**MATRIX CHAIN**

#include<stdio.h>

#include<limits.h>

int MatrixChainMultiplication(int p[], int n)

{

int m[n][n];

int i, j, k, L, q;

for (i=1; i<n; i++)

m[i][i] = 0;

for (L=2; L<n; L++)

{

for (i=1; i<n-L+1; i++)

{

j = i+L-1;

m[i][j] = INT\_MAX;

for (k=i; k<=j-1; k++)

{

q = m[i][k] + m[k+1][j] + p[i-1]\*p[k]\*p[j];

if (q < m[i][j])

{

m[i][j] = q;

}

}

}

}

return m[1][n-1];

}

int main()

{

int n,i;

printf("Enter number of matrices\n");

scanf("%d",&n);

n++;

int arr[n];

printf("Enter dimensions \n");

for(i=0;i<n;i++)

{

printf("Enter d%d :: ",i);

scanf("%d",&arr[i]);

}

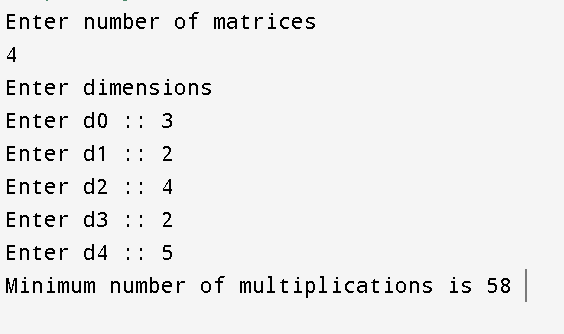
int size = sizeof(arr)/sizeof(arr[0]);

printf("Minimum number of multiplications is %d ", MatrixChainMultiplication(arr, size));

return 0;

}

**Output:**



**BIPARTIATE COVER**

#include <iostream>

#define MAX 10

using namespace std;

int graph[MAX][MAX], setA[MAX], setB[MAX];

int vertA, vertB;

bool bipartiteMatch(int u, bool \*visited, int \*check)

{

for (int i = 0; i < vertA; i++)

{

if (graph[u][i] && !visited[i])

{

visited[i] = true;

if (check[i] < 0 || bipartiteMatch(check[i], visited, check))

{

check[i] = u;

return true;

}

}

}

return false;

}

int maxMatch()

{

int check[vertA];

bool visited[vertA];

for(int i = 0; i<vertA; i++)

{

check[i] = -1;

}

int result = 0;

for (int u=0; u<vertB; u++)

{

for(int i=0; i<vertA; i++) visited[i] = false;

if(bipartiteMatch(u, visited, check)) result++;

}

cout<<"Matched vertices set = {";

for(int i=0; i<vertA; i++)

{

cout<<"("<<setA[i]<<", "<<setB[check[i]]<<")";

if(i != vertA-1) cout<<", ";

}

cout<<"}";

cout<<”\nMaximum number of matching: “;

return result;

}

int main() {

cout<<"Enter number of vertices in setA and setB of bipartite graph: ";

cin>>vertA>>vertB;

cout<<"Enter vertices of setA: ";

for(int i=0; i<vertA; i++) cin>>setA[i];

cout<<"Enter vertices of setB: ";

for(int i=0; i<vertB; i++) cin>>setB[i];

cout<<"Enter "<<vertA<<" x "<<vertB<<" adjacency matrix representing relationship between set (A -> B):\n";

for(int i=0; i<vertA; i++){

for(int j=0; j<vertB; j++){

cin>>graph[i][j];

}

}

cout<<maxMatch();

return 0;

}

Output:

