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 الثاني

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 (MST) :

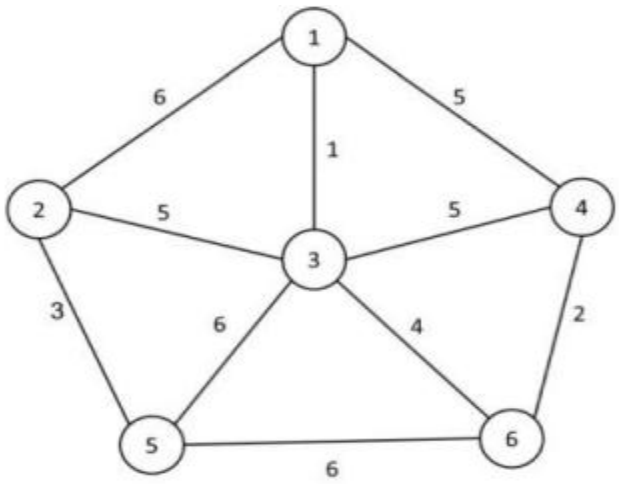
- ➢ تعرف الشجرة المولدة الصغرى لبيان متصل و غير موجه و موزون : بأنها شجرة غير موجهة تمثل بيان جزئي من البيان الأصلي تحوي على جميع رؤوس البيان الأصلي ، وتملك هذه الشجرة أقل وزن .

درسنا في الجلسة الماضية خوارزمية بريم لإيجاد الشجرة المولدة الصغرى لبيان و الآن ندرس خوارزمية كروسكال لإيجاد الشجرة المولدة الصغرى لبيان.

2- خوارزمية كروسكال (Kruskal's algorithm) :

- ➢ تستخدم خوارزمية كروسكال لإيجاد الشجرة المولدة الصغرى للبيان .
- ➢ يستخدم في تطبيق خوارزمية كروسكال مفهوم المجموعات المتباعدة disjoint sets.
- ➢ تعرف المجموعات المتباعدة على أنها مجموعات لا تحوي أية عناصر مشتركة فيمابينها.
- ➢ يوجد لكل مجموعة أب جذر (root) يمثل هذه المجموعة و يميزها عن المجموعات الأخرى، و تنتمي كل العناصر الموجودة في المجموعة الواحدة لأب جذر واحد .
- ➢ توجد عدة عمليات على المجموعات المتباعدة نذكر منها :
  - ✓ Find_root(i) : يوجد الأب الجذر للمجموعة التي ينتمي لها العنصر i .
  - ✓ Union(i , j) : يقوم بدمج المجموعتين المتباعدتين التي ينتمي لهما العنصرين i,j في مجموعة واحدة .

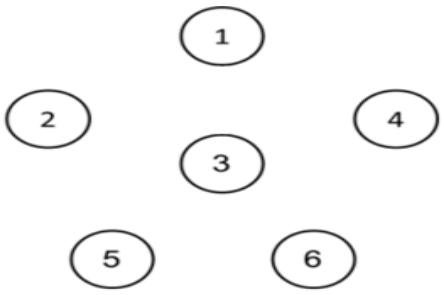
تمرين: طبق خوارزمية كروسكال لإيجاد الشجرة المولدة الصغرى للبيان التالي :



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 disjoint subsets لاكتشاف الأحرف التي تشكل حلقات في الشجرة المولدة الصغرى ، لذلك نعتبر من البداية أنه لدينا ست مجموعات متباعدة 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

3- نستخدم التابع find_root(i) الذي يوجد الأب الجذر للمجموعة التي ينتمي لها العنصر i .

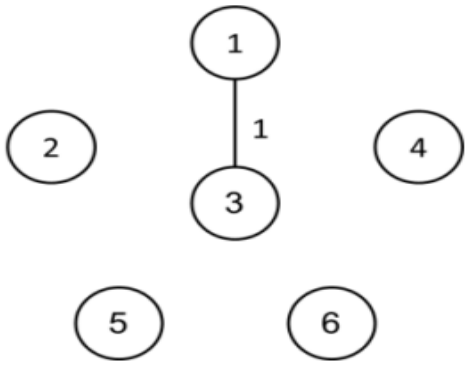
لذلك نختار في كل مرة من البيان الحرف (i,j) ذي الوزن الأقل و الذي يحقق find_root(i)#find_root(j) أي أن الرأسين i,j يقعان في مجموعتين مختلفتين بالتالي فإن إضافة الحرف (i,j) لن يؤدي إلى تكوين دورة في الشجرة المولدة الصغرى ، ثم نقوم بدمج مجموعتي الرأسين مع بعضهما في مجموعة واحدة، و نتجنب إضافة أي حرف (i,j) يحقق find_root(i)= find_root(j) لأنه سيؤدي إلى تكوين دورة في الشجرة المولدة الصغرى.

