جامعة المنارة

كلية: الهندسة

قسم: الهندسة المعلوماتية

اسم المقرر: الخوارزميات و بنى المعطيات2

رقم الجلسة (الخامسة)

عنوان الجلسة

خوارزمية كروسكال لإيجاد الشجرة المولدة الصغرى للبيان
(Kruskal’s algorithm)

الفصل الدراسي الأول

العام الدراسي 2023- 2024

الغاية من الجلسة

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

الشجرة المولدة الصغرى للبيان Minimum Spanning Tree :

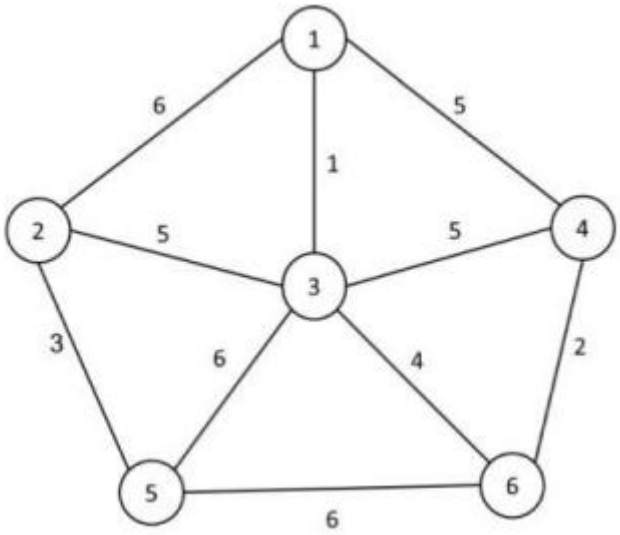
- ➢ تعرف شجرة التغطية الصغرى لبيان متصل و غير موجه و موزون : بأنها شجرة غير موجهة تمثل بيان جزئي من البيان الأصلي تحوي على جميع رؤوس البيان الأصلي ، و تملك هذه الشجرة أقل وزن .

خوارزمية كروسكال :

- ➢ تستخدم خوارزمية كروسكال لإيجاد الشجرة المولدة الصغرى للبيان .

تمرين:

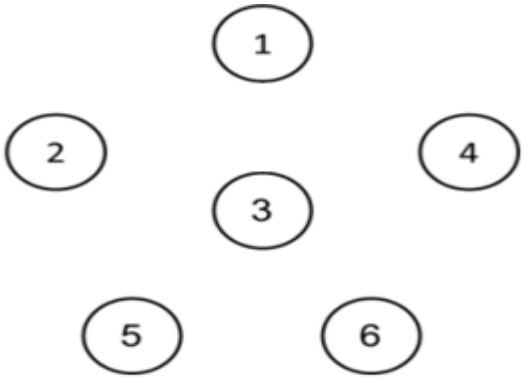
طبق خوارزمية كروسكال لإيجاد شجرة مولدة صغرى للبيان التالي :



1- نرتب الأحرف حسب أوزانها تصاعدياً .

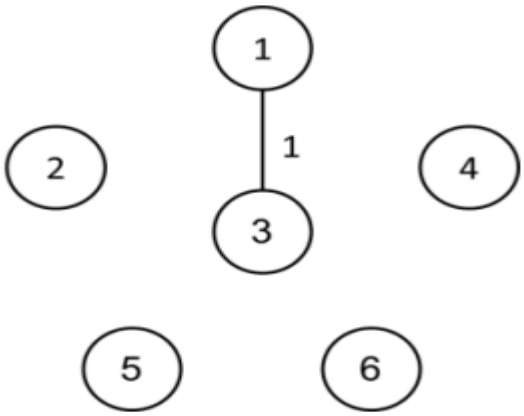
edge	(1,3)	(4,6)	(2,5)	(3,6)	(1,4)	(3,4)	(2,3)	(1,2)	(3,5)	(5,6)
weight	1	2	3	4	5	5	5	6	6	6

2- سوف نستخدم مفهوم المجموعات المتباعدة لاكتشاف الأحرف التي تشكل حلقات في الشجرة المولدة الصغرى ، لذلك نعتبر من البداية أنه لدينا ست مجموعات بعدد الرؤوس ، و كل مجموعة تحوي رأس من رؤوس البيان ، و نهيئ المصفوفة Parent بالقيم التالية :

