

CÁCH ĐÁNH GIÁ ĐIỂM THỰC HÀNH

HỌC PHẦN: IT3150 – Project 1- 2023.1

I. Quy định, yêu cầu:

- Tài liệu và nội dung thực hành chấm điểm trên hệ thống:
<https://lab.soict.hust.edu.vn/>
- Bài tập trên lớp chấm điểm tự động (các bài không chấm trên hệ thống làm vào máy tính → làm báo cáo thực hành – Theo mẫu).
- Hạn nộp báo cáo trên Teams (Bài tập trên lớp + Bài tập về nhà): 1 tuần.

II. Đánh giá điểm thực hành

- Chuyên cần (đúng giờ, nghiêm túc trong giờ học) - Điểm danh trên Teams: 10%
- Báo cáo thực hành (bài tập trên lớp + Về nhà) theo mẫu nộp trên Teams: 40%
- Trắc nghiệm – Form trên Teams: 10%
- Kiểm tra thực hành: 40%. (Tiết 2,3 buổi thực hành thứ 5).

Điểm thưởng: 5% → 10% (Cho Mục 1,2 điểm TB từ 9-10).

Tham gia thực hành đúng giờ đầy đủ theo thời khóa biểu (nếu có lý do không đi thực hành đúng kíp được thì gửi mail xin phép thực hành bù trước 1 ngày qua mail hoalt@soict.hust.edu.vn, Tiêu đề: đăng ký học bù – IT3040 – MaLopTH.

Các kíp có thể bù:

TT	Thời gian, địa điểm, Tuần học	Mã nhóm	Mã lớp
1			
2			
3			
4			
5			
6			
7			

Nếu nghỉ không có lý do 3 buổi, không thực hành bù thì điểm chuyên cần, báo cáo và BTVN coi như 0 điểm thực hành.

Table of Contents

Bài toán: Cây khung tối thiểu - Kruskal.....	3
Vấn đề: Liệt kê thứ tự các nút được DFS truy cập.....	7
Vấn đề: Trình tự các nút được BFS truy cập.....	10
Vấn đề: Chu trình Hamiton	13

Table of figures

Figure 1 code bài 1	3
Figure 2 test bài 1	6
Figure 3 code bài 2	8
Figure 4 test bài 2	9
Figure 5 code bài 3	10
Figure 6 test bài 3	12
Figure 7 code bài 4	14
Figure 8 test bài 4	17

Bài toán: Cây khung tối thiểu - Kruskal

Cho đồ thị liên thông vô hướng $G=(V,E)$ trong đó $V=\{1,...,N\}$. Mỗi cạnh $(u,v) \in E$ có trọng số $w(u,v)$. Tính cây khung nhỏ nhất của G .

Đầu vào

- Dòng 1: N và M ($1 < N, M < 105$) trong đó NN là số nút và MM là số cạnh.
- Dòng $i+1$ ($i=1,...,M$): chứa 3 số nguyên dương u, v, w trong đó w là trọng số của cạnh (u,v)

Đầu ra

Viết trọng số của cây bao trùm tối thiểu được tìm thấy.

Figure 1 code bài 1

Source code

```
1 //Mai Minh Hoàng
2 //20215381
3 #include <iostream>
4 #include <vector>
5 #include <algorithm>
6 #include <map>
7 using namespace std;
8
9 class Edge_81 { // Định nghĩa lớp Edge_81
10 public:
11     int u_81;
12     int v_81;
13     int w_81;
14     Edge_81(int u, int v, int w) : u_81(u), v_81(v), w_81(w) {}
15 };
16
17 int v_81; // Số lượng đỉnh
18 vector<Edge_81> edges_81; // Danh sách các cạnh
19 vector<int> parent_81; // Sử dụng vector để lưu trữ các nút cha
20
21 bool comparator_81(const Edge_81& obj1, const Edge_81& obj2) { // Hàm so sánh hai
22     return obj1.w_81 < obj2.w_81;
23 }
24
```

```

25 int findRoot_81(int node) { // Tìm gốc của cây chứa nút
26     if (parent_81[node] == -1) {
27         return node; // Nếu nút là gốc của cây
28     }
29     parent_81[node] = findRoot_81(parent_81[node]); // Nén đường đi: cập nhật par
30     return parent_81[node];
31 }
32
33 void mergeTree_81(int root1, int root2) { // Hợp nhất hai cây
34     if (root1 != root2) {
35         parent_81[root1] = root2; // Hợp nhất cây root1 vào cây root2
36     }
37 }
38
39 int main() {
40     ios_base::sync_with_stdio(0);
41     cin.tie(0);
42     cout.tie(0);
43
44     int e;
45     cin >> v_81 >> e;
46     parent_81.assign(v_81 + 1, -1); // Khởi tạo vector parent
47
48     for (int i = 0; i < e; i++) {
49         int a, b, c;
50         cin >> a >> b >> c;
51         edges_81.emplace_back(a, b, c); // Thêm cạnh vào danh sách
52     }
53     sort(edges_81.begin(), edges_81.end(), comparator_81); // Sắp xếp danh sách c
54
55     int sum = 0;
56     for (const Edge_81& edge : edges_81) { // Duyệt qua từng cạnh
57         int pos_u = findRoot_81(edge.u_81); // Tìm gốc của cây chứa u
58         int pos_v = findRoot_81(edge.v_81); // Tìm gốc của cây chứa v
59         if (pos_u != pos_v) { // Nếu u và v không thuộc cùng một cây
60             mergeTree_81(pos_u, pos_v); // Hợp nhất hai cây lại
61             sum += edge.w_81; // Cộng trọng số của cạnh vào tổng trọng số
62         }
63     }
64
65     cout << sum; // In ra tổng trọng số
66     return 0;
67 }

