جامعة المنارة

كلية: الهندسة

قسم: الهندسة المعلوماتية

اسم المقرر: الخوارزميات و بنى المعطيات2

رقم الجلسة (الرابعة)

عنوان الجلسة

خوارزمية بريم لإيجاد الشجرة المولدة الصغرى للبيان
(Prim's algorithm)

الفصل الدراسي الثاني

العام الدراسي 2023- 2024

الغاية من الجلسة

- ✓ تطبيق خوارزمية بريم لإيجاد الشجرة المولدة الصغرى للبيان .
- ✓ تنفيذ الكود الخاص بخوارزمية بريم.

الشجرة المولدة الصغرى للبيان Minimum Spanning Tree (MST) :

- ➢ تعرف الشجرة المولدة الصغرى لبيان متصل و غير موجه و موزون : بأنها شجرة غير موجهة تمثل بيان جزئي من البيان الأصلي تحوي على جميع رؤوس البيان الأصلي ، وتملك هذه الشجرة أقل وزن .

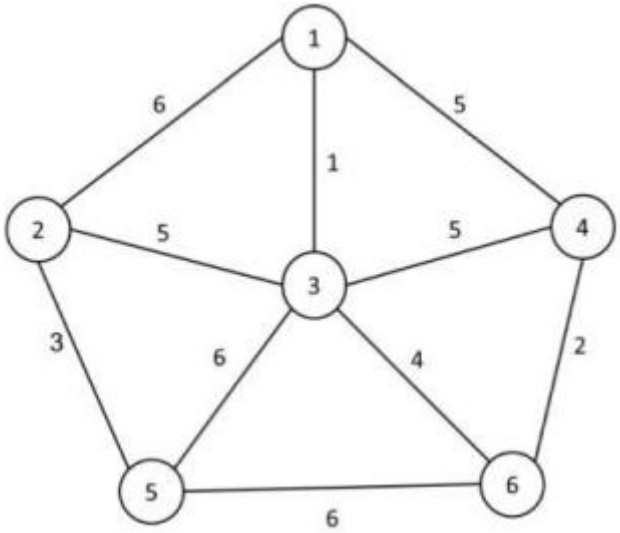
سوف ندرس خوارزميتين لإيجاد الشجرة المولدة الصغرى لبيان هما : خوارزمية بريم وخوارزمية كروسكال.

خوارزمية بريم (Prim's algorithm) :

- ➢ تستخدم خوارزمية بريم لإيجاد الشجرة المولدة الصغرى للبيان .

تمرين1 :

طبق خوارزمية بريم لإيجاد الشجرة المولدة الصغرى للبيان التالي :



1- نختار الرأس 1 ليكون رأس البداية و نهيئ المصفوفة distance بقيم مصفوفة التجاور للرأس 1 ، كما نهيئ المصفوفة nearest بالقيمة 1 دلالة على الرأس 1 .

Nearest array				
2	3	4	5	6
1	1	1	1	1

Distance array				
2	3	4	5	6
6	1	5	∞	∞

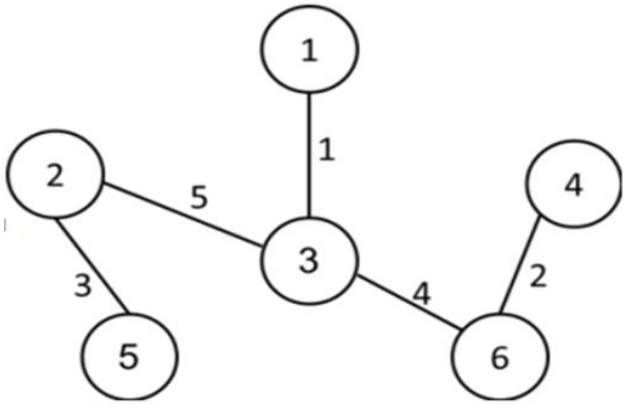
2- في كل تكرار نختار من المصفوفة distance الرأس vnear ذي الوزن الأقل ، و نقارن أوزان الأحرف التي تصل الرأس vnear بالرؤوس المجاورة له مع أوزان نفس تلك الرؤوس في المصفوفة distance ،فإذا كانت أقل يتم تعديل المصفوفة distance بالأوزان الجديدة ،ويقابل ذلك تعديل في قيم المصفوفة nearest الموافقة، و نكرر هذه العمليات حتى يتم زيارة جميع رؤوس البيان كمايلي :

رقم التكرار	Nearest				
	2	3	4	5	6
1	1	1	1	1	1
2	3	1	1	3	3
3	3	1	6	3	3
4	3	1	6	3	3
5	3	1	6	2	3

رقم التكرار	Distance				
	2	3	4	5	6
1	6	1	5	∞	∞
2	5	-1	5	6	4
3	5	-1	2	6	-1
4	5	-1	-1	6	-1
5	-1	-1	-1	3	-1
stop	-1	-1	-1	-1	-1

MIN	vnear
1	3
4	6
2	4
5	2
3	5

MST tree edges
(1,3)
(3,6)
(6,4)
(3,2)
(2,5)



البرنامج الخاص بخوارزمية بريم :

```
/* prim minimum spanning tree from node 1 */
#include <iostream>
using namespace std;
#define  INFF 999
const int n=6;
struct edge{
        int x1;
        int x2;
    };
void prim( int w[n+1][n+1],edge* F){
        int vnear;
        int min;
        int i,j;
        int f_size=0;
        //the nearest node from the vertex 2 to n
        int * nearest=new int [n+1];
        //distance of each vertex and the nearest node
        int *distance=new int[n+1];
        for(i=2;i<n+1;i++){
            nearest[i]=1;
            distance[i]=w[1][i];
        }
        //find vertex with minimum distance
        for(i=2;i<n+1;i++)
        {
            min =INFF;
            for(j=2;j<n+1;j++)
            {

                if(distance[j]<min&&distance[j]>=0)
                {
                    min=distance[j];
                    vnear=j;
                }
            }
            edge new_edge;
            new_edge.x1=vnear;
            new_edge.x2=nearest[vnear];
            //add edge to MST
            F[f_size]=new_edge;
            f_size++;
```

```
            //vnear is visited
            distance[vnear]=-1;

        for(j=2;j<n+1;j++){
         if (w[j][vnear]<distance[j]){
            distance[j]=w[j][vnear];
            nearest[j]=vnear;
        }
        }
        }
}
int main()
{
    int weigth_matrix[n+1][n+1]={{0,0,0,0,0,0,0},
                                 {0,0,6,1,5,INFF,INFF},
                                 {0,6,0,5,INFF,3,INFF},
                                 {0,1,5,0,5,6,4},
                                 {0,5,INFF,5,0,INFF,2},
                                 {0,INFF,3,6,INFF,0,6},
                                 {0,INFF,INFF,4,2,6,0}
                                };
     //MST maximum edge number is n-1
     edge * F =new edge [n-1];
     prim(weigth_matrix,F);
     int i;
     int totalcost=0;
     cout<<"minimum spanning tree:\n";
     for(i=0;i<n-1;i++)
     {
         cout<<"edge:"<<F[i].x1<<","<<F[i].x2<< "
weight:"<<weigth_matrix[F[i].x1][F[i].x2]<<endl;
         totalcost+=weigth_matrix[F[i].x1][F[i].x2];
     }
     cout<<"total cost="<<totalcost<<endl;
}
```

خرج البرنامج :

تمرين غير محلول:

طبق خوارزمية بريم لإيجاد شجرة مولدة صغرى للبيان التالي بحيث يتم البدء من الرأس v1 :