Big-O Blue

LECTURE 05

BREADTH-FIRST SEARCH ALGORITHM

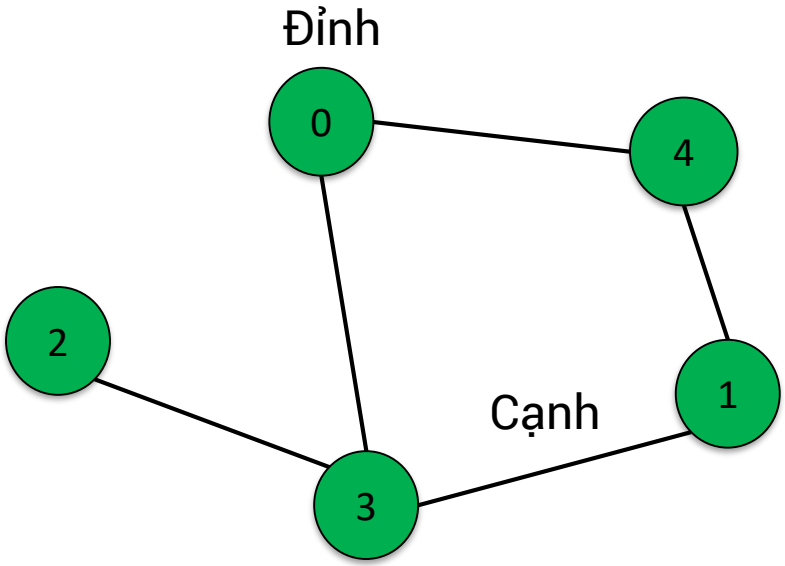
Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

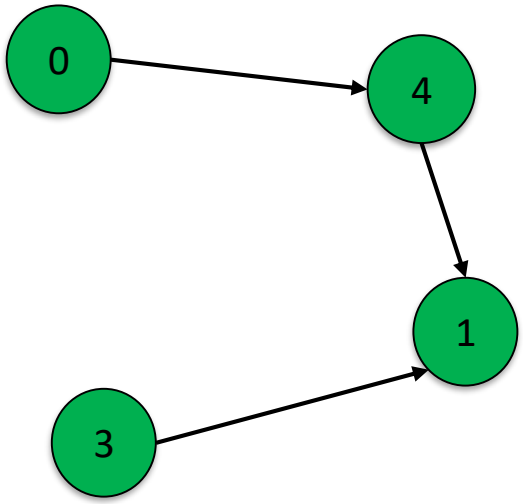
Giới thiệu sơ nét về đồ thị

Đồ thị là tập các đối tượng bao gồm các đỉnh (vertices) được nối với nhau bởi các cạnh (edges). Đồ thị có 2 dạng cơ bản thường gặp:

- Đồ thị có hướng (Directed graph)
- Đồ thị vô hướng (Undirected graph)



Đồ thị vô hướng



Đồ thị có hướng

Đồ thị trong thực tế có thể là các điểm, trạng thái trò chơi, công việc, môn học…

Thuật toán BFS

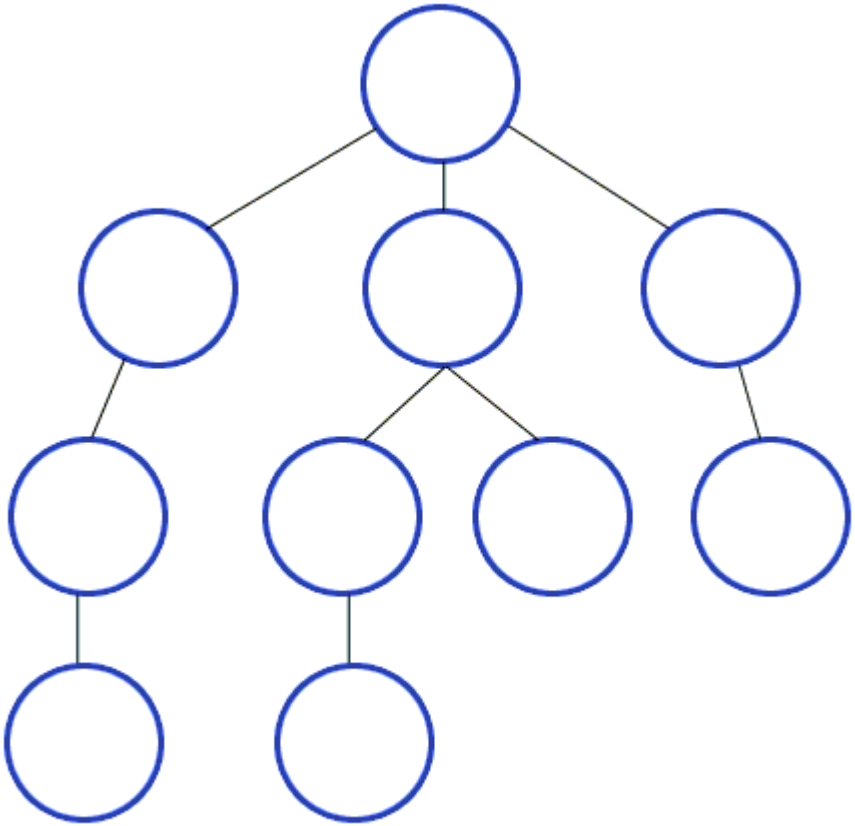
Breadth-first Search (Thuật toán tìm kiếm theo chiều rộng): là thuật toán tìm kiếm trên đồ thị **vô hướng** hoặc **có hướng**, **không** trọng số, nếu có trọng số thì trọng số các đường đi đều giống nhau, BFS giải quyết bài toán:

- Tìm kiếm đường đi từ một đỉnh bất kỳ tới tất cả các đỉnh khác trong đồ thị (nếu 2 đỉnh thuộc cùng thành phần liên thông với nhau).
- **Luôn tìm được đường đi ngắn nhất (nếu tồn tại đường đi).**

Độ phức tạp: O(V + E)

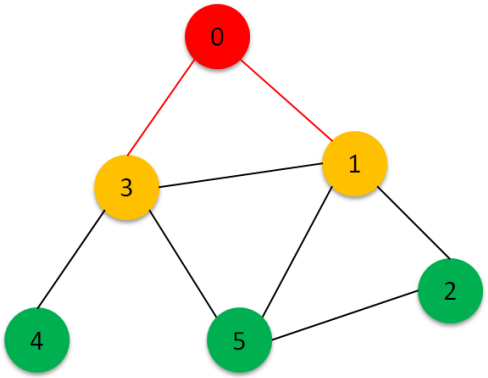
- V (Vertices): số lượng đỉnh của đồ thị.
- E (Edges): số lượng cạnh của đồ thị.

Mô phỏng cách chạy thuật toán

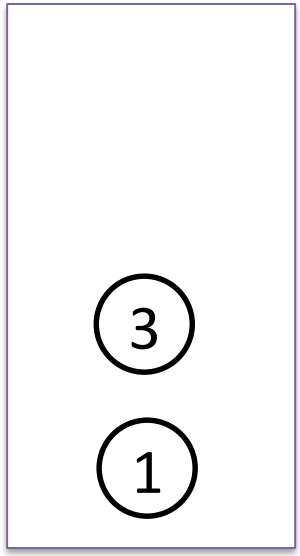


Ý tưởng thuật toán

Xuất phát từ 1 đỉnh bất kỳ, đi tới tất cả các đỉnh kề của đỉnh này và lưu các đỉnh kề này lại.



Lưu vào kho



Đem các đỉnh trong kho ra xét như bước đầu

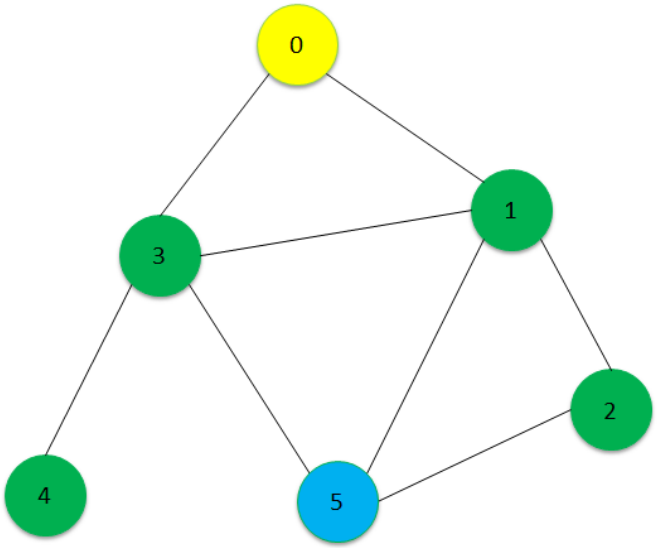
Lưu vết đường đi lại

Đỉnh	0	1	2	3
Lưu vết	-1	0	-1	0

Dừng thuật toán khi kho rỗng và in kết quả bài toán.

Bài toán minh họa

Cho đồ thị vô hướng như hình vẽ. Tìm **đường đi ngắn nhất** từ đỉnh **0** đến đỉnh **5**.



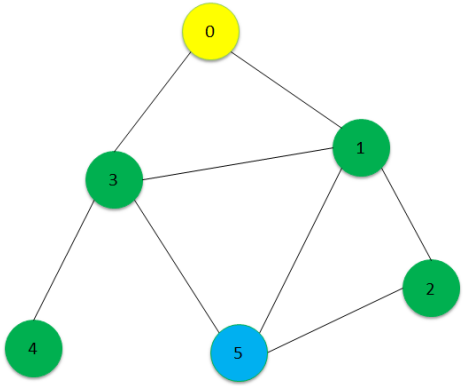
Adjacency Matrix

```
6
0 1 0 1 0 0
1 0 1 1 0 1
0 1 0 0 0 1
1 1 0 0 1 1
0 0 0 1 0 0
0 1 1 1 0 0
```

Edge List

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

Bước 0: Chuẩn bị dữ liệu



Chuyển danh sách cạnh kề vào **graph**.

Đỉnh	0	1	2	3	4	5
Đỉnh kề	1, 3	0, 2, 3, 5	1, 5	0, 1, 4, 5	3	1, 2, 3

Mảng đánh dấu các đỉnh đã xét **visited**.

Đỉnh	0	1	2	3	4	5
Trạng thái	false	false	false	false	false	false

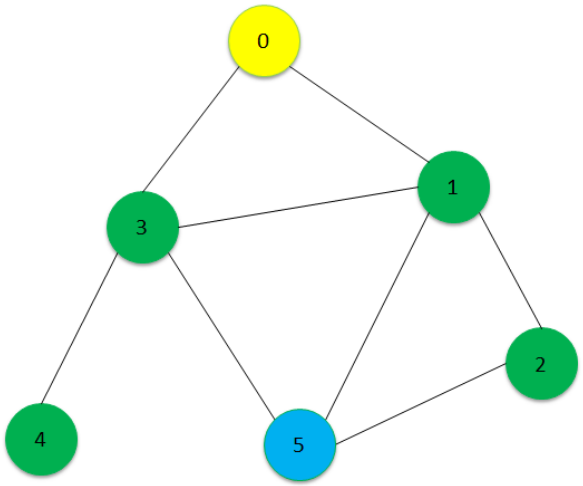
Mảng lưu vết đường đi **path**.

Đỉnh	0	1	2	3	4	5
Lưu vết	-1	-1	-1	-1	-1	-1

Tạo hàng đợi lưu các đỉnh đang xét **queue**. **...**

Bước 0: Chuẩn bị dữ liệu (tiếp theo)

Đỉnh 0 là đỉnh bắt đầu đi. Bỏ đỉnh 0 vào hàng đợi và đánh dấu đã xét đỉnh 0.



Mảng đánh dấu các đỉnh đã xét **visited**.

Đỉnh	**0**	1	2	3	4	5
Trạng thái	**true**	false	false	false	false	false

Hàng đợi lưu các đỉnh đang xét **queue**.

0

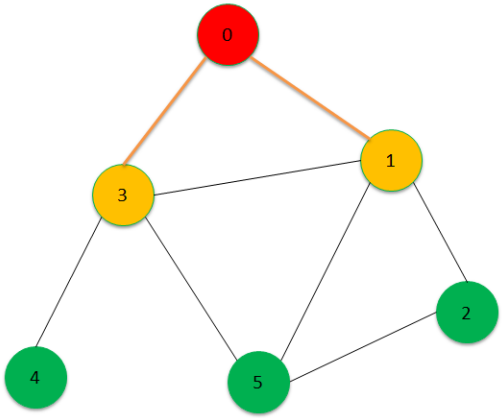
0

Bước 1: Chạy thuật toán lần 1

queue

0
0

Lấy **đỉnh 0** ra xét và tìm những đỉnh có kết nối với đỉnh 0 mà chưa được xét, bỏ vào queue.



graph

Đỉnh	**0**	1	2	3	4	5
Đỉnh kề	**1, 3**	0, 2, 3, 5	1, 5	0, 1, 4, 5	3	1, 2, 3

visited

Đỉnh	0	**1**	2	**3**	4	5
Trạng thái	true	**true**	false	**true**	false	false

queue

0	**1**
1	**3**

path

Đỉnh	0	**1**	2	**3**	4	5
Lưu vết	-1	**0**	-1	**0**	-1	-1

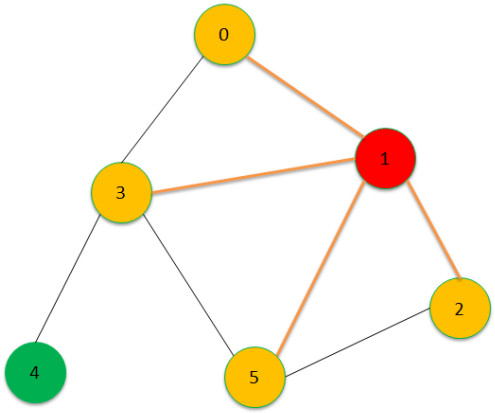
Bước 2: Chạy thuật toán lần 2

Big-O Blue

queue

0	1
1	3

Lấy **đỉnh 1** ra xét và tìm những đỉnh có kết nối với đỉnh 1 mà chưa được xét, bỏ vào queue.