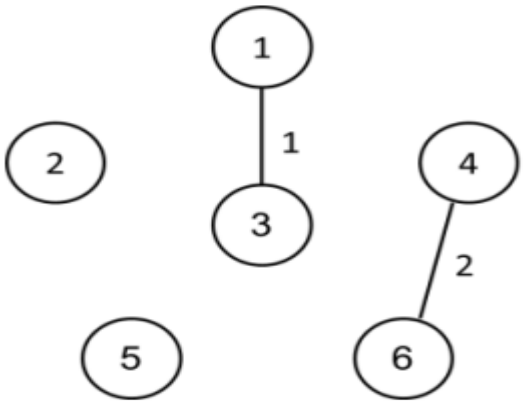


Parent array					
1	2	3	4	5	6
1	2	3	4	5	6

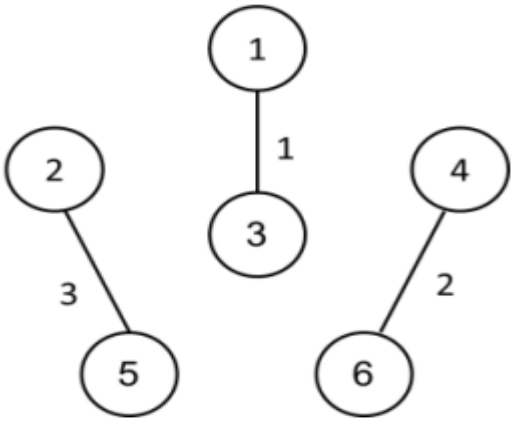
2- نختار في كل مرة الحرف (i,j) ذي الوزن الأقل و الذي يحقق parent[i]#parent[j] أي أن اختياره لن يؤدي إلى تكوين دورة في الشجرة المولدة الصغرى ثم نقوم بدمج مجموعتي الرأسين مع بعضهما، و نتجنب اختيار كل حرف (i,j) يحقق parent[i]=parent[j] لأنه سيؤدي إلى تكوين دورة .



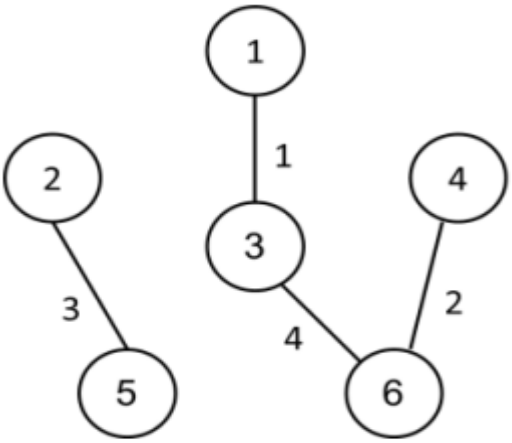
Parent array					
1	2	3	4	5	6
1	2	1	4	5	6



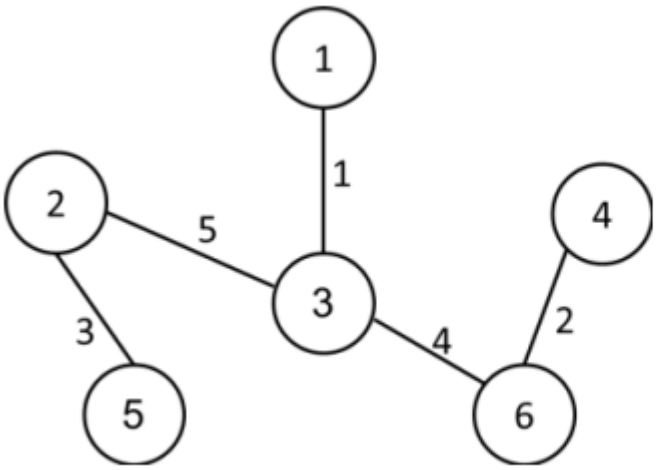
Parent array					
1	2	3	4	5	6
1	2	1	4	5	4



Parent array					
1	2	3	4	5	6
1	2	1	4	2	4



Parent array					
1	2	3	4	5	6
1	2	1	1	2	4



Parent array					
1	2	3	4	5	6
2	2	1	1	2	4

البرنامج الخاص بخوارزمية كروسكال:

```
/*Kruskal Algorithm to find minimum spanning tree (in
undirected  graph)*/

 #include<iostream>
 #include<algorithm>

 using namespace std;
  struct Edge
           {
    int src;
    int dest;
    int wt;
          };

   bool compare(Edge e1,Edge e2){

    return e1.wt<e2.wt;

   }
 int getParent(int v,int parent[])
   {

    if(parent[v]==v){

        return v;
```

```
    }

    return getParent(parent[v],parent);

   }

int main()

{
   int n,E;
   cout<<"enter number of vertices : ";
   cin>>n;
   cout<<"enter number of edges : ";
   cin>>E;
   cout<<"enter edges (edge start,edge end, weight): \n";

   Edge edges[100];

 for(int i=0;i<E;i++){
        cin>>edges[i].src>>edges[i].dest>>edges[i].wt;
    }

    //sorted the edges array in increasing order

    sort(edges,edges+E,compare);

    Edge output[100];

    //Union find algorithm to detect cycle

    int parent[n+1];

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

        parent[i]=i;
    }
    int count=0;
    int i=0;
    while(count<n-1){
        Edge currentEdge=edges[i];

        int p1=getParent(currentEdge.src,parent);
```

```
        int p2=getParent(currentEdge.dest,parent);

        if(p1!=p2){

            output[count]=currentEdge;

            count++;

            parent[p2]=p1;

        }

        i++;

    }

    int totalcost=0;
    cout<<"minimum spanning tree\n";

    //Printing the MST

    for(int i=0;i<n-1;i++)
        {

        cout<<output[i].src<<" "<<output[i].dest<<"
"<<output[i].wt<<endl;

        totalcost+=output[i].wt;

        }
    cout<<"total cost="<<totalcost;
    cout<<endl;


    return 0;

    }
```

خرج البرنامج :

```
enter number of vertices : 6
enter number of edges : 10
enter edges (edge start,edge end, weight):
1 2 6
1 3 1
1 4 5
2 3 5
2 5 3
3 4 5
3 6 4
3 5 6
4 6 2
5 6 6
minimum spanning tree
1 3 1
4 6 2
2 5 3
3 6 4
2 3 5
total cost=15

Process returned 0 (0x0)   execution time : 103.921 s
Press any key to continue.
```

تمرين غير محلول:

طبق خوارزمية كروسكال لإيجاد شجرة مولدة صغرى للبيان التالي :