جامعة المنارة

كلية: الهندسة

قسم: الهندسة المعلوماتية

اسم المقرر: الخوارزميات و بنى المعطيات2

رقم الجلسة (الثانية)

عنوان الجلسة

خوارزمية البحث بالعمق أولاً في البيان
(DFS) Depth First Search

الفصل الدراسي الثاني

العام الدراسي 2023- 2024

الغاية من الجلسة

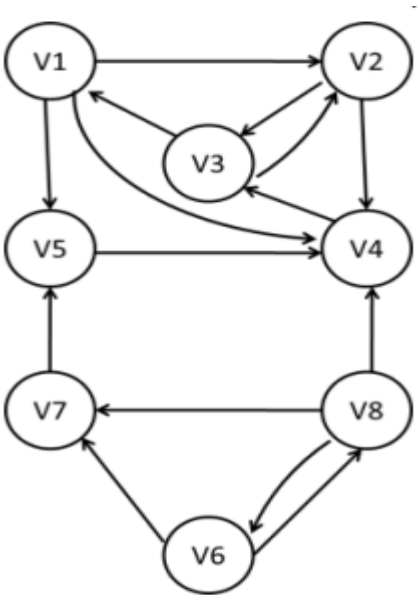
- ✓ تطبيق خوارزمية البحث بالعمق أولاً في البيان .

خوارزمية البحث بالعمق أولاً في البيان

- ➢ تستخدم خوارزمية البحث بالعمق أولاً لزيارة رؤوس البيان بحيث أنه عند زيارة أي رأس جديد نستمر في البحث متجهين إلى العمق قدر الاستطاعة .
- ➢ يتم تطبيق الخورازمية برمجياً عن طريق الاستدعاء العودي .

تمرين:

طبق الاستدعاء العودي لخوارزمية البحث بالعمق على البيان التالي :



قبل البدء بعملية التجول عبر البيان نقوم بتهيئة المصفوفة visited بالقيم المنطقية false ، دلالة على أنه لم تتم زيارة أي رأس بعد .

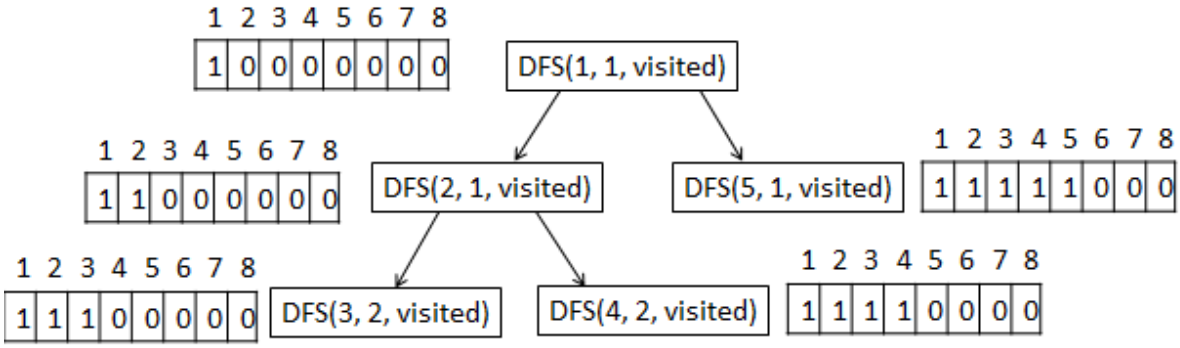
visited

1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0

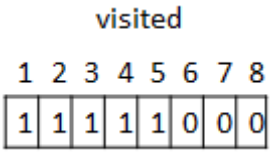
و نبدأ باستدعاء تابع البحث بالعمق بدءاً من الرأس 1

```
DFS(v, pred, visited)
```

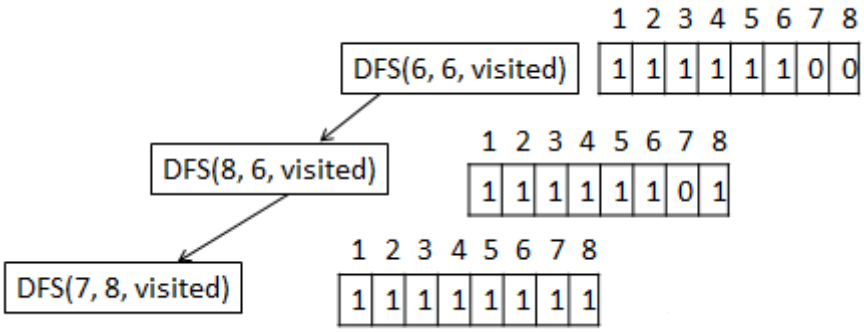
الشجرة التالية تمثل الشجرة الناتجة عن تنفيذ خوارزمية البحث بالعمق بدءاً من الرأس 1 .