graph

Đỉnh	0	**1**	2	3	4	5
Đỉnh kề	1, 3	**0, 2, 3, 5**	1, 5	0, 1, 4, 5	3	1, 2, 3

visited

Đỉnh	0	1	**2**	3	4	**5**
Trạng thái	true	true	**true**	true	false	**true**

queue

0	**1**	**2**
3	**2**	**5**

path

Đỉnh	0	1	**2**	3	4	**5**
Lưu vết	-1	0	**1**	0	-1	**1**

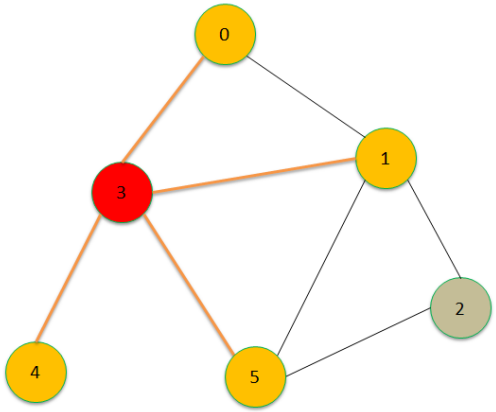
Bước 3: Chạy thuật toán lần 3

Big-O Blue

queue

0	1	2
3	2	5

Lấy **đỉnh 3** ra xét và tìm những đỉnh có kết nối với đỉnh 3 mà chưa được xét, bỏ vào queue.



graph

Đỉnh	0	1	2	**3**	4	5
Đỉnh kề	1, 3	0, 2, 3, 5	1, 5	**0, 1, 4, 5**	3	1, 2, 3

visited

Đỉnh	0	1	2	3	**4**	5
Trạng thái	true	true	true	true	**true**	true

queue

0	1	**2**
2	5	**4**

path

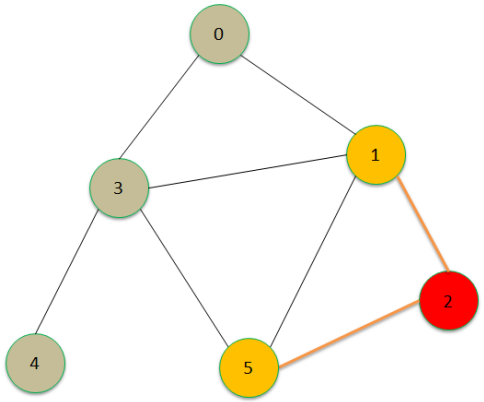
Đỉnh	0	1	2	3	**4**	5
Lưu vết	-1	0	1	0	**3**	1

Bước 4: Chạy thuật toán lần 4

queue

0	1	2
2	5	4

Lấy **đỉnh 2** ra xét và tìm những đỉnh có kết nối với đỉnh 2 mà chưa được xét, bỏ vào queue.



graph

Đỉnh	0	1	**2**	3	4	5
Đỉnh kề	1, 3	0, 2, 3, 5	**1, 5**	0, 1, 4, 5	3	1, 2, 3

visited

Đỉnh	0	1	2	3	4	5
Trạng thái	true	true	true	true	true	true

queue

0	1
5	4

path

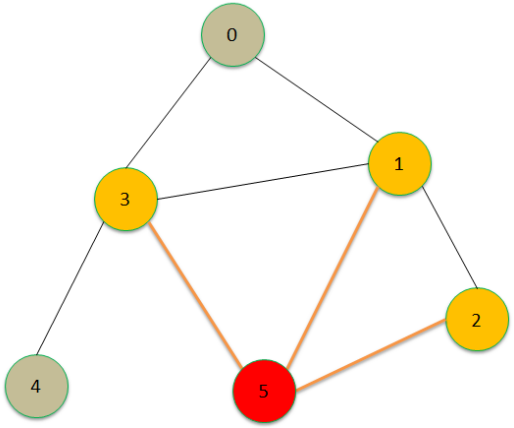
Đỉnh	0	1	2	3	4	5
Lưu vết	-1	0	1	0	3	1

Bước 5: Chạy thuật toán lần 5

queue

0	1
5	4

Lấy **đỉnh 5** ra xét và tìm những đỉnh có kết nối với đỉnh 5 mà chưa được xét, bỏ vào queue.



