جامعة المنارة

كلية: الهندسة

قسم: الهندسة المعلوماتية

اسم المقرر: الخوارزميات و بنى المعطيات2

رقم الجلسة (السابعة)

عنوان الجلسة

# خوارزمية الترتيب الطوبولوجي في البيان
# (Topological sorting Algorithm )

الفصل الدراسي الثاني

العام الدراسي 2023- 2024

## الغاية من الجلسة

- ✓ تطبيق خوارزمية Kahn لإيجاد الترتيب الطوبولوجي للبيان .
- ✓ تنفيذ الكود الخاص بخوارزمية Kahn.

## خوارزمية Topological sorting في البيان الموجه و غيرالحلقي

- ✓ تستخدم الخوارزمية لزيارة رؤوس البيان الموجه بترتيب خطي يقتضي أنه من أجل كل حافة في البيان vw يتم المرور على الرأس v أولاً ثم على الرأس w .
- ✓ تطبق هذه الخوارزمية فقط على البيان الموجه وغير الحلقي (أي لا يحتوي على أية حلقة).
- ✓ تعد خوارزمية Kahn من أهم الخوارزميات التي تستخدم لإيجاد الترتيب الطوبولوجي للبيان .

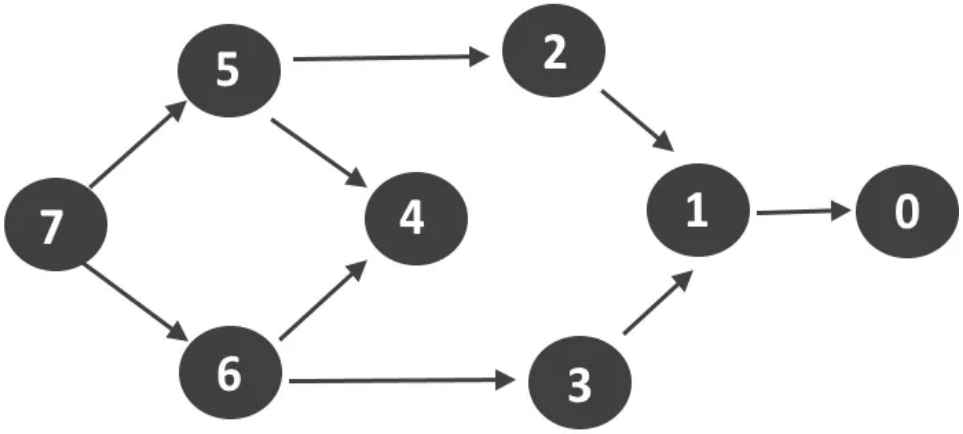
## خوارزمية Kahn لإيجاد الترتيب الطوبولوجي للبيان:

تستخدم في الخوارزمية المصفوفة in_degree التي تعبر عن عدد الحواف الداخلة إلى كل رأس في البيان ، و يستخدم أيضاً الرتل لإضافة الرؤوس التي يكون عدد الحواف الداخلة لها يساوي الصفر.

و يعبر عن الخوارزمية بالخطوات التالية:

1. يتم حساب عدد الحواف الداخلة لكل رأس موجود في البيان الموجه و تخزينها في المصفوفة in_degree .
2. اختيار كل رأس i يمتلك درجة دخول تساوي الصفر (in_degree[i]=0) وإضافته إلى الرتل .
3. حذف رأس v من مقدمة الرتل ثم تنفيذ ما يلي :
   a. إضافة الرأس v إلى مصفوفة الرؤوس المزارة output .
   b. يتم تعديل المصفوفة in_degree من خلال إنقاص عدد الحواف بمقدار 1 لكل الرؤوس المجاورة للرأس v.
   c. يتم اختبار القيم التي تم تعديلها في المصفوفة in_degree ، و كل رأس i يحقق in_degree[i]=0 يتم إضافته إلى الرتل .
4. يتم تكرار الخطوة 3 إلى أن يصبح الرتل فارغاً.

**مثال:** طبق خوارزمية الترتيب الطوبولوجي على البيان التالي :



✓ نهيئ المصفوفة in_degree :

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 1 | 1 | 2 | 1 | 1 | 0 |

✓ نضيف إلى الرتل كل رأس i يحقق in_degree[i]=0

Queue

| 7 | |
|---|---|

✓ يتم في كل تكرار حذف رأس من مقدمة الرتل وإضافته إلى مصفوفة الخرج، و تعديل قيم المصفوفة in_degree للرؤوس المجاورة لذلك الرأس،ثم اختبار الرؤوس في المصفوفة in_degree التي تم تعديلها بحيث يضاف إلى الرتل كل رأس يصبح عدد الحواف الداخلة إليه يساوي الصفر ،و يستمر التكرار حتى يصبح الرتل فارغاً .