```

Figure 2 test bài 1

Input

```
550 1280
528 1 258
384 2 511
455 3 358
296 4 654
548 5 468
66 6 349
533 7 394
93 8 288
336 9 266
201 10 739
215 11 825
400 12 954
285 13 697
309 14 921
312 15 924
119 16 19
```



```
555 585 400
312 374 928
478 268 648
173 351 381
522 448 796
489 214 232
22 38 993
115 431 319
275 419 112
28 487 955
240 59 18
222 437 917
351 460 772
327 478 928
```

Correct output

146379



User output

146379



Input

```
10000 16782
2195 1 4096
4385 2 3381
5946 3 5081
649 4 2606
3860 5 3957
6357 6 2906
16 7 6090
7018 8 3138
94 9 3757
8331 10 116
369 11 4125
5595 12 7769
7158 13 4800
8541 14 2566
4300 15 4809
4450 16 15
```

```
398 6549 5974
7206 3952 5222
883 5418 1830
5967 4170 2859
2366 6386 2085
6305 6198 7331
414 4065 404
192 9860 627
4391 5026 3433
125 5902 5133
2887 1091 3401
1609 4722 1262
```

Correct output

26374598

User output

26374598

Vấn đề: Liệt kê thứ tự các nút được DFS truy cập

Cho đồ thị vô hướng $G=(V,E)$ trong đó $V = \{1,2,...,n\}$ là tập hợp các nút. Viết chương trình truy cập các nút của G bằng DFS (xem xét thứ tự từ điển của các nút).

Đầu vào

- Dòng 1: chứa 2 số nguyên n và m ($1 \leq n, m \leq 100000$)
- Dòng i+1: chứa u và v là hai điểm cuối của cạnh thứ i

đầu ra

Trình tự các nút được DFS truy cập

Figure 3 code bài 2

Source code

```
1 //Mai Minh Hoàng
2 //20215381
3 #include <iostream>
4 #include <vector>
5 using namespace std;
6
7 const int MAXN_81 = 100000; // Định nghĩa hằng số MAXN_81
8
9 vector<int> adj_81[MAXN_81]; // Khai báo mảng vector adj_81 để lưu danh sách kề
10 bool visited_81[MAXN_81]; // Khai báo mảng visited_81 để đánh dấu các đỉnh đã duyệt
11
12 void DFS_81(int u) { // Hàm DFS_81 để duyệt đồ thị theo chiều sâu
13     visited_81[u] = true; // Đánh dấu đỉnh u đã được duyệt
14     cout << u << " "; // In ra đỉnh u
15
16     for (int v : adj_81[u]) { // Duyệt qua các đỉnh kề với u
17         if (!visited_81[v]) { // Nếu đỉnh v chưa được duyệt
18             DFS_81(v); // Gọi hàm DFS_81 cho đỉnh v
19         }
20     }
21 }
22
23 int main() { // Hàm main - điểm bắt đầu của chương trình
24     int n, m; // Khai báo số lượng đỉnh và số lượng cạnh
25     cin >> n >> m; // Nhập số lượng đỉnh và số lượng cạnh
26 }
```



```

22
23 int main() { // Hàm main - điểm bắt đầu của chương trình
24     int n, m; // Khai báo số lượng đỉnh và số lượng cạnh
25     cin >> n >> m; // Nhập số lượng đỉnh và số lượng cạnh
26
27     // Khởi tạo danh sách kề
28     for (int i = 0; i < m; i++) { // Duyệt qua từng cạnh
29         int u, v;
30         cin >> u >> v; // Nhập hai đỉnh u, v của cạnh
31         adj_81[u].push_back(v);
32         adj_81[v].push_back(u); // Vì đồ thị là vô hướng, nên cần thêm cả hai chi
33     }
34
35     // Khởi tạo mảng visited
36     for (int i = 1; i <= n; i++) {
37         visited_81[i] = false; // Ban đầu, tất cả các đỉnh đều chưa được duyệt qua
38     }
39
40     // Bắt đầu duyệt DFS từ một nút bất kỳ
41     for (int i = 1; i <= n; i++) {
42         if (!visited_81[i]) { // Nếu đỉnh i chưa được duyệt qua
43             DFS_81(i); // Gọi hàm DFS_81 cho đỉnh i
44         }
45     }
46
47     return 0;
48 }

