Meeting Prof. Miguel...

Big-O Blue - Lecture 11: Floyd Warshall

Tóm tắt đề bài

Tóm tắt đề bài

- 1 thành phố có 26 địa điểm (‘A’, ‘B’, ..., ‘Z’) và N con đường kết nối chúng.
- Mỗi con đường:
  - Hao tốn C năng lượng để đi qua ($0 \leq C < 500$)
  - Có thể là đường một chiều (‘U’) hoặc đường 2 chiều (‘B’)
  - Chỉ dành cho người dưới 30 tuổi (‘Y’) hoặc người trên 30 tuổi (‘M’)
- Nhân vật chính 25 tuổi, đứng ở điểm S. Giáo sư Miguel đã ngoài 40 tuổi đứng tại điểm T.
- Tìm các địa điểm để 2 người gặp nhau, sao cho tổng năng lượng 2 người bỏ ra là ít nhất.
  - Nếu có nhiều địa điểm thì in năng lượng ít nhất và các địa điểm theo thứ tự tăng dần của thứ tự từ điển
  - Nếu không có địa điểm nào thì in ra là “You will never meet.”

Giải thích ví dụ

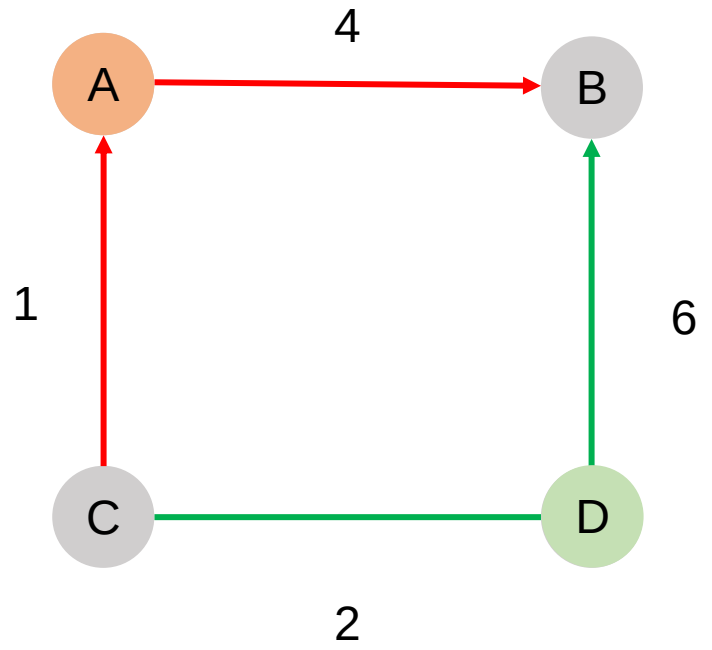
Ví dụ 1

Input

```
4
Y U A B 4
Y U C A 1
M U D B 6
M B C D 2
A D
```

Output

```
10 B
```



Bạn xuất phát ở A, Giáo sư xuất phát ở D

- Chọn địa điểm là A thì GS không có đường đi đến đó
- Chọn địa điểm gặp nhau là B.
  - Bạn tốn 4 năng lượng
  - Giáo sư tốn 6 năng lượng
- Chọn địa điểm là C thì bạn không thể đi đến đó
- Chọn địa điểm là D thì bạn không thể đi đến đó

→ Chọn địa điểm B và tổng năng lượng là 10.