graph

Đỉnh	0	1	2	3	4	**5**
Đỉnh kề	1, 3	0, 2, 3, 5	1, 5	0, 1, 4, 5	3	**1, 2, 3**

visited

Đỉnh	0	1	2	3	4	5
Trạng thái	true	true	true	true	true	true

queue

0
4

path

Đỉnh	0	1	2	3	4	5
Lưu vết	-1	0	1	0	3	1

Bước 6: Chạy thuật toán lần 6

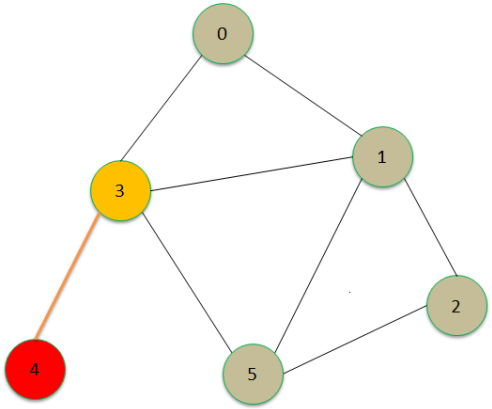
Big-O Blue

queue

0

4

Lấy **đỉnh 4** ra xét và tìm những đỉnh có kết nối với đỉnh 4 mà chưa được xét, bỏ vào queue.



graph

Đỉnh	0	1	2	3	**4**	5
Đỉnh kề	1, 3	0, 2, 3, 5	1, 5	0, 1, 4, 5	**3**	1, 2, 3

visited

Đỉnh	0	1	2	3	4	5
Trạng thái	true	true	true	true	true	true

queue

...

...

path

Đỉnh	0	1	2	3	4	5
Lưu vết	-1	0	1	0	3	1

Dừng thuật toán

Hàng đợi rỗng, tất cả các đỉnh đều được xét → dừng thuật toán.

path

Đỉnh	0	1	2	3	4	5
Lưu vết	-1	0	1	0	3	1

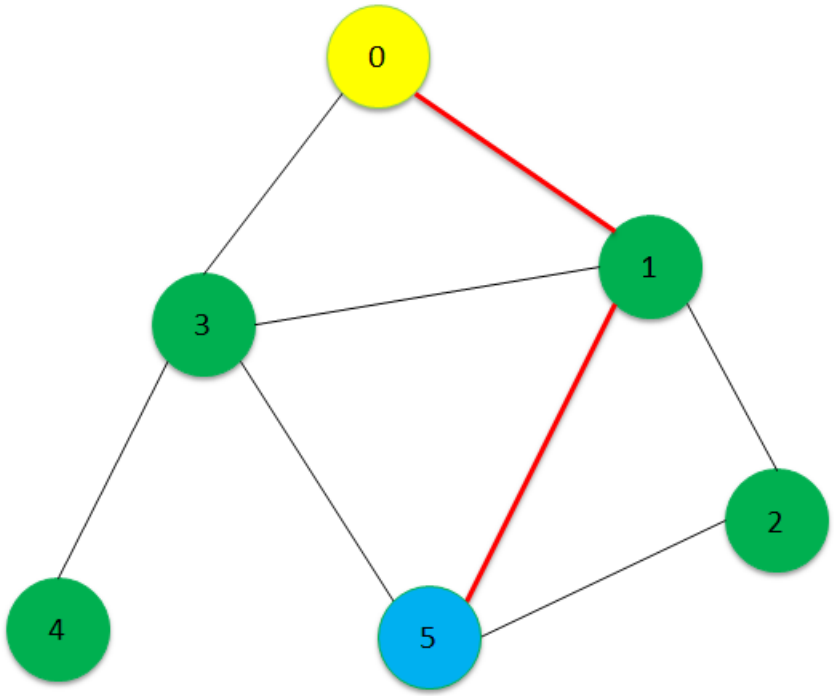
0 → 1 → 5

Thứ tự duyệt BFS là **0, 1, 3, 2, 5, 4**.

