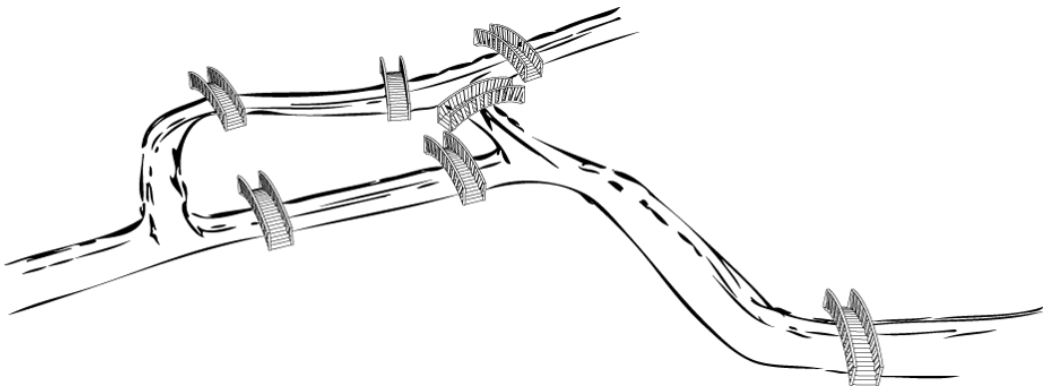
ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

LÝ THUYẾT ĐỒ THỊ

BÀI TẬP THỰC HÀNH



Biên soạn: Tôn Quang Toại

THÁNG 9, 2018

NỘI DUNG

MỘT SỐ CẤU TRÚC DỮ LIỆU THƯỜNG DÙNG TRONG GRAPH

(THAM KHẢO TRONG QUÁ TRÌNH THỰC HÀNH)

Trong phần này, chúng ta sẽ được giới thiệu một số cấu trúc dữ liệu thường được sử dụng để cài đặt hiệu quả các thuật toán trong Lý thuyết đồ thị như

- Mảng hai chiều, LinkedList (Danh sách liên kết), Tuple
- Stack
- Queue
- SortedSet

Để cài đặt nhanh (và hiệu quả) các thuật toán trong Lý thuyết đồ thị, chúng ta cần nắm vững một số cấu trúc dữ liệu có sẵn trong ngôn ngữ C#. Chẳng hạn như để biểu diễn đồ thị trên máy tính chúng ta cần biết cách sử dụng mảng 2 chiều, LinkedList và Tuple; Để cài đặt được thuật toán tìm kiếm theo chiều rộng (Breadth First Search – BFS) chúng ta cần biết cách sử dụng Queue; Để cài thuật toán Dijkstra hiệu quả chúng ta cần biết cách sử dụng SortedSet, …

Các collection thường được chia làm hai loại: Collection tuyến tính và Collection không tuyến tính.

Collection tuyến tính

- Mảng 2 chiều
- LinkedList, List
- Tuple
- Stack
- Queue

Collection không tuyến tính

- SortedSet

A. ***LinkedList:*** LinkedList là một cấu trúc dữ liệu gồm một dãy các phần tử (tuyến tính) và cho phép chúng ta thực hiện nhanh các thao tác **thêm, xóa** phần tử trong thời gian $O(1)$.

Nhận xét: Chúng ta thường dùng LinkedList để tạo danh sách kề, danh sách cạnh trong lý thuyết đồ thị.

1. Tạo LinkedList

```
LinkedList<int> l = new LinkedList<int>();
l.AddLast(4);
l.AddLast(7);
l.AddLast(3);
l.AddLast(4);


foreach(int x in l)
  Console.Write(x + " ");
```

2. Một số properties và methods thường dùng

Properties

Property	Ý nghĩa
First	Lấy phần tử đầu trong danh sách
Last	Lấy phần tử cuối trong danh sách
Count	Số lượng phần tử trong danh sách

Methods

Method	Ý nghĩa
AddFirst(giá trị)	Thêm giá trị vào đầu danh sách
AddLast(giá trị)	Thêm giá trị vào cuối danh sách
Contains(giá trị)	Kiểm tra có giá trị trong danh sách không
Clear()	Xóa mọi phần tử trong danh sách

B. **List:** List cơ bản là một mảng có thể truy cập bằng chỉ mục (index).

Nhận xét: Chúng ta thường dùng List hiệu quả khi biết trước kích thước của danh sách, và trong lý thuyết đồ thị thường được dùng để tạo danh sách cạnh khi cài đặt thuật toán Kruskal.

1. Tạo List

```
List<int> l = new List<int>(MAX); // MAX là khả năng của List
l.Add(4);
l.Add(7);
l.Add(3);
l.Add(4);


foreach(int x in l)
  Console.Write(x + " ");
```

2. Một số properties và methods thường dùng

Properties

Property	Ý nghĩa
Count	Số lượng phần tử trong danh sách

Methods

Method	Ý nghĩa
Add(giá trị)	Thêm giá trị vào cuối danh sách
Sort()	Sắp xếp danh sách tăng dần
Contains(giá trị)	Kiểm tra có giá trị trong danh sách không
Clear()	Xóa mọi phần tử trong danh sách

Chú ý:

- Hàm Add() có độ phức tạp hằng amortized, trong trường hợp xấu nhất là $O(n)$. Để hàm Add chạy hiệu quả chúng ta cần dự tính trước kích thước cho List khi tạo List.
- Hàm Sort có độ phức tạp là $O(\log n)$
- List có thể dùng thay thế cho LinkedList ở những nơi khi biết trước khả năng List chứa, không cần phải thêm, xóa ở vị trí ở giữa list.

C. **Tuple:** Tuple là một cấu trúc dữ liệu tiện ích dùng để lưu trữ một số phần tử (có thể có kiểu khác nhau) vào trong một đối tượng.

Nhận xét: Chúng ta thường dùng Tuple để tạo cạnh (edge), hay để lưu đồ thị có trọng số bằng danh sách kề.