Output

| 7 | | | | | | | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 1 | 1 | 2 | 0 | 0 | 0 |

Queue

| 5 | 6 | |
|---|---|---|

---

Output

| 7 | 5 | | | | | | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 0 | 1 | 1 | 0 | 0 | 0 |

Queue

| 6 | 2 | |
|---|---|---|

---

Output

| 7 | 5 | 6 | | | | | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue

| 2 | 3 | 4 | |
|---|---|---|---|

---

Output

| 7 | 5 | 6 | 2 | | | | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue

| 3 | 4 | |
|---|---|---|

---

Output

| 7 | 5 | 6 | 2 | 3 | | | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue

| 4 | 1 | |
|---|---|---|

---

Output

| 7 | 5 | 6 | 2 | 3 | 4 | | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue

| 1 | |
|---|---|

---

Output

| 7 | 5 | 6 | 2 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|---|

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue

| 0 | |
|---|---|

---

Output

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

| in_degree Array | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue

| |
|---|

---

أصبح الرتل فارغاً ، لذلك نوقف التكرار

البرنامج الخاص بخوارزمية Topological Sorting :

```
// A C++ program to print topological
// sorting of a graph using indegrees.
#include <iostream>
#include<list>
#include<queue>
using namespace std;

// Class to represent a graph
class Graph {
    // No. of vertices'
    int V;

    // Pointer to an array containing
    // adjacency listsList
    list<int>* adj;

public:
    // Constructor
    Graph(int V);
     // Function to add an edge to graph
    void addEdge(int u, int v);

    // prints a Topological Sort of
    // the complete graph
    void topologicalSort();
};

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

void Graph::addEdge(int u, int v)
{
    adj[u].push_back(v);
}

// The function to do  Topological Sort.
void Graph::topologicalSort()
{
    /* Create an array to store indegrees of all
     vertices. Initialize all indegrees as 0*/
    int * in_degree=new int[V];
    for(int i=0;i<V;i++)
```

```
        in_degree[i]=0;

    /* Traverse adjacency lists to fill indegrees of
     vertices.  This step takes O(V+E) time */
    for (int u = 0; u < V; u++) {
        list<int>::iterator itr;
        for (itr = adj[u].begin();itr != adj[u].end(); itr++)
            in_degree[*itr]++;
    }

    // Create an queue and enqueue
    // all vertices with indegree 0
    queue<int> q;
    for (int i = 0; i < V; i++)
        if (in_degree[i] == 0)
            q.push(i);

    // Initialize count of visited vertices
    int count = 0;
    /* Create an array to store
     result (A topological
     ordering of the vertices)*/
     int *output=new int[V];

    /* One by one dequeue vertices from queue and enqueue
     adjacents if indegree of adjacent becomes 0 */
    while (!q.empty()) {
        /* Extract front of queue (or perform dequeue)
         and add it to topological order */
        int u = q.front();
        q.pop();
        output[count]=u;

        /* Iterate through all its
         neighbouring nodes of dequeued node u and
         decrease their in-degree by 1 */
        list<int>::iterator itr;
        for (itr = adj[u].begin();itr != adj[u].end(); itr++)
            {

            // If in-degree becomes zero,
            // add it to queue
            in_degree[*itr]--;
            if (in_degree[*itr] == 0)
                q.push(*itr);
            }
```

```
            count++;
        }

        // Check if there was a cycle
        if (count != V) {
            cout << "There exists a cycle in the graph\n";
            return;
        }

        // Print topological order
        for (int i = 0; i < count; i++)
            cout << output[i] << " ";
        cout << endl;
}

// main program to test above functions
int main()
{
        /* Create a graph given in the
         above diagram */
        Graph g(8);

        g.addEdge(7,5);
        g.addEdge(7, 6);
        g.addEdge(6, 4);
        g.addEdge(6, 3);
        g.addEdge(5, 2);
        g.addEdge(5,4);
        g.addEdge(2,1);
        g.addEdge(3,1);
        g.addEdge(1,0);


        cout << "Following is a Topological Sort of graph\n";
        g.topologicalSort();

        return 0;
}
```

تمرين غير محلول:

طبق خوارزمية Topological Sorting على البيان التالي :