و نتوقف عن التكرار عندمايكون : عدد أحرف الشجرة = عدد عقد البيان - 1 .

i	j	p = find_root(i)	q = find_root(j)	(p # q) ?	MST edges	update Parent Array: Parent[q] = p	Number of MST edges
1	3	1	3	✓	(1,3)	Parent array: 1 2 3 4 5 6 / 1 2 1 4 5 6	1
4	6	4	6	✓	(4,6)	Parent array: 1 2 3 4 5 6 / 1 2 1 4 5 4	2
2	5	2	5	✓	(2,5)	Parent array: 1 2 3 4 5 6 / 1 2 1 4 2 4	3

3	6	1	4	✓	(3,6)	Parent array: 1 2 3 4 5 6 / 1 2 1 1 2 4	4
1	4	1	1	✗	Edge makes a cycle in MST tree		
3	4	1	1	✗	Edge makes a cycle in MST tree		
2	3	2	1	✓	(2,3)	Parent array: 1 2 3 4 5 6 / 2 2 1 1 2 4	5
Stop iteration because number of MST edges = number of graph vertices - 1							

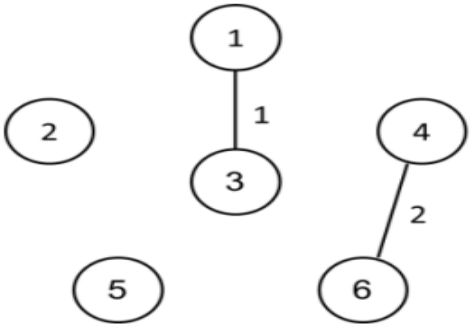
✓ نضيف الحرف (1,3) إلى الشجرة :



Parent array

1	2	3	4	5	6
1	2	1	4	5	6

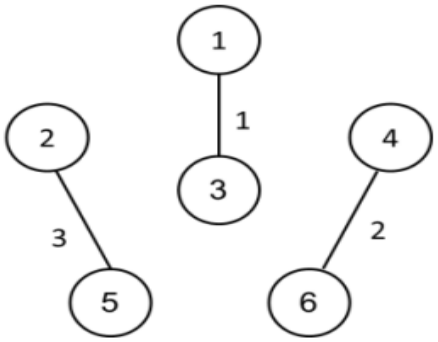
✓ نضيف الحرف (4,6) إلى الشجرة :



Parent array

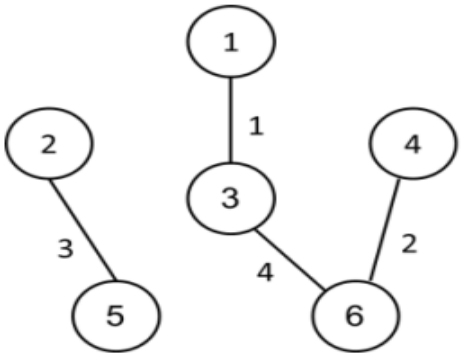
1	2	3	4	5	6
1	2	1	4	5	4

✓ نضيف الحرف (2,5) إلى الشجرة :



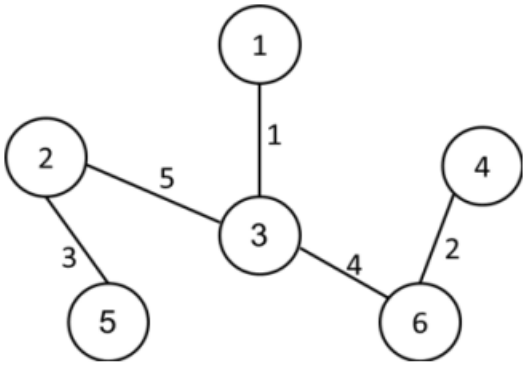
Parent array					
1	2	3	4	5	6
1	2	1	4	2	4

✓ نضيف الحرف (3,6) إلى الشجرة :



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

✓ نضيف الحرف (2,3) إلى الشجرة :



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 find_root(int v,int parent[])
   {

    if(parent[v]==v){

        return v;

    }

    return find_root(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[100];

  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 p=find_root(currentEdge.src,parent);

      int q=find_root(currentEdge.dest,parent);

      if(p!=q){

          output[count]=currentEdge;

          count++;

          parent[q]=p;

      }

      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;

    }
```

تمرين غير محلول:

طبق خوارزمية كروسكال لإيجاد شجرة مولدة صغرى للبيان التالي :