1. Tạo Tuple

```
Tuple<int, int> t;
t= new Tuple<int, int>(2,3);

Console.Write(t.Item1 + " " + t.Item2);
```

2. Một số properties và methods thường dùng

Properties

- Item1, Item2, Item3, Item4, … dùng để truy cập phần tử thứ 1, 2, 3, 4, …

Method

- Tuple.Create(giá trị 1, giá trị 2, …) dùng để tạo bộ giá trị cho Tuple

D. *Stack:* Stack là cấu trúc dữ liệu dạng LIFO (Last In First Out) và cho phép chúng ta thực hiện nhanh các thao tác Push, Pop phần tử trong thời gian $O(1)$.

1. Tạo Stack

```
Stack<int> s = new Stack<int>();
s.Push(5);
s.Push(7);
s.Push(4);
s.Push(8);

while (s.Count>0)
{
  int num = s.Pop();
  Console.Write(num + " ");
}
```

2. Một số properties và methods thường dùng

Property

Property	Ý nghĩa
Count	Số lượng phần tử trong stack

Methods

Method	Ý nghĩa
Push(giá trị)	Thêm giá trị vào stack
Pop()	Lấy và xóa phần tử khỏi stack
Peek()	Lấy giá trị trên đỉnh stack (nhưng không xóa khỏi stack)
Contains(giá trị)	Kiểm tra có giá trị trong stack không
Clear()	Xóa mọi phần tử trong stack

E. *Queue:* Queue là cấu trúc dữ liệu dạng FIFO (First In First Out) và cho phép chúng ta thực hiện nhanh các thao tác Enqueue, Dequeue phần tử trong thời gian $O(1)$.

1. Tạo Queue

```
Queue<int> q = new Queue<int>();
q.Enqueue(5);
q.Enqueue(7);
q.Enqueue(4);
q.Enqueue(8);


while (q.Count>0)
{
  int num = q.Dequeue();
```

```
  Console.Write(num + " ");
}
```

2. Một số properties và methods thường dùng

Property

Property	Ý nghĩa
Count	Số lượng phần tử trong queue

Methods

Method	Ý nghĩa
Enqueue(giá trị)	Thêm giá trị vào queue
Dequeue()	Lấy và xóa phần tử khỏi queue
Peek()	Lấy giá trị trên đầu queue (nhưng không xóa khỏi queue)
Contains(giá trị)	Kiểm tra có giá trị trong queue không
Clear()	Xóa mọi phần tử trong queue

F. SortedSet: SortedSet là cấu trúc dữ liệu dùng để biểu diễn một tập hợp trên máy tính. Trong SortedSet, các phần tử không trùng nhau và được sắp xếp tăng dần. Các thao tác trên SortedSet như Thêm, Xóa, Tìm kiếm được thực hiện nhanh trong thời gian $O(\log n)$.

Nhận xét: SortedSet thường được dùng để tạo priority queue (hàng đợi ưu tiên)

1. Tạo SortedSet

```
SortedSet<int> s = new SortedSet<int>();
s.Add(4);
s.Add(2);
s.Add(4);
s.Add(6);
s.Add(8);

foreach(int x in s)
  Console.Write(x + " ");
```

2. Một số properties và methods thường dùng

Property

Property	Ý nghĩa
Count	Số lượng phần tử trong SortedSet
Min	Giá trị nhỏ nhất trong SortedSet
Max	Giá trị lớn nhất trong SortedSet

Methods

Method	Ý nghĩa
Add(giá trị)	Thêm giá trị vào SortedSet
Remove(giá trị)	Xóa giá trị khỏi SortedSet
Contains(giá trị)	Kiểm tra có giá trị trong SortedSet không
Clear()	Xóa mọi phần tử trong SortedSet

Buổi 1. Nhập xuất đồ thị và tính bậc của đỉnh

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Tạo file văn bản chứa đồ thị dạng: ma trận kề/danh sách kề/danh sách cạnh.
- Xây dựng ba loại cấu trúc dữ liệu lưu trữ đồ thị: ma trận kề, danh sách kề, danh sách cạnh.
- Nhập/Xuất file văn bản chứa đồ thị.
- Thao tác cơ bản trên đồ thị: Tính bậc đỉnh

Bai 1. Bậc của đồ thị vô hướng

Cho đơn đồ thị vô hướng $G = (V, E)$ có n $(n \leq 1000)$ đỉnh, được đánh số từ 1 đến n. Đồ thị G được lưu trong một file văn bản dưới dạng một ma trận kề. Hãy tổ chức cấu trúc dữ liệu ma trận kề để lưu trữ đồ thị, và viết chương trình đọc đồ thị G từ file đã cho, sau đó tính bậc của các đỉnh trong đồ thị.

Dữ liệu vào: File văn bản BACDOTHIVOHUONG.INP

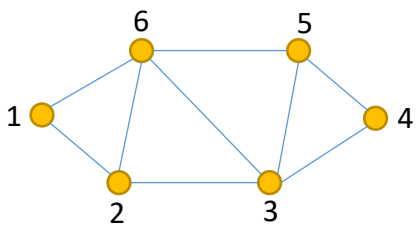
- Dòng đầu tiên chứa số nguyên n là số đỉnh của đồ thị.
- n dòng tiếp theo, mỗi dòng chứa n số biểu diễn ma trận kề của đồ thị.

Dữ liệu ra: File văn bản BACDOTHIVOHUONG.OUT

- Dòng thứ nhất chưa số n là số đỉnh của đồ thị.
- Dòng thứ hai chứa n số nguyên tương ứng là bậc của các đỉnh $1, 2, \ldots, n$

Các số trên cùng một dòng, cách nhau ít nhất 1 khoảng trắng.

Ví dụ:



BACDOTHIVOHUONG.INP	BACDOTHIVOHUONG.OUT
6 0 1 0 0 0 1 1 0 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 1 1 1 0 1 0	6 2 3 4 2 3 4

Bài 2. Bậc vào, bậc ra

Cho đơn đồ thị có hướng $G = (V, E)$ có n đỉnh được đánh số từ 1 đến n. Hãy tính **bậc vào và bậc ra** của tất cả các đỉnh của đồ thị.