```

Figure 4 test bài 2

Input

```

7 12
1 2
1 3
2 3
2 4
2 7
3 5
3 7
4 5
4 6
4 7
5 6
5 7

```

Correct output

1 2 3 5 4 6 7

User output

1 2 3 5 4 6 7

Vấn đề: Trình tự các nút được BFS truy cập

Cho đồ thị vô hướng $G = (V, E)$ trong đó $V = \{1, 2, \dots, n\}$ là tập hợp các nút và E là tập hợp m cạnh.

Viết chương trình tính toán chuỗi các nút được truy cập bằng thuật toán BFS (các nút được xem xét theo thứ tự từ điển)

Đầu vào

- Dòng 1: chứa 2 số nguyên n và m là số nút và số cạnh
- Dòng $i+1$ ($i = 1, \dots, m$): chứa 2 số nguyên dương u và v là điểm cuối của cạnh thứ i

Đầu ra

Viết chuỗi các nút được thủ tục BFS truy cập (các nút a được phân tách bằng ký tự SPACE)

Figure 5 code bài 3

Source code

```

1 //Mai Minh Hoàng
2 //20215381
3 #include <iostream>
4 #include <set>
5 #include <queue>
6 using namespace std;
7
8 int done_81[10000]; // Mảng đánh dấu các đỉnh đã duyệt qua
9 set<int> list_81[1000]; // Mảng các set lưu danh sách kề
10 queue<int> queu_81[1000]; // Mảng các queue lưu các đỉnh cần duyệt
11 int u_81; // Số lượng đỉnh
12 int minn_81 = 999; // Biến lưu giá trị nhỏ nhất
13
14 int check_81() { // Hàm kiểm tra xem còn đỉnh nào chưa duyệt không
15     for (int i = 1; i <= u_81; i++) {
16         if (done_81[i] == 0) {
17             return i;
18         }
19     }
20     return -1;
21 }
22
23 void BFS_81(int n) { // Hàm BFS_81 để duyệt đồ thị theo chiều rộng
24     while (!queu_81[n].empty()) { // Trong khi queue của n không rỗng
25         int element = queu_81[n].front(); // Lấy phần tử đầu tiên trong queue
26         cout << element << " "; // In ra phần tử đó

```

```

27
28     for (const int& j : list_81[element]) { // Duyệt qua các đỉnh kề với element
29         if (done_81[j] == 0) { // Nếu đỉnh j chưa được duyệt qua
30             queu_81[n + 1].push(j); // Đẩy j vào queue của n+1
31             done_81[j] = 1; // Đánh dấu j đã được duyệt qua
32         }
33     }
34     queu_81[n].pop(); // Loại bỏ phần tử đầu tiên khỏi queue của n
35 }
36
37 if (queu_81[n + 1].empty()) { // Nếu queue của n+1 rỗng
38     if (check_81() != -1) { // Nếu còn đỉnh chưa được duyệt qua
39         queu_81[n + 1].push(check_81()); // Đẩy đỉnh chưa được duyệt vào queue của n+1
40         done_81[check_81()] = 1; // Đánh dấu đỉnh đã được duyệt qua
41     }
42     else {
43         return; // Kết thúc khi không còn đỉnh nào để duyệt
44     }
45 }
46
47 BFS_81(n + 1); // Gọi hàm BFS cho n+1
48 }
49
50 int main() { // Hàm main - điểm bắt đầu của chương trình
51     int v_81;
52     cin >> u_81 >> v_81; // Nhập số lượng đỉnh và số lượng cạnh
53

```

```

50 int main() { // Hàm main - điểm bắt đầu của chương trình
51     int v_81;
52     cin >> u_81 >> v_81; // Nhập số lượng đỉnh và số lượng cạnh
53
54     for (int i = 0; i < v_81; i++) { // Duyệt qua từng cạnh
55         int a, b;
56         cin >> a >> b; // Nhập hai đỉnh a, b của cạnh
57         list_81[a].insert(b);
58         list_81[b].insert(a); // Vì đồ thị là vô hướng, nên cần thêm cả hai chiều a->b và b->a vào danh sách kề
59
60         if (a < minn_81) {
61             minn_81 = a;
62         }
63     }
64
65     done_81[1] = 1;
66     queu_81[1].push(1);
67
68     BFS_81(1);
69
70     return 0;
71 }