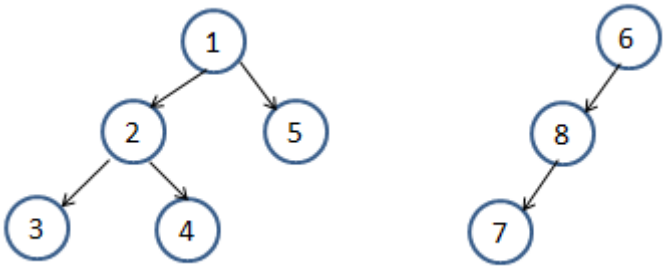
و لكن نلاحظ من خلال المصفوفة visited أنه لم تتم زيارة جميع رؤوس البيان .



لذلك نستدعي تابع البحث بالعمق انطلاقاً من الرأس 6 .



نلاحظ أنه تمت زيارة جميع رؤوس البيان ، و قد نتج عن تطبيق خوارزمية البحث بالعمق الشجرتين التاليتين :



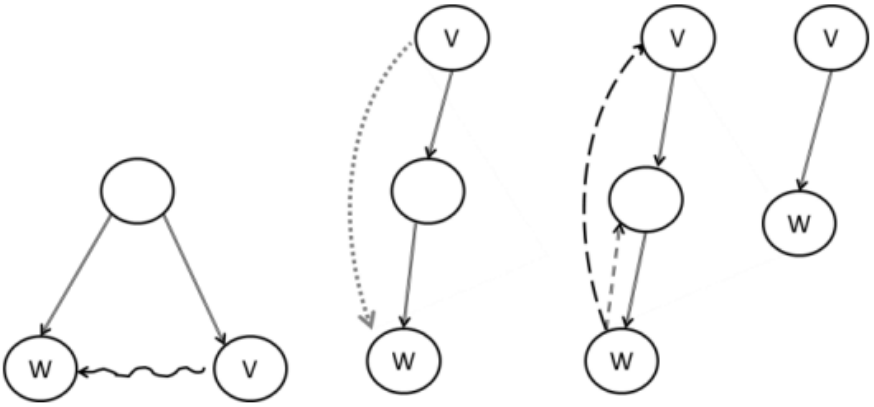
الانواع الاربعة من الاحرف.

حرف شجرة: خط متصل

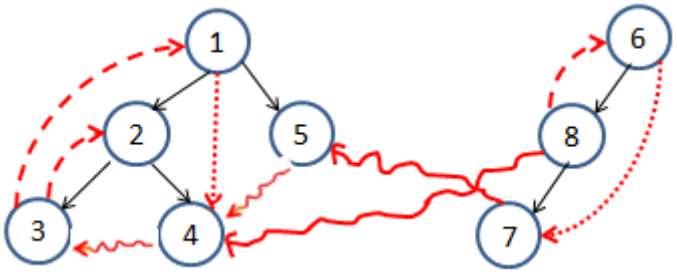
حرف خلفى: خط متقطع

حرف أمامي: خط منقط

حرف مستعرض: خط متعرج



تحديد أنواع الأحرف المختلفة على غابة البحث بالعمق الناتجة عن البيان السابق .



البرنامج الخاص بخوارزمية البحث بالعمق أولاً :

```
#include <iostream>
#include <list>
using namespace std;
//graph class for DFS travesal
class DFSGraph
{
int V;    // No. of vertices
list<int> *adjList;    // adjacency list
void DFS(int v,int pred, bool visited[]);
public:
 // class Constructor
 DFSGraph(int V)
    {
  this->V = V;
  adjList = new list<int>[V+1];
    }
 // function to add an edge to graph
 void addEdge(int v, int w){
  adjList[v].push_back(w); // Add w to v's list.
    }

 void applyDFS();
};
void DFSGraph::DFS(int v,int pred, bool visited[])
{
 // current node v is visited
 visited[v] = true;
 if(pred==v)
    cout<<"\n new tree, root="<<v<<" , edeges : ";
    else
 cout << pred<<v<< " ";

//recursively process all the adjacent vertices of the
node
 list<int>::iterator i;
 for(i = adjList[v].begin(); i != adjList[v].end(); ++i)
  if(!visited[*i])
   DFS(*i,v, visited);
```

```
}
// DFS traversal
void DFSGraph::applyDFS()
{
    // initially none of the vertices are visited
 bool visited[1000];
 for (int i = 1; i < V+1; i++){
   visited[i] = false;

}
 // explore the vertices one by one by recursively calling
apply
for (int i = 1; i < V+1; i++)
 if (visited[i] == false)
  DFS(i,i, visited);
}
int main()
{
 // Create a graph
 cout << "Depth-first traversal for the given
graph:"<<endl;
 DFSGraph g(8);

 g.addEdge(1,2);
 g.addEdge(1,5);
 g.addEdge(1,4);
 g.addEdge(2,3);
 g.addEdge(2,4);
 g.addEdge(3,1);
 g.addEdge(3,2);
 g.addEdge(4,3);
 g.addEdge(5,4);
 g.addEdge(6,8);
 g.addEdge(6,7);
 g.addEdge(7,5);
 g.addEdge(8,4);
 g.addEdge(8,6);
 g.addEdge(8,7);

 g.applyDFS();

 return 0;
}
```

خرج البرنامج :

```
Depth-first traversal for the given graph:

 new tree, root=1 , edeges : 12 23 24 15
 new tree, root=6 , edeges : 68 87

Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.
```

تمرين غير محلول:

طبق خوارزمية البحث بالعمق على البيان التالي :