Dữ liệu vào: File văn bản BACVAOBACRA.INP

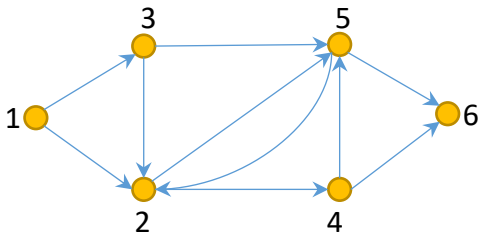
- Dòng đầu tiên chứa số nguyên n ($n \leq 1000$) là số đỉnh của đồ thị.
- n dòng tiếp theo, mỗi dòng chứa n số biểu diễn ma trận kề của đồ thị.

Dữ liệu ra: File văn bản BACVAOBACRA.OUT

- Dòng đầu là số nguyên dương n là số đỉnh của đồ thị.
- n dòng tiếp theo, mỗi dòng gồm hai số nguyên là bậc vào và bậc ra của đỉnh $1, 2, \ldots, n$

Các số trên cùng một dòng, cách nhau ít nhất 1 khoảng trắng.

Ví dụ:



BACVAOBACRA.INP	BACVAOBACRA.OUT
6 0 1 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0	6 0 2 3 2 1 2 1 2 3 2 2 0

Bài 3. Danh sách kề

Cho đơn đồ thị vô hướng $G = (V, E)$ có n ($n \leq 10^6$) đỉnh được đánh số từ 1 đến n và m cạnh ($m \leq 10^6$). Hãy tổ chức cấu trúc dữ liệu cho đồ thị dưới dạng danh sách kề, và viết chương trình đọc đồ thị G từ file đã cho, sau đó tính bậc của các đỉnh trong đồ thị.

Dữ liệu vào: File văn bản DANHSACHKE.INP

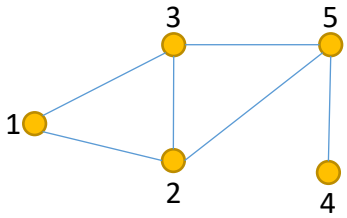
- Dòng đầu tiên chứa số đỉnh n của đồ thị.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cạnh (i, j) của đồ thị.

Chú ý: Đỉnh cô lập thì dóng đó rỗng

Dữ liệu ra: File văn bản DANHSACHKE.OUT

- Dòng thứ nhất chưa số n là số đỉnh của đồ thị.
- Dòng thứ hai chứa n số nguyên tương ứng là bậc của các đỉnh $1, 2, \ldots, n$

Ví dụ:



DANHSACHKE.INP	DANHSACHKE.OUT
5 2 3 1 3 5 1 2 5 5 2 3 4	5 2 3 3 1 3

Bài 4. Danh sách cạnh

Cho đơn đồ thị vô hướng $G = (V, E)$ có n ($n \leq 10^6$) đỉnh được đánh số từ 1 đến n và m cạnh ($m \leq 10^6$) được biểu diễn dưới dạng danh sách cạnh trong một file văn bản. Hãy tổ chức cấu trúc dữ liệu cho đồ thị dưới dạng danh sách cạnh, và viết chương trình đọc đồ thị G từ file đã cho, sau đó tính bậc của các đỉnh trong đồ thị.

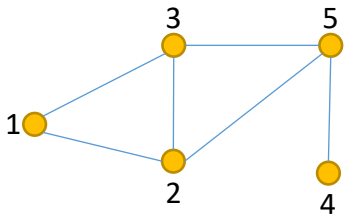
Dữ liệu vào: File văn bản DANHSACHCANH.INP

- Dòng đầu tiên chứa hai số nguyên n, m là số đỉnh và số cạnh của đồ thị.
- m dòng tiếp theo, mỗi dòng là cặp số biểu diễn một cạnh của đồ thị (các số cách nhau ít nhất 1 khoảng trắng)

Dữ liệu ra: File văn bản DANHSACHCANH.OUT

- Dòng đầu là số nguyên dương n là số đỉnh của đồ thị.
- Dòng thứ hai chứa n số nguyên tương ứng là bậc của các đỉnh $1, 2, \ldots, n$

Ví dụ:



DANHSACHCANH.INP	DANHSACHCANH.OUT
5 6 1 2 1 3 2 3 2 5 3 5 4 5	5 2 3 3 1 3

Bài tập làm thêm

Bài 1. Chuyển đổi cách biểu diễn đồ thị

Viết các hàm tiện ích cho phép chuyển đổi qua lại giữa các cách biểu diễn đồ thị khác nhau

- Chuyển từ Ma trận kề sang danh sách kề.
- Chuyển từ Ma trận kề sang danh sách cạnh.
- Chuyển từ Danh sách kề sang ma trận kề.
- Chuyển từ Danh sách kề sang danh sách cạnh.
- Chuyển từ Danh sách cạnh sang ma trận kề.
- Chuyển từ Danh sách cạnh sang danh sách kề.

Bài 2. Bồn chứa

Cho đơn đồ thị có hướng $G = (V, E)$. "Bồn chứa" trong đồ thị G là đỉnh chỉ có cung vào mà không có cung ra. Viết chương trình tìm các đỉnh bồn chứa trong đồ thị G.

Dữ liệu vào: File văn bản BONCHUA.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 1000$) của đồ thị.
- n dòng tiếp theo là ma trận kề của đồ thị.

Dữ liệu ra: File văn bản BONCHUA.OUT

- Dòng đầu là số nguyên dương k là số lượng bồn chứa trong đồ thị (Ghi 0 nếu G không có bồn chứa).
- Nếu $k > 0$ thì dòng thứ hai chứa danh sách các đỉnh bồn chứa.

