جامعة المنارة

جامعة المنارة

كلية: الهندسة

قسم: الهندسة المعلوماتية

اسم المقرر: الخوارزميات و بنى المعطيات2

رقم الجلسة (الثالثة)

عنوان الجلسة

خوارزمية البحث بالعرض أولاً في البيان
(BFS) Breadth First Search

الفصل الدراسي الثاني

العام الدراسي 2023- 2024

الغاية من الجلسة

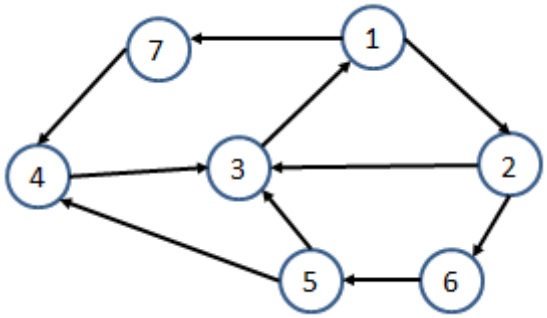
- ✓ تطبيق خوارزمية البحث بالعرض أولاً في البيان .

خوارزمية البحث بالعرض أولاً في البيان

- ➢ وتستخدم لزيارة رؤوس البيان بترتيب المسافات المتزايدة من نقطة البداية V (level by level) ، بحيث يتم التوجه إلى زيارة جميع الرؤوس التي تبعد مسافة d من الرأس V ،قبل زيارة أي رؤوس تبعد مسافة تساوي d+1 عن الرأس V .
- ➢ يتم تطبيق الخوارزمية برمجياً باستخدام الرتل .

تمرين:

طبق خوارزمية البحث بالعرض 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
0	0	0	0	0	0	0

و نبدأ باستدعاء تابع البحث بالعرض بدءاً من الرأس 1 : BFS(1) .

نقوم بزيارة رأس البداية 1 و نضيفه إلى الرتل الفارغ :

Queue

1							

visited

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

تعتمد الخوارزمية في كل تكرار على حذف رأس من مقدمة الرتل و طباعته و إضافة جميع رؤوس البيان المجاورة لذلك الرأس و التي لم تتم زيارتها بعد في نهاية الرتل ، و يستمر تكرار العملية حتى يصبح الرتل فارغاً :

Queue

2	7						

visited

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

Queue

7	3	6					

visited

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

Queue

3	6	4					

visited

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

Queue

6	4						

visited

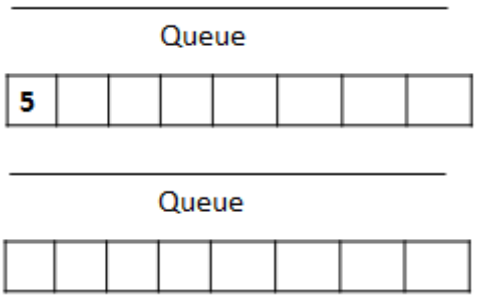
1	2	3	4	5	6	7
1	1	1	1	0	1	1

Queue

4	5						

visited

1	2	3	4	5	6	7
1	1	1	1	1	1	1



ويصبح ترتيب زيارة رؤوس البيان حسب خوارزمية البحث بالعرض كالتالي :

1,2,7,3,6,4,5

البرنامج الخاص بخوارزمية البحث بالعرض أولاً في بيان متصل :

```
// Program to print BFS traversal from a given vertex.
// traverses vertices reachable from s.
#include<iostream>
#include <list>

using namespace std;

// This class represents a directed graph using adjacency
list representation
class Graph
{
    int V;    // No. of vertices
    list<int> *adj;    // Pointer to an array containing
adjacency lists
public:
    Graph(int V);  // Constructor
    void addEdge(int v, int w); // function to add an edge
to graph
    void BFS(int s);  // prints BFS traversal from a given
source s
};

Graph::Graph(int V)
{
    this->V = V;
    adj = new list<int>[V+1];
}
```

```
void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to v's list.
}

void Graph::BFS(int s)
{
    // Mark all the vertices as not visited
    bool *visited = new bool[V+1];
    for(int i = 1; i < V+1; i++)
        visited[i] = false;

    // Create a queue for BFS
    list<int> queue;

    // Mark the current node as visited and enqueue it
    visited[s] = true;
    queue.push_back(s);

    // 'i' will be used to get all adjacent vertices of a
vertex
    list<int>::iterator i;

    while(!queue.empty())
    {
        // Dequeue a vertex from queue and print it
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        // Get all adjacent vertices of the dequeued
vertex s
        // If a adjacent has not been visited, then mark
it visited
        // and enqueue it
        for(i = adj[s].begin(); i != adj[s].end(); ++i)
        {
            if(!visited[*i])
            {
                visited[*i] = true;
                queue.push_back(*i);
            }
        }
```

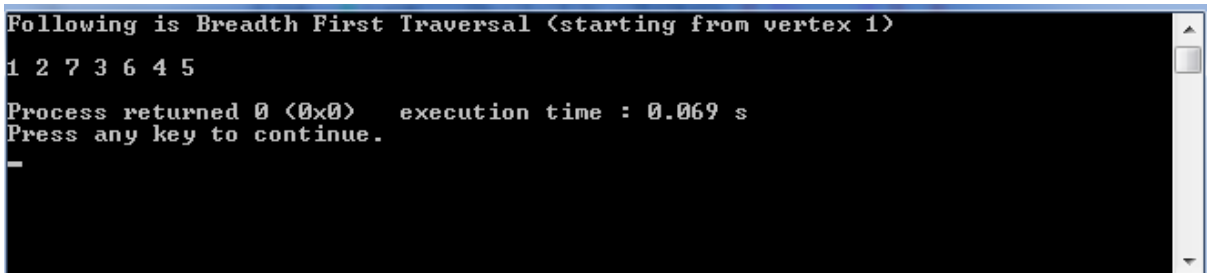
```
    }
}

// Driver program to test methods of graph class
int main()
{
    // Create a graph given in the above diagram
 Graph g(7);
 g.addEdge(1,2);
 g.addEdge(1,7);
 g.addEdge(2,3);
 g.addEdge(2,6);
 g.addEdge(3,1);
 g.addEdge(4,3);
 g.addEdge(5,3);
 g.addEdge(5,4);
 g.addEdge(6,5);
 g.addEdge(7,4);
    cout << "Following is Breadth First Traversal \n ";

    g.BFS(1);

    return 0;
}
```

خرج البرنامج :



تمرين غير محلول:

طبق خوارزمية البحث بالعرض على البيان التالي بحيث يتم البدء من الرأس v1 :