member of the Computer Society of India. He has received competitive grants over Rs 2 crores from various funding agencies and faculty awards from IBM and EMC.

Prof. Ashok Kumar - He has over 21 years of experience in industry, academics, research and academic administration. He did his Doctorate from IKG Punjab Technical University, Jalandhar; Master of Engineering from Punjab Engineering College, Chandigarh and Bachelor's degree from NIT, Calicut, Kerala. He is Fellow of Institution of Electronics and Telecommunication Engineers (IETE). His research interest areas are Optical Networks, Electronics Product Design, Wireless Sensor Networks and Digital Signal Processing.. He is reviewer of many reputed national and international journals. He has also coordinated many national and international conferences.

Mr. Saurabh Sharma - He is working as Assistant Professor in the Department of Computer Science at Model Institute of Engineering and Technology, Jammu and has over 10 years of experience in academics and research. He has done his M. Tech CSE in Web Mining from Maharishi Markandeshwar University, Mullana, Ambala (HR) in the year 2011.His research interest areas are Web Mining, Data Mining, Machine Learning, ANN, Pattern Classification and Object Identification. He has, to his credit, 25 research papers published in journals of national and international repute and he has guided 11 M. Tech Dissertations.

ABSTRACT

Deadlock is a situation in which two or more processes are blocked and unable to proceed because they are waiting for a resource held by another process. Deadlock avoidance is a technique used to prevent deadlocks from occurring in a system by detecting and resolving potential deadlock situations before they occur. One well-known technique for deadlock avoidance is the Banker's Algorithm, which uses a resource-allocation graph to determine whether a system is in a safe state or not. If a system is in a safe state, it is possible to allocate resources to processes in a way that avoids deadlocks. If a system is not in a safe state, the Banker's Algorithm can be used to identify processes that must be delayed in order to avoid deadlocks. This report discusses the principles and implementation of the Banker's Algorithm for deadlock avoidance, its code and both the safety and request algorithm for deadlock avoidance.

OBJECTIVE

Implementation of most representative Banker's Algorithm in C Language for deadlock avoidance.

Idea is to introduce the application of an example of deadlock avoidance banker's algorithm. In the method of avoiding deadlock, the limited condition is weak, and it is possible to obtain satisfactory system performance. In this method, the system state is divided into safe state and unsafe state. As long as the system is always in safe state, deadlock can be avoided. We can do this by using bankers algorithm.The Banker's Algorithm is a resource allocation and deadlock avoidance algorithm that was developed by Edsger Dijkstra in the 1960s. It is used to ensure that a set of processes requesting resources from a common pool do not end up in a deadlock, where each process is waiting for a resource that is held by another process.

The Banker's Algorithm works by maintaining a set of available resources and a set of maximum resources needed by each process. It then allocates resources to processes in a way that ensures that no process exceeds its maximum resource needs and that no deadlocks occur.

REQUIREMENT ANALYSIS

The requirements of this project are:

1. Linux Operating System
2. GCC
3. Nano Editor
4. Bankers Algorithm

ALGORITHM DESIGN

Safety Algorithm:

1) Let Work and Finish be Arrays of length ‘m’ and ‘n’ respectively.

Initialize: Work = Available

Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both

a) Finish[i] = false

b) Needi <= Work

if no such i exists goto step (4)

3) Work = Work + Allocation[i]

Finish[i] = true

goto step (2)

4) if Finish [i] = true for all i

then the system is in a safe state

Request Algorithm:

1) If Request i <= Need i

Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2) If Request i <= Available

Goto step (3); otherwise, Pi must wait, since the resources are not available.

3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as

follows:

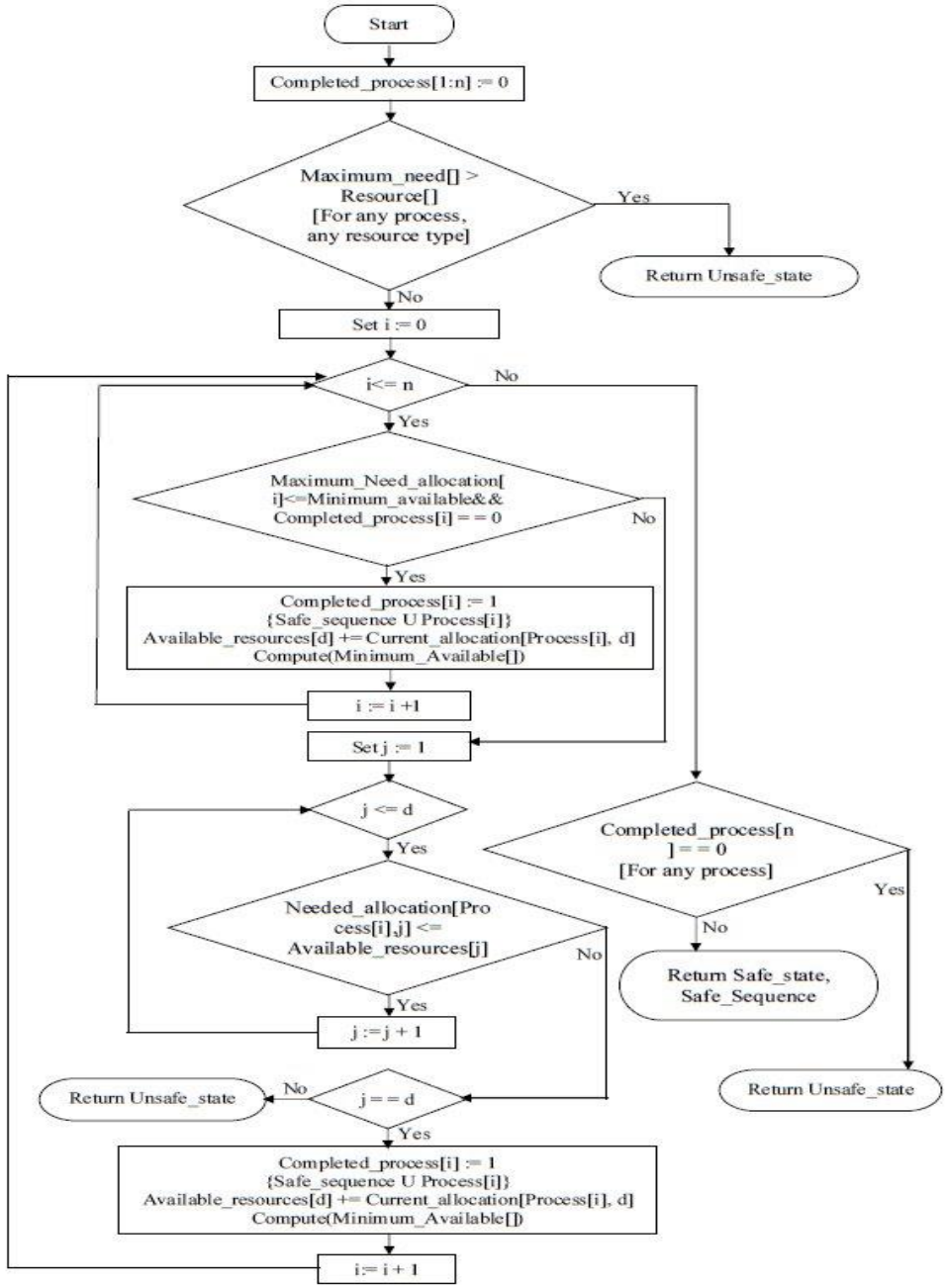
Available = Available – Request i

Allocation i = Allocation i +Request i

Need i = Need i– Request i

METHODOLOGY

FLOWCHART



IMPLEMENTATION

Code:

```
#include<stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();
int main()
{
 int i,j;
 printf("********** Banker's Algo ************\n");
 input();
 show();
 cal();
 getch();
 return 0;
}void input()
{
```

```
int i,j;
 printf("Enter the no of Processes\t");
 scanf("%d",&n);
 printf("Enter the no of resources instances\t");
 scanf("%d",&r);
 printf("Enter the Max Matrix\n");
 for(i=0;i<n;i++)
 {
  for(j=0;j<r;j++)
  {
   scanf("%d",&max[i][j]);
  }
 }
 printf("Enter the Allocation Matrix\n");
 for(i=0;i<n;i++)
 {
  for(j=0;j<r;j++)
  {
   scanf("%d",&alloc[i][j]);
  }
 }
 printf("Enter the available Resources\n");
 for(j=0;j<r;j++)
```

```
 {
  scanf("%d",&avail[j]);
 }
}
void show()
#include<stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();
int main()
{
 int i,j;
 printf("********** Banker's Algo ************\n");
 input();
 show();
 cal();
 getch();
 return 0;
}void input()
{
```

```
int i,j;
 printf("Enter the no of Processes\t");
 scanf("%d",&n);
 printf("Enter the no of resources instances\t");
 scanf("%d",&r);
 printf("Enter the Max Matrix\n");
 for(i=0;i<n;i++)
 {
  for(j=0;j<r;j++)
  {
   scanf("%d",&max[i][j]);
  }
 }
 printf("Enter the Allocation Matrix\n");
 for(i=0;i<n;i++)
 {
  for(j=0;j<r;j++)
  {
   scanf("%d",&alloc[i][j]);
  }
 }
 printf("Enter the available Resources\n");
 for(j=0;j<r;j++)
 {
  scanf("%d",&avail[j]);
 }
```

```
}
void show()
int i,j;
 printf("Process\t Allocation\t Max\t Available\t");
 for(i=0;i<n;i++)
 {
  printf("\nP%d\t   ",i+1);
  for(j=0;j<r;j++)
  {
   printf("%d ",alloc[i][j]);
  }
  printf("\t");
  for(j=0;j<r;j++)
  {
   printf("%d ",max[i][j]);
  }
  printf("\t");
  if(i==0)
  {
   for(j=0;j<r;j++)
   printf("%d ",avail[j]);
  }
 }
}
void cal()
{
```

```
int finish[100],temp,need[100][100],flag=1,k,c1=0;
int safe[100];
int i,j;
for(i=0;i<n;i++)
{
 finish[i]=0;
}
//find need matrix
for(i=0;i<n;i++)
{
 for(j=0;j<r;j++)
 {
  need[i][j]=max[i][j]-alloc[i][j];
 }
}
printf("\n");
while(flag)
{
 flag=0;
   for(i=0;i<n;i++)
 {
  int c=0;
  for(j=0;j<r;j++)
  {
   if((finish[i]==0)&&(need[i][j]<=avail[j]))
   {
```

```
      c++;
   if(c==r)
   {
    for(k=0;k<r;k++)
    {
     avail[k]+=alloc[i][j];
     finish[i]=1;
     flag=1;
         }
     printf("P%d->",i);
     if(finish[i]==1)
    {
      i=n;
     }
         }
  }
 }
  }
}
for(i=1;i<n;i++)
{
if(finish[i]==1)
{
 c1++;
}
else
```

```
 {
  printf("P%d->",i);
 }
}
if(c1==n)
{
 printf("\n The system is in safe state");
}
else
{  printf("\n Process are in dead lock");
printf("\n System is in unsafe state");
}
}
```

OUTPUT

Safe state:

```
********** Banker's Algo ************
Enter the no of Processes        3
Enter the no of resources instances      2
Enter the Max Matrix
4
8
3
8
2
8
Enter the Allocation Matrix
1
9
3
8
6
8
Enter the available Resources
6
9
Process  Allocation       Max      Available
P1          1 9           4 8              6 9
P2          3 8           3 8
P3          6 8           2 8
P1->P2->P3->
 The system is in safe stateaditya@Aditya:~$
```

Unsafe State:

```
********** Banker's Algo ************
Enter the no of Processes        3
Enter the no of resources instances      2
Enter the Max Matrix
5
6
2
55
8
3
Enter the Allocation Matrix
9
2
5
3
9
3
Enter the available Resources
6
9
Process  Allocation       Max      Available
P1          9 2           5 6              6 9
P2          5 3           2 55
P3          9 3           8 3
P2->P3->P1->
 Process are in dead lock
 System is in unsafe stateaditya@Aditya:~$
```

CONCLUSION

In conclusion, the Banker's Algorithm is a valuable tool for avoiding deadlocks in a system that allocates resources to multiple processes. By keeping track of the available resources and the maximum resources needed by each process, the algorithm ensures that no process exceeds its resource needs and that no deadlocks occur. The Banker's Algorithm is widely used in operating systems and other software that manage resources in a multi-tasking environment, and has proven to be an effective method for ensuring the efficient and deadlock-free allocation of resources. While the algorithm is not foolproof and can still lead to deadlocks in certain situations, it is a valuable tool for avoiding deadlocks in many cases.

REFERENCES

1. www.geekforgeeks.com
2. http://suraj1693.blogspot.com/2013/11/program-for-bankers-algorithm-for.html
3. https://chat.openai.com
4. www.programiz.com
5. www.javapoint.com
6. www.researchgate.com
7. www.codewithharry.in