Bai 3. Độ dài trung bình của cạnh

Cho đơn đồ thị vô hướng có trọng số $G = (V, E)$ được biểu diễn dưới dạng ma trận trọng số trong một file văn bản. Tìm các cạnh dài nhất và tính độ dài trung bình của các cạnh.

Dữ liệu vào: File văn bản TRUNGBINHCANH.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 1000$) của đồ thị.
- n dòng tiếp theo là ma trận trọng số của đồ thị (các số cách nhau ít nhất 1 khoảng trắng)

Dữ liệu ra: File văn bản TRUNGBINHCANH.OUT

- Dòng đầu tiên chứa số k là số lượng cạnh có độ dài dài nhất.
- Dòng thứ hai chứa k cặp số là k cạnh dài nhất
- Dòng thứ ba chứa độ dài trung bình các cạnh

Buổi 2. Tìm kiếm trên đồ thị bằng thuật toán BFS

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách cài đặt thuật toán tìm kiếm theo chiều rộng (Breadth First Search – BFS).
- Giải quyết một số bài toán dùng BFS: Tìm đường đi, Miền liên thông, Xác định cạnh cầu, đỉnh khớp của đồ thị, …

Chú ý:

1. Các bài tập từ buổi này trở về sau, nếu không nói gì thêm, chúng ta ngầm định dùng cách biểu diễn đồ thị là **danh sách kề**.
2. Nếu file chứa đồ thị không phải là danh sách kề thì vẫn có thể dùng danh sách kề để biểu diễn đồ thị.

Bài 1. Các đỉnh liên thông với x

Cho đơn đồ thị vô hướng $G = (V, E)$ và một đỉnh x cho trước. Hãy cho biết từ đỉnh x có thể đi đến những đỉnh nào bằng thuật toán BFS.

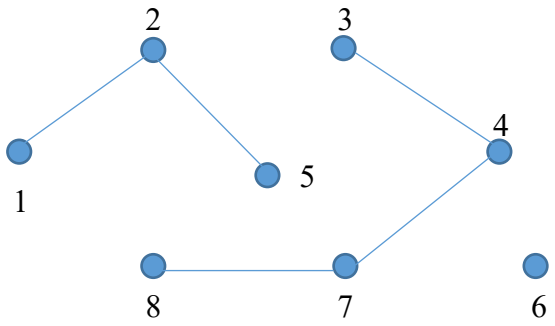
Dữ liệu vào: File văn bản LIENTHONGBFS.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 10^6$) của đồ thị và đỉnh x.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cạnh (i, j) của đồ thị.

Dữ liệu ra: File văn bản LIENTHONGBFS.OUT

- Dòng đầu tiên ghi số k là số lượng đỉnh có thể đi đến từ đỉnh x.
- Dòng thứ hai ghi k đỉnh tìm được.

Ví dụ:



LIENTHONGBFS.INP	LIENTHONGBFS.OUT
8 7 2 1 5 4 3 7 2 4 8 7	3 4 8 3

Bài 2. Tìm đường đi

Cho đơn đồ thị vô hướng $G = (V, E)$ và hai đỉnh x, y $(x \neq y)$. Hãy tìm đường đi từ đỉnh x đến đỉnh y bằng thuật toán BFS.

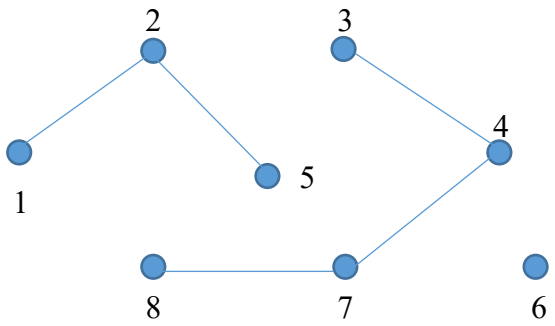
Dữ liệu vào: File văn bản TIMDUONGDIBFS.INP

- Dòng đầu tiên chứa số đỉnh n $(n \leq 10^6)$ của đồ thị và hai đỉnh x, y.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cạnh (i, j) của đồ thị.

Dữ liệu ra: File văn bản TIMDUONGDIBFS.OUT

- Dòng đầu tiên ghi số nguyên dương k là số đỉnh nằm trên đường đi từ đỉnh x đến đỉnh y (Tính luôn cả đỉnh x và y).
- Dòng thứ hai chứa k số nguyên là các đỉnh trên đường đi từ x đến y.

Ví dụ:



TIMDUONGDIBFS.INP	TIMDUONGDIBFS.OUT
8 3 8 2 1 5 4 3 7 2 4 8 7	4 3 4 7 8

Bài 3. Liệt kê thành phần liên thông

Cho đơn đồ thị vô hướng $G = (V, E)$. Hãy liệt kê các thành phần liên thông trong đồ thị bằng thuật toán BFS.

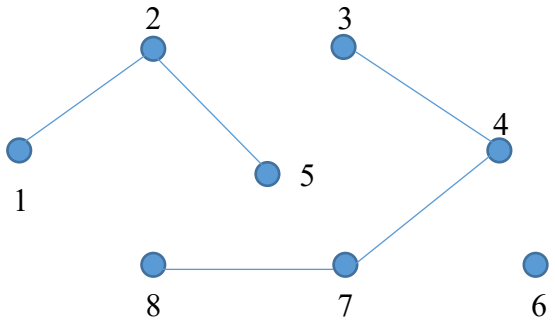
Dữ liệu vào: File văn bản LIETKELIENTHONGBFS.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 10^6$) của đồ thị.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cung (i, j) của đồ thị.

Dữ liệu ra: File văn bản LIETKELIENTHONGBFS.OUT

- Dòng đầu tiên ghi số nguyên dương k là số lượng thành phần liên thông của đồ thị.
- Dòng thứ i trong k dòng tiếp theo liệt kê các đỉnh của từng thành phần liên thông thứ k.