Đáp án bài toán

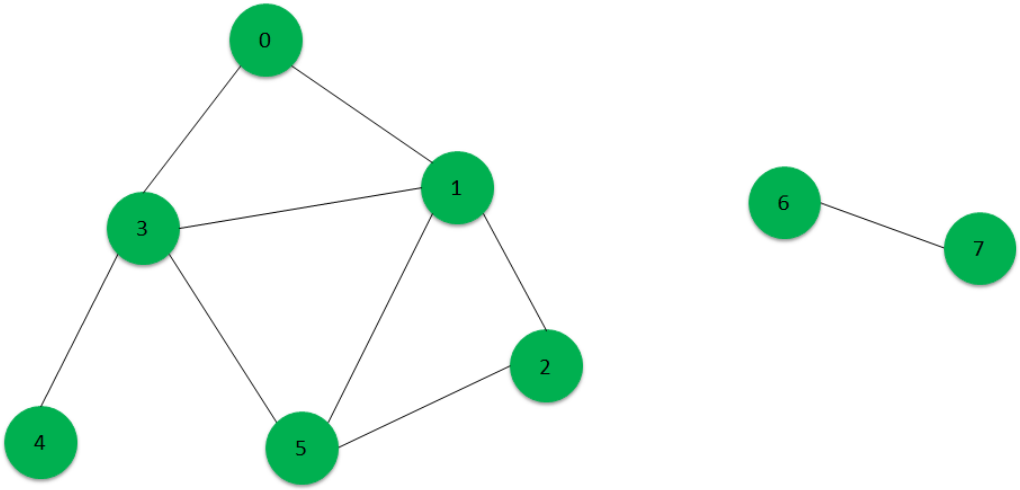
0 → 1 → 5

Đường đi ngắn nhất từ đỉnh **0** đến đỉnh **5** như hình vẽ.



Lưu ý khi sử dụng BFS

Khi 2 đỉnh cần tìm đường đi ngắn nhất nhưng lại không có đường đi tới nhau được thì kết quả trả về sẽ như thế nào?



Trường hợp chạy bắt đầu từ đỉnh 0.

path

Đỉnh	0	1	2	3	4	5	6	7
Lưu vết	-1	0	1	0	3	1	-1	-1

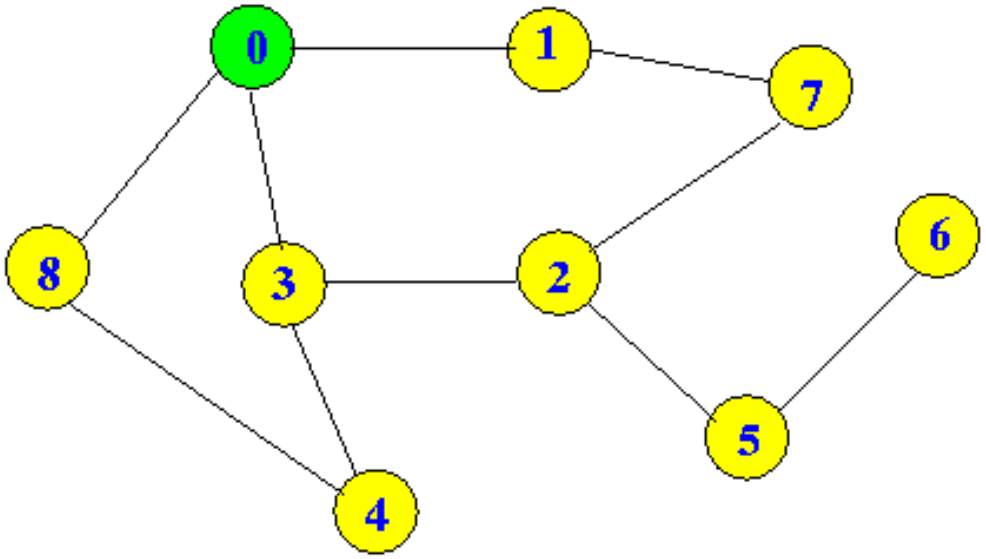
Trường hợp chạy bắt đầu từ đỉnh 6.

path

Đỉnh	0	1	2	3	4	5	6	7
Lưu vết	-1	-1	-1	-1	-1	-1	-1	6

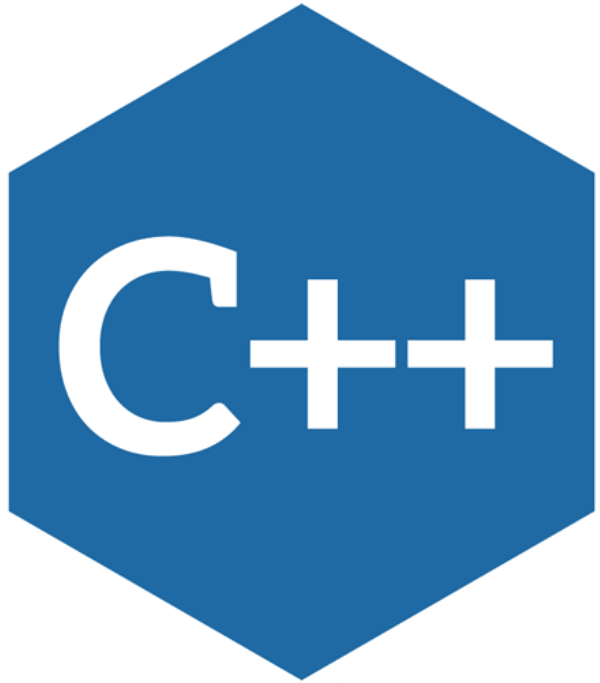
Bài tập luyện tập

Tìm đường đi ngắn nhất từ 0 đến 5 của đồ thị sau:



0 → 3 → 2 → 5

MÃ NGUỒN MINH HỌA BẰNG C++



Source Code BFS

Khai báo thư viện và các biến toàn cục:

```cpp
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
#define MAX 100
int V, E;
bool visited[MAX];
int path[MAX];
vector<int> graph[MAX];
```

Source Code BFS

Thuật toán chính BFS

```
10. void BFS(int s)
11. {
12.     for (int i = 0; i < V; i++)
13.     {
14.         visited[i] = false;
15.         path[i] = -1;
16.     }
17.     queue<int> q;
18.     visited[s] = true;
19.     q.push(s);
        // to be continued
```

Source Code BFS

```
20.     while (!q.empty())
21.     {
22.         int u = q.front();
23.         q.pop();
24.         for (int i = 0; i < graph[u].size(); i++)
25.         {
26.             int v = graph[u][i];
27.             if (!visited[v])
28.             {
29.                 visited[v] = true;
30.                 q.push(v);
31.                 path[v] = u;
32.             }
33.         }
34.     }
35. }
```

Source Code BFS

In đường đi từ mảng lưu vết (KHÔNG dùng đệ quy):

```
36. void printPath(int s, int f)
37. {
38.     int b[MAX];
39.     int m = 0;
40.     if (f == s)
41.     {
42.         cout << s;
43.         return;
44.     }
45.     if (path[f] == -1)
46.     {
47.         cout << "No path" << endl;
48.         return;
49.     }
    // to be continued
```

Source Code BFS

In đường đi từ mảng lưu vết (KHÔNG dùng đệ quy):

```
50.       while (true)
51.       {
52.           b[m++] = f;
53.           f = path[f];
54.           if (f == s)
55.           {
56.               b[m++] = s;
57.               break;
58.           }
59.       }
60.       for (int i = m - 1; i >= 0; i--)
61.           cout << b[i] << " ";
62.   }
```

Source Code BFS

In đường đi từ mảng lưu vết (dùng đệ quy):

```
63. void printPathRecursion(int s, int f)
64. {
65.     if (s == f)
66.         cout << f << " ";
67.     else
68.     {
69.         if (path[f] == -1)
70.             cout << "No path" << endl;
71.         else
72.         {
73.             printPathRecursion(s, path[f]);
74.             cout << f << " ";
75.         }
76.     }
77. }
```

Source Code BFS

Hàm main để gọi thực hiện:

```
78. int main()
79. {
80.     int u, v;
81.     cin >> V >> E;
82.     for (int i = 0; i < E; i++)
83.     {
84.         cin >> u >> v;
85.         graph[u].push_back(v);
86.         graph[v].push_back(u);
87.     }
88.     int s = 0;
89.     int f = 5;
90.     BFS(s);
91.     printPath(s, f);
92.     return 0;
93. }
```

MÃ NGUỒN MINH HỌA BẰNG PYTHON

Source Code BFS

Khai báo thư viện và các biến toàn cục:

```
1. from queue import Queue
2.
3. MAX = 100
4. V = None
5. E = None
6. visited = [False for i in range(MAX)]
7. path = [0 for i in range(MAX)]
8. graph = [[] for i in range(MAX)]
```

Source Code BFS

Thuật toán chính BFS

```
9.  def BFS(s):
10.     for i in range(V):
11.         visited[i] = False
12.         path[i] = -1
13.     q = Queue()
14.     visited[s] = True
15.     q.put(s)
16.     while not q.empty():
17.         u = q.get()
18.         for v in graph[u]:
19.             if not visited[v]:
20.                 visited[v] = True
21.                 q.put(v)
22.                 path[v] = u
```

