BANKER’S ALGORITHM

Safety Algorithm

#include<iostream>

using namespace std;

int main(){

    int n=5;

    int m=3;

    int total[3]={10,5,7};

    int available[3]={3,3,2};

    int work[3]={3,3,2};// copy of available to make changes when the resources are released

    int max[5][3] = { { 7, 5, 3 }, { 3, 2, 2 }, { 9, 0, 2 }, { 2, 2, 2 }, { 4, 3, 3 } }; // max matrix for P0,P1,P2,P3

    int alloc[5][3] = { { 0, 1, 0 }, { 2, 0, 0 },{ 3, 0, 2 },{ 2, 1, 1 },{ 0, 0, 2 } }; //allocation matrix for P0,P1,P2,P3

    int need[n][m];

    for(int i=0;i<n;i++)

    {

        for(int j=0;j<m;j++)

        {

            need[i][j]=max[i][j]-alloc[i][j];

        }

    }

    bool finish[n]={0};

    int safeseq[n];

    int count =0;

    while(count<n){

        bool found =false;

        for(int i=0;i<n;i++){

            if(finish[i]==false){

                int a=0;

                while(a<m)

                {

                    if(need[i][a]>work[a]){

                    break;}

                    a++;

                }

                if(a==m){

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

                        work[k]+=alloc[i][k];

                    }

                    finish[i]=1;

                    found=true;

                    safeseq[count]=i;

                    count++;

                }

            }

        }

        if (found == false)

        {

            cout<< "System is not in safe state";

            return 0;

        }

    }

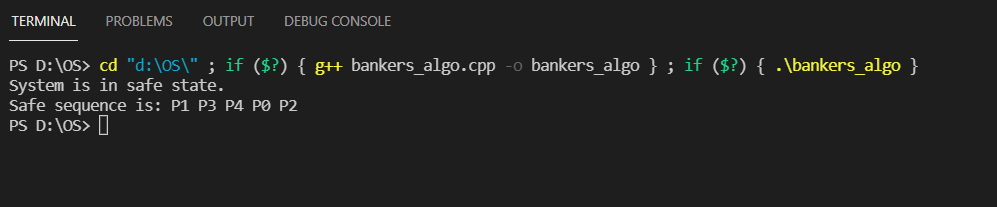
    cout << "System is in safe state.\nSafe sequence is: ";

    for (int i = 0; i < n ; i++)

        cout <<"P"<<safeseq[i]<< " ";

    return 0;

}



Resource-Request Allocation Algorithm

P1 makes the request for resources -> {1,0,2}

#include<iostream>

using namespace std;

//Resource allocation algorithm when P1 makes the requests

void res\_alloc(int available[],int m,int need[][3],int alloc[][3]){

    int req[1][3]={1,0,2};

    for(int i=0;i<m;i++){

        if(req[0][i]>need[1][i]){

            cout<<"error encountered"<<endl;

            exit(0);

        }

    }

    for(int i=0;i<m;i++){

        if(req[0][i]>available[i]){

            cout<<"Resources unavailable"<<endl;

            exit(0);

        }

    }

    for(int i=0;i<m;i++){

        available[i]-=req[0][i];

        alloc[1][i]+=req[0][i];

        need[1][i]-=req[0][i];

    }

}

int main(){

    int n=5;

    int m=3;

    int total[3]={10,5,7};//total instances of each resource type

    int available[3]={3,3,2};//available instances

    int max[5][3] = { { 7, 5, 3 }, { 3, 2, 2 }, { 9, 0, 2 }, { 2, 2, 2 }, { 4, 3, 3 } }; // max matrix for P0,P1,P2,P3

    int alloc[5][3] = { { 0, 1, 0 }, { 2, 0, 0 },{ 3, 0, 2 },{ 2, 1, 1 },{ 0, 0, 2 } }; //allocation matrix for P0,P1,P2,P3

    int need[5][3];

    for(int i=0;i<n;i++)

    {

        for(int j=0;j<m;j++)

        {

            need[i][j]=max[i][j]-alloc[i][j];

        }

    }

    //applying resource allocation algo for p1

    res\_alloc(available,3,need,alloc);

    int work[3];

    for(int i=0;i<m;i++){

        work[i]=available[i];

    }

    bool finish[n]={0};

    int safeseq[n];

    int count =0;

    while(count<n){

        bool found =false;

        for(int i=0;i<n;i++){

            if(finish[i]==false){

                int a=0;

                while(a<m)

                {

                    if(need[i][a]>work[a]){

                    break;}

                    a++;

                }

                if(a==m){

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

                        work[k]+=alloc[i][k];

                    }

                    finish[i]=1;

                    found=true;

                    safeseq[count]=i;

                    count++;

                }

            }

        }

        if (found == false)

        {

            cout<< "System is not in safe state";

            cout<<"REQUEST CANNOT BE GRANTED"<<endl;

            return 0;

        }

    }

    cout << "System is in safe state.\nREQUEST CAN BE GRANTED\nSafe sequence is: ";

    for (int i = 0; i < n ; i++)

        cout <<"P"<<safeseq[i]<< " ";

    return 0;

}