Ví dụ:



LIETKELIENTHONGBFS.INP	LIETKELIENTHONGBFS.OUT
8 2 1 5 4 3 7 2 4 8 7	3 1 2 5 3 4 7 8 6

Bài 4. Cạnh cầu và đỉnh khớp

Cho đơn đồ thị vô hướng $G = (V, E)$, cạnh (x, y) và đỉnh z. Hãy kiểm tra xem cạnh $(x,\ y)$ có phải là cầu không, đỉnh z có phải là đỉnh khớp hay không bằng thuật toán BFS.

Dữ liệu vào: File văn bản CanhCauDinhKhopBFS.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 10^6$) của đồ thị, hai số nguyên x, y mô tả cạnh (x, y) và đỉnh z.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cung (i, j) của đồ thị.

Dữ liệu ra: File văn bản CanhCauDinhKhopBFS.OUT

- Dòng đầu tiên chữ "canh cau" nếu (x, y) là cạnh cầu hay "khong la canh cau" nếu (x, y) không phải là cạnh cầu.
- Dòng thứ hai ghi "dinh khop" nếu z là đỉnh khớp hay "khong la dinh khop" nếu z không phải là đỉnh khớp.

Buổi 3. Tìm kiếm trên đồ thị bằng thuật toán DFS

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách cài đặt thuật toán DFS (Depth First Search)
- Giải quyết một số bài toán dùng DFS: Tìm đường đi, đồ thị phân đôi, …

Bài 1. Các đỉnh liên thông với x

Cho đơn đồ thị vô hướng $G = (V, E)$ và một đỉnh x cho trước. Hãy cho biết từ đỉnh x có thể đi đến những đỉnh nào bằng thuật toán DFS.

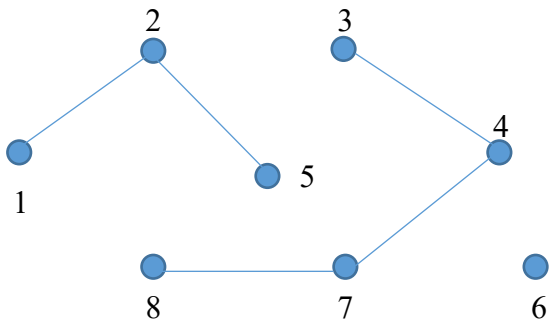
Dữ liệu vào: File văn bản LIENTHONGDFS.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 10^4$) của đồ thị và đỉnh x.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cạnh (i, j) của đồ thị.

Dữ liệu ra: File văn bản LIENTHONGDFS.OUT

- Dòng đầu tiên ghi số k là số lượng đỉnh tìm được.
- Dòng thứ hai ghi k đỉnh tìm được.

Ví dụ:



LIENTHONGDFS.INP	LIENTHONGDFS.OUT
8 7 2 1 5 4 3 7 2	3 4 3 8

4 8 7	

Bài 2. Tìm đường đi

Cho đơn đồ thị vô hướng $G = (V, E)$ và hai đỉnh x, y $(x \neq y)$. Hãy tìm đường đi từ đỉnh x đến đỉnh y bằng thuật toán DFS.

Dữ liệu vào: File văn bản TIMDUONGDFS.INP

- Dòng đầu tiên chứa số đỉnh n $(n \leq 10^4)$ của đồ thị và hai đỉnh x, y.
- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cạnh (i, j) của đồ thị.

Dữ liệu ra: File văn bản TIMDUONGDFS.OUT

- Dòng đầu tiên ghi số nguyên dương k là số đỉnh nằm trên đường đi từ đỉnh x đến đỉnh y (tính luôn cả đỉnh x và y).
- Dòng thứ hai chứa k số nguyên là các đỉnh trên đường đi từ x đến y.

Ví dụ:

TIMDUONGDFS.INP	TIMDUONGDFS.OUT
8 3 8 2 1 5 4 3 7 2 4 8 7	4 3 4 7 8

Bài 3. Đồ thị phân đôi

Cho đơn đồ thị vô hướng $G = (V, E)$. G được gọi là đồ thị phân đôi nếu chúng ta có thể chia tập đỉnh V thành 2 tập đỉnh V_1, V_2 không giao nhau sao cho các cạnh của G chỉ nối đỉnh trong V_1 với đỉnh trong V_2 (không có cạnh nối hai đỉnh trong cùng một tập đỉnh). Hãy kiểm tra đồ thị G có thể phân đôi không.

Dữ liệu vào: File văn bản DOTHIPHANDOI.INP

- Dòng đầu tiên chứa số đỉnh n $(n \leq 10^4)$ của đồ thị và hai đỉnh x, y.

- n dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cạnh (i, j) của đồ thị.

Dữ liệu ra: File văn bản DOTHIPHANDOI.OUT

- Dòng duy nhất câu "Do thi phan doi" hay "Do thi khong phan doi".

Bài tập làm thêm

Bài 1. DFS Không đệ quy

Hãy cài đặt thuật toán DFS không đệ quy.

Bài 2. Mê cung

Mê cung hình chữ nhật kích thước nxm $(n, m \leq 1000)$ gồm các ô vuông đơn vị. Trên mỗi ô ghi ba ký tự

- : Nếu ô đó an toàn, có thể đi qua
- X : Nếu ô đó có cạm bẩy, không thể đi qua
- E : Nếu ô đó có một nhà thám hiểm đang đứng (Duy nhất chỉ có một ô trên mê cung)

Nhà tham hiểm có thể từ một ô đi sang một trong số các ô có chung cạnh với ô đang đứng. Một cách đi thoát khỏi mê cung là một hành trình đi qua các ô an toàn ra một ô biên. Hãy chỉ giúp cho nhà tham hiểm một hành trình thoát khỏi mê cung đi qua ít ô nhất.

Buổi 4. Đồ thị Euler và Đồ thị Hamilton

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách cài đặt thuật toán tìm chu trình Euler trong đồ thị, Tìm chu trình Hamilton trong đồ thị.
- Giải một số bài toán liên quan đến đồ thị Euler.