```

Test:

Figure 6 test bài 3

Input

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



Correct output

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



User output

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



Input

```
16 16
2 4
1 3
3 4
5 6
1 2
3 5
2 3
7 2
6 9
9 10
4 10
5 11
8 12
12 14
13 15
13 16
```



Correct output

```
1 2 3 4 7 5 10 6 11 9 8 12 14 13 15 16
```



User output

```
1 2 3 4 7 5 10 6 11 9 8 12 14 13 15 16
```



Vấn đề: Chu trình Hamiton

Cho đồ thị vô hướng $G = (V, E)$. Viết chương trình kiểm tra xem G có phải là đồ thị Hamilton hay không.

Đầu vào

- Dòng 1: số nguyên dương T (số đồ thị)
- Các dòng tiếp theo là thông tin về đồ thị T , mỗi dòng có định dạng sau:
 - Dòng 1: n và m (số nút và cạnh)
 - Dòng $i+1$ ($i = 1, 2, \dots, m$): u và v : hai điểm cuối của cạnh thứ i

đầu ra

- trong mỗi dòng, viết 1 nếu tương ứng là đồ thị Hamilton và viết 0, nếu không

Figure 7 code bài 4

Source code

```

1 //Mai Minh Hoàng
2 //20215381
3 #include <iostream>
4 #include <sstream>
5 #include <stack>
6 #include <queue>
7 #define MAX 100 // Định nghĩa hằng số MAX
8 using namespace std; // Sử dụng không gian tên chuẩn
9
10 int canh_81[MAX][2]; // Mảng lưu thông tin các cạnh
11 int duyet_81[MAX]; // Mảng đánh dấu các đỉnh đã duyệt qua
12 queue<int> queuvu_81;
13 queue<int> queu_81[MAX]; // Mảng các queue lưu các đỉnh cần duyệt
14 int target_81;
15 int value_81;
16
17 void searchHamiton_81(int n, int v) { // Hàm kiểm tra đồ thị có phải là đồ thị Hai
18     int status = 1;
19     for (int i = 1; i <= v; i++) {
20         if (duyet_81[i] == 0) {
21             status = 0;
22         }
23     }
24     if (status == 1 && n == target_81) {
25         value_81 = 1;
26     }

```

```

27     queue<int> clone;
28     while (!queu_81[n].empty()) {
29         int j = queu_81[n].front();
30         clone.push(j);
31         if (duyet_81[j] == 0 || j == target_81) {
32             duyet_81[j] = 1;
33             searchHamiton_81(j, v);
34             duyet_81[j] = 0;
35         }
36         queu_81[n].pop();
37     }
38     queu_81[n] = clone;
39
40 }
41
42 int main() { // Hàm main
43     int n;
44     string s;
45     getline(cin, s);
46     stringstream ss(s);
47     ss >> n;
48     for (int j = 0; j < n; j++) { //nhập đầu vào
49         string s2;
50         getline(cin, s2);
51         stringstream ss2(s2);
52         int v = 0;
53         int e = 0;
54         ss2 >> v >> e;

```

```

54     ss2 >> v >> e;
55     for (int i = 0; i < e; i++) {
56         string s1;
57         int a, b;
58         getline(cin, s1);
59         stringstream ss1(s1);
60         ss1 >> canh_81[i][0] >> canh_81[i][1];
61         queu_81[canh_81[i][0]].push(canh_81[i][1]); //đổi thị vô hướng nên thêm
62         queu_81[canh_81[i][1]].push(canh_81[i][0]);
63     }
64
65
66     target_81 = 1;
67     duyet_81[1] = 1;
68     searchHamiton_81(target_81, v);
69     cout << value_81 << "\n";
70     value_81 = 0;
71     for (int i = 1; i <= v; i++) { //reset trạng thái của tất cả nút để sang t
72         queu_81[i] = queue<int>();
73         duyet_81[i] = 0;
74     }
75 }
76
77 return 0;
78
79 }

```


Figure 8 test bài 4

Input

```
2
5 5
1 2
1 3
2 4
2 5
3 5
7 13
1 3
1 5
1 7
2 4
2 5
2 6
3 4
3 5
3 7
4 6
```

```
2 5
2 6
3 4
3 5
3 7
4 6
4 7
5 7
6 7
```

Correct output

```
0
1
```

User output

```
0
1
```