Source Code BFS

In đường đi từ mảng lưu vết (KHÔNG dùng đệ quy):

```
23. def printPath(s, f):
24.     b = []
25.     if f == s:
26.         print(s)
27.         return
28.     if path[f] == -1:
29.         print("No path")
30.         return
31.     while True:
32.         b.append(f)
33.         f = path[f]
34.         if f == s:
35.             b.append(s)
36.             break
37.     for i in range(len(b)-1,-1,-1):
38.         print(b[i], end=' ')
```

Source Code BFS

In đường đi từ mảng lưu vết (dùng đệ quy):

```
39. def printPathRecursion(s, f):
40.     if s == f:
41.         print(f, end=' ')
42.     else:
43.         if path[f] == -1:
44.             print("No path")
45.         else:
46.             printPathRecursion(s, path[f])
47.             print(f, end=' ')
```

Source Code BFS

Hàm main để gọi thực hiện:

```
48. if __name__ == '__main__':
49.     V, E = map(int, input().split())
50.     for i in range(E):
51.         u, v = map(int, input().split())
52.         graph[u].append(v)
53.         graph[v].append(u)
54.     s = 0
55.     f = 5
56.     BFS(s)
57.     printPath(s, f)
```

MÃ NGUỒN MINH HỌA BẰNG JAVA

Source Code BFS

Khai báo thư viện:

```
1. import java.util.Scanner;
2. import java.util.ArrayList;
3. import java.util.LinkedList;
4. import java.util.Queue;
```

Khai báo biến toàn cục (thuộc class Main)

```
1. public class Main {
2.     private static ArrayList<ArrayList<Integer>> graph;
3.     private static int V;
4.     private static int E;
5.     private static ArrayList<Integer> path;
6.     private static ArrayList<Boolean> visited;
```

Source Code BFS

Thuật toán chính BFS (part 1)

```
1.      private static void BFS(int s) {
2.          Queue<Integer> q = new LinkedList<>();
3.          path = new ArrayList<>();
4.          visited = new ArrayList<>();
5.          for (int i = 0; i < V; i++) {
6.              visited.add(false);
7.              path.add(-1);
8.          }
9.          visited.set(s, true);
10.         q.add(s);
11.         // to be continued
```

Source Code BFS

Thuật toán chính BFS (part 2)

```
1.          while (!q.isEmpty()) {
2.              int u = q.remove();
3.              for (int i = 0; i < graph.get(u).size(); i++) {
4.                  int v = graph.get(u).get(i);
5.                  if (!visited.get(v)) {
6.                      visited.set(v, true);
7.                      path.set(v, u);
8.                      q.add(v);
9.                  }
10.             }
11.         }
12.     }
```

Source Code BFS

In đường đi từ mảng lưu vết (KHÔNG dùng đệ quy):

```
1.      private static void printPath(int s, int f) {
2.          if (s == f) {
3.              System.out.print(s);
4.              return;
5.          }
6.          if (path.get(f) == -1) {
7.              System.out.print("No path");
8.              return;
9.          }
10.         // to be continued
```

Source Code BFS

In đường đi từ mảng lưu vết (KHÔNG dùng đệ quy):

```
1.          ArrayList<Integer> b = new ArrayList<>();
2.          while (true) {
3.              b.add(f);
4.              f = path.get(f);
5.              if (s == f) {
6.                  b.add(f);
7.                  break;
8.              }
9.          }
10.         for (int i = b.size() - 1; i >= 0; i--) {
11.             System.out.printf("%d ", b.get(i));
12.         }
13.     }
```

Source Code BFS

In đường đi từ mảng lưu vết (dùng đệ quy):

```
1.      private static void printPathRecursion(int s, int f) {
2.          if (s == f)
3.              System.out.print(f + " ");
4.          else {
5.              if (path.get(f) == -1)
6.                  System.out.println("No path");
7.              else {
8.                  printPathRecursion(s, path.get(f));
9.                  System.out.printf("%d ", f);
10.             }
11.         }
12.     }
```

Source Code BFS

Hàm main để gọi thực hiện:

```
1.      public static void main(String[] args) {
2.          Scanner sc = new Scanner(System.in);
3.          V = sc.nextInt();
4.          E = sc.nextInt();
5.          graph = new ArrayList<>(V);
6.          for (int i = 0; i < V; i++)
7.              graph.add(new ArrayList<>());
8.          for (int i = 0; i < E; i++) {
9.              int u = sc.nextInt();
10.             int v = sc.nextInt();
11.             graph.get(u).add(v);
12.             graph.get(v).add(u);
13.         }
14.         int s = 0, f = 5;
15.         BFS(s);
16.         printPath(s, f);
17.     }
18. }
```

Hỏi đáp