Bài 1. Đồ thị Euler

Cho đơn đồ thị vô hướng $G = (V, E)$. Hãy kiểm tra xem đồ thị G có phải là đồ thị Euler hay không. Nếu là đồ thị Euler hãy tìm chu trình Euler của đồ thị.

Dữ liệu vào: File văn bản DOTHIEULER.INP

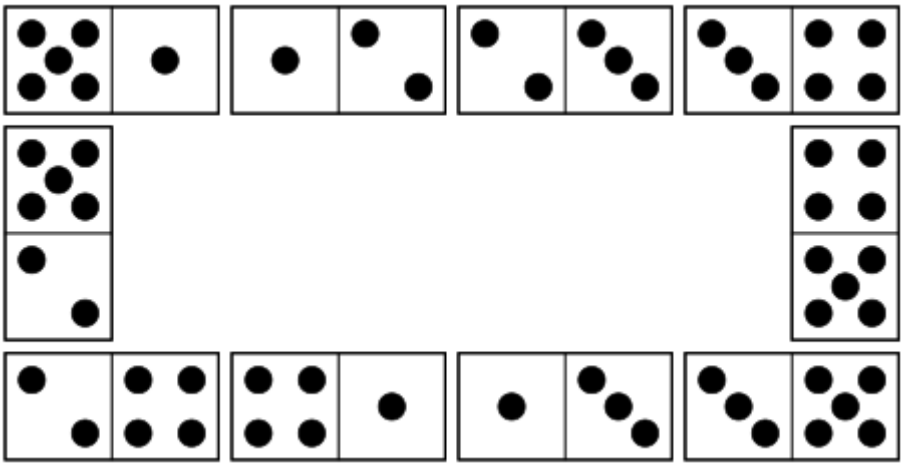
- Dòng đầu tiên chứa số đỉnh n ($n \leq 1000$) của đồ thị.
- n dòng tiếp theo là ma trận kề của đồ thị.

Dữ liệu ra: File văn bản DOTHIEULER.OUT

- Dòng đầu tiên ghi số 1 hay 0 tương ứng với đồ thị là đồ thi Euler hay không Euler.
- Nếu dòng đầu là 1 thì dòng thứ hai chứa các đỉnh tạo thành chu trình Euler.

Bài 2. Domino

Một domino là một hình chữ nhật $2x1$. Mỗi nửa của domino là một con số được thể hiện bằng các dấu chấm. Trong hình sau, chúng ta có 10 domino, mỗi domino là một cặp số tương ứng với cặp số được chọn ra trong tập $\{1, 2, 3, 4, 5\}$. Theo luật chơi domino thì hai domino để gần nhau nếu hai phần gần nhau có cùng số.



Hãy sắp xếp 28 quân domino thành một vòng tròn theo luật chơi domino.

Bài 3. Đồ thị nửa Euler

Cho đơn đồ thị vô hướng $G = (V, E)$. Hãy kiểm tra xem đồ thị G có phải là đồ thị nửa Euler hay không. Nếu là đồ thị nửa Euler hãy tìm đường đi Euler của đồ thị.

Dữ liệu vào/ra giống bài 1

Bài 4. Đồ thị Hamilton

Cho đơn đồ thị vô hướng $G = (V, E)$. Hãy tìm một chu trình Hamilton của đồ thị G theo phương pháp đệ quy và quay lui.

Dữ liệu vào/ra giống bài 1

Bài tập làm thêm

Bài 1. Euler không đệ quy

Hãy cài hàm không đệ quy để tìm chu trình Euler trên ma trận kề.

Bài 2. Euler trên danh sách kề

Hãy cài hàm không đệ quy để tìm chu trình Euler trên danh sách kề.

Buổi 5. Tìm đường đi ngắn nhất

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách cài đặt thuật toán Dijkstra hiệu quả.
- Biết cách cài đặt thuật toán Floyd (Sử dụng ma trận trọng số).
- Giải quyết một số bài toán liên quan đến đường đi ngắn nhất.

Bài 1. Đường đi ngắn nhất

Cho đơn đồ thị có hướng có trọng số $G = (V, E)$ và hai đỉnh s, t. Hãy tìm đường đi ngắn nhất từ đỉnh s đến đỉnh t theo thuật toán Dijkstra.

Dữ liệu vào: File văn bản DIJKSTRA.INP

- Dòng đầu tiên chứa số đỉnh n ($m \leq 10^6$) , số cung m ($m \leq 10^6$) của đồ thị và 2 đỉnh x, y.
- m dòng tiếp theo, mỗi dòng chứa 3 số u, v, w mô tả cung (u, v) có trọng số w ($0 < w \leq 10^5$).

Dữ liệu ra: File văn bản DIJKSTRA.OUT

- Dòng đầu tiên ghi số đỉnh đường đi ngắn nhất đi qua (tính luôn hai đỉnh x và y) và độ dài đường đi.
- Dòng thứ hai ghi danh sách các đỉnh của đường đi ngắn nhất tìm được từ s đến t.

Bài 2. Đường đi ngắn nhất qua đỉnh trung gian

Cho đơn đồ thị có hướng có trọng số không âm $G = (V, E)$ và ba đỉnh s, t và x. Hãy tìm đường đi ngắn từ đỉnh s đến đỉnh t và đường đi đó phải đi qua đỉnh x.

Bài 3. Đường đi ngắn nhất

Cho đơn đồ thị có hướng có trọng số không âm $G = (V, E)$. Hãy tìm đường đi ngắn nhất giữa các cặp đỉnh theo thuật toán Floyd.

Buổi 6. Vận dụng thuật toán tìm đường đi ngắn nhất

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Mô hình hóa bài toán bằng đồ thị.
- Ứng dụng thuật toán tìm đường đi ngắn nhất để giải bài toán.

Bài 1. Đi trên bảng

Cho bảng kích thước $n\ x\ m$ $(n, m \leq 1000)$ các số tự nhiên. Từ một ô có thể di chuyển sang một ô kề cạnh với nó. Hãy tìm một cách đi từ ô (x, y) đến một ô biên sao cho tổng các số của các ô đi qua là nhỏ nhất.

Dữ liệu vào: File văn bản BANGSO.INP

- Dòng đầu tiên chứa các số n, m, x, y.
- n dòng tiếp theo, mỗi dòng chứa m số.

Dữ liệu ra: File văn bản BANGSO.OUT

- Giá trị tổng các số của các ô đã đi qua.

Bài 2. Đường tròn

Trên mặt phẳng cho n đường tròn, đường tròn thứ i được cho bởi bộ ba số thực (x_i, y_i, r_i), trong đó (x_i, y_i) là toạ độ tâm của đường tròn và r_i là bán kính của đường tròn. Chi phí di chuyển trên mỗi đường tròn bằng 0. Chi phí di chuyển giữa hai đường tròn bằng khoảng cách giữa chúng. Hãy tìm phương án di chuyển giữa hai đường tròn s, t cho trước với chi phí ít nhất.

Dữ liệu vào: File văn bản DUONGTRON.INP

- Dòng đầu tiên chứa n, s, t.
- n dòng tiếp theo, dòng i chứa ba số nguyên x_i, y_i, r_i.

Dữ liệu ra: File văn bản DUONGTRON.OUT

- Dòng duy nhất chứa chi phí ít nhất tìm được

Bài tập làm thêm

Bài 1. Chọn thành phố

Có n ($n \leq 100$) thành phố, khoảng cách giữa hai thành phố i và j là a_{ij}. Người ta muốn tổ chức một cuộc họp lãnh đạo cho n thành phố. Hãy tìm một thành phố tổ chức sao cho khoảng cách của người đi xa nhất là nhỏ nhất có thể.

Dữ liệu vào: File văn bản THANHPHO.INP

- Dòng đầu tiên chứa n.
- n dòng tiếp theo, mỗi dòng gồm n mô tả a_{ij}.

Dữ liệu ra: File văn bản THANHPHO.OUT

- Dòng đầu là thành phố đăng cai tổ chức.
- Dòng thứ hai, thời gian của người phải đi xa nhất.

Bài 2. Đến trường

Gia đình Tuấn sống ở thành phố XYZ. Hàng ngày, mẹ đi ô tô đến cơ quan làm việc còn Tuấn đi bộ đến trường học. Thành phố XYZ có N nút giao thông được đánh số từ 1 đến N. Nhà Tuấn nằm ở nút giao thông 1, trường của Tuấn nằm ở nút giao thông K, cơ quan của mẹ nằm ở nút giao thông N. Từ nút đến nút có không quá một đường đi một chiều, tất nhiên, có thể có đường đi một chiều khác đi từ nút đến nút. Nếu từ nút đến nút có đường đi thì thời gian đi bộ từ nút đến nút hết a_{ij} phút, còn đi ô tô hết b_{ij} ($0 < b_{ij} \leq a_{ij}$) phút.

Hôm nay, mẹ và Tuấn xuất phát từ nhà lúc 7 giờ. Tuấn phải có mặt tại trường lúc 7 giờ 59 phút để kịp vào lớp học lúc 8 giờ. Tuấn băn khoăn không biết có thể đến trường đúng giờ hay không, nếu không Tuấn sẽ phải nhờ mẹ đưa đi từ nhà đến một nút giao thông nào đó.

Yêu cầu: Cho biết thông tin về các đường đi của thành phố XYZ. Hãy tìm cách đi để Tuấn đến trường không bị muộn giờ còn mẹ đến cơ quan làm việc sớm nhất.

Dữ liệu: Vào từ file văn bản SCHOOL.INP có dạng:

- Dòng đầu ghi ba số nguyên dương N, M, K ($3 \leq N \leq 10.000; M \leq 10^5$; $1 < K < N$), trong đó N là số nút giao thông, M là số đường đi một chiều, K là nút giao thông - trường của Tuấn.
- M dòng tiếp theo, mỗi dòng chứa 4 số nguyên dương i, j, a_{ij}, b_{ij} ($1 \leq i, j, \leq N, b_{ij} \leq a_{ij} \leq 60$) mô tả thông tin đường đi một chiều từ i đến j.

Hai số liên tiếp trên một dòng cách nhau một dấu cách. Dữ liệu bảo đảm luôn có nghiệm.

SCHOOL.INP	SCHOOL.OUT
3 5 6 1 4 60 40 1 2 60 30 2 3 60 30 4 5 30 15 4 3 19 10 3 5 20 10	55

Kết quả: Đưa ra file văn bản SCHOOL.OUT gồm một dòng chứa một số nguyên là thời gian sớm nhất mẹ Tuấn đến được cơ quan còn Tuấn thì không bị muộn học.

Buổi 7. Cây khung và Cây khung nhỏ nhất

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách cài đặt thuật toán tìm cây khung của đồ thị bằng DFS hay BFS.
- Biết cách cài đặt thuật toán tìm cây khung nhỏ nhất bằng thuật toán: Kruskal và Prim.

Bài 1. Tìm cây khung

Cho đơn đồ thị vô hướng $G = (V, E)$. Hãy tìm cây khung của đồ thị G theo thuật toán BFS và DFS.

Dữ liệu vào: File văn bản CAYKHUNG.INP

- Dòng đầu tiên chứa hai n, m $(n, m \leq 10^6)$, trong đó n là số đỉnh, m là số cạnh của đồ thị.
- m dòng tiếp theo, mỗi dòng chứa hai số u, v mô tả cạnh (u, v) trong đồ thị.

Dữ liệu ra: File văn bản CAYKHUNG.OUT

- Dòng đầu tiên ghi số $(n - 1)$ là số cạnh trong cây khung.
- $n - 1$ dòng tiếp theo, mỗi dòng gồm hai số nguyên là hai đỉnh của một cạnh trong cây khung.

Bài 2. Kruskal

Cho đơn đồ thị vô hướng có trọng số $G = (V, E)$. Hãy tìm cây khung nhỏ nhất của đồ thị G theo thuật toán Kruskal và tính tổng độ dài các cạnh của cây khung tìm được.

Dữ liệu vào: File văn bản KRUSKAL.INP

- Dòng đầu tiên chứa hai n, m $(n, m \leq 10^6)$, trong đó n là số đỉnh, m là số cạnh của đồ thị.
- m dòng tiếp theo, mỗi dòng chứa ba số u, v, w cho biết cạnh (u, v) có trọng số w.

Dữ liệu ra: File văn bản KRUSKAL.OUT

- Dòng đầu tiên ghi số $(n - 1)$ là số cạnh trong cây khung nhỏ nhất.
- $n - 1$ dòng tiếp theo, mỗi dòng gồm ba số u, v, w cho biết cạnh (u, v) là cạnh trong cây khung nhỏ nhất có trọng số w.

Bài 3. Prim

Cho đơn đồ thị vô hướng có trọng số $G = (V, E)$. Hãy tìm cây khung nhỏ nhất của đồ thị theo thuật toán Prim.

Buổi 8. Vận dụng thuật toán tìm cây khung nhỏ nhất

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách mô hình bài toán thực tế bằng đồ thị.
- Biết cách vận dụng thuật toán tìm cây khung nhỏ nhất của đồ thị.

Bài 1. Cây khung x

Cho đồ thị $G = (V, E)$ có n đỉnh ($1 \leq n \leq 10^6$), m cạnh ($1 \leq m \leq 10^6$) và độ dài x. Hãy tìm cây khung nhỏ nhất có độ dài cạnh nhỏ nhất bằng x.

Input: CayKhungX.INP

- Dòng thứ nhất chứa ba số n, m, x
- M dòng sau chứa bộ ba số $(u_i, v_i, , w_i)$ mô tả cạnh thứ (u_i, v_i) có trọng số w_i

Output: CayKhungX.OUT

- Chứa số -1 nếu không có cây khung thỏa điều kiện bài toán, nếu có thì ghi tổng trọng số các cạnh của cây khung tìm được.

Ví dụ :

CayKhungX.INP	CayKhungX.OUT
6 9 5 1 2 2 1 6 5 2 6 4 2 3 10 5 2 8 5 3 13 6 5 7 5 4 7 3 4 1	37

Bài 2. Constructing Roads

There are N villages, which are numbered from 1 to N, and you should build some roads such that every two villages can connect to each other. We say two village A and B are connected, if and only if there is a road between A and B, or there exists a village C such that there is a road between A and C, and C and B are connected.

We know that there are already some roads between some villages and your job is the build some roads such that all the villages are connect and the length of all the roads built is minimum.

Input

The first line is an integer N ($3 \leq N \leq 100$), which is the number of villages. Then come N lines, the i-th of which contains N integers, and the $j-th$ of these N integers is the distance (the distance should be an integer within $[1, 1000]$) between village i and village j.

Then there is an integer Q ($0 \leq Q \leq \frac{N(N+1)}{2}$). Then come Q lines, each line contains two integers a and b ($1 \leq a < b \leq N$), which means the road between village a and village b has been built.

Output

You should output a line contains an integer, which is the length of all the roads to be built such that all the villages are connected, and this value is minimum.

Sample Input	Sample Output
3 0 990 692 990 0 179 692 179 0 1 1 2	179

Buổi 9. Tìm luồng cực đại trong mạng

Mục tiêu. Sau khi hoàn thành bài thực hành này sinh viên có thể:

- Biết cách cài đặt thuật toán Ford-Fulkerson tìm luồng cực đại trong mạng.
- Ứng dụng luồng cực đại giải một số bài toán.

Bài 1. Ford-Fulkerson

Cho đồ thị có hướng có trọng số nguyên $G = (V, E)$ và 2 đỉnh s, t. Hãy tìm một luồng cực đại từ đỉnh phát s đến đỉnh thu t trong G.

Dữ liệu vào: File văn bản Ford_Fulkerson.INP

- Dòng đầu tiên chứa số đỉnh n ($n \leq 1000$) và số cung m ($m \leq 10^5$) của đồ thị và đỉnh phát s, đỉnh thu t.
- m dòng tiếp theo, mỗi dòng chứa ba số nguyên dương u, v, c tương ứng với một cung nối từ u đến v với sức chứa c ($c \leq 10^4$).

Dữ liệu ra: File văn bản Ford_Fulkerson.OUT

- Dòng đầu ghi hai số nguyên k và l tương ứng là số lượng đỉnh luồng cực đại đi qua và giá trị của luồng.
- k dòng tiếp theo mô tả các cạnh luồng đi qua, mỗi dòng gồm ba số nguyên u, v, f mô tả cạnh (u, v) và giá trị luồng f trên cạnh đó.

Bài 2. Bài toán ghép cặp

Ở một vùng nọ có m chàng trai và n cô gái đến tuổi thành hôn, nhưng việc kết hôn là do già làng quyết định. Hàng năm cứ mùa xuân về, mỗi chàng trai đưa cho già làng một danh sách các cô gái mà anh ta thích. Già làng phải chọn ra một số cặp để tổ chức đám cưới sao cho trong mỗi cặp đó các chàng trai đều lấy cô gái trong danh sách mà mình đã đưa và đảm bảo luật hôn nhân gia đình. Hãy giúp già làng tổ chức nhiều đám cưới nhất.

Bài 3. Điền số trên ma trận

Cho hai dãy số nguyên dương $A = (a_1, a_2, \dots, a_n)$ và $B = (b_1, b_2, \dots, b_m)$ thỏa điều kiện $a_i \leq m$ và $b_j \leq n$. Hãy tạo một bảng $C[nxm]$ sao cho

- Giá trị mỗi ô là 0 hay 1
- Tổng các số trên dòng i bằng a_i
- Tổng các số trên cột j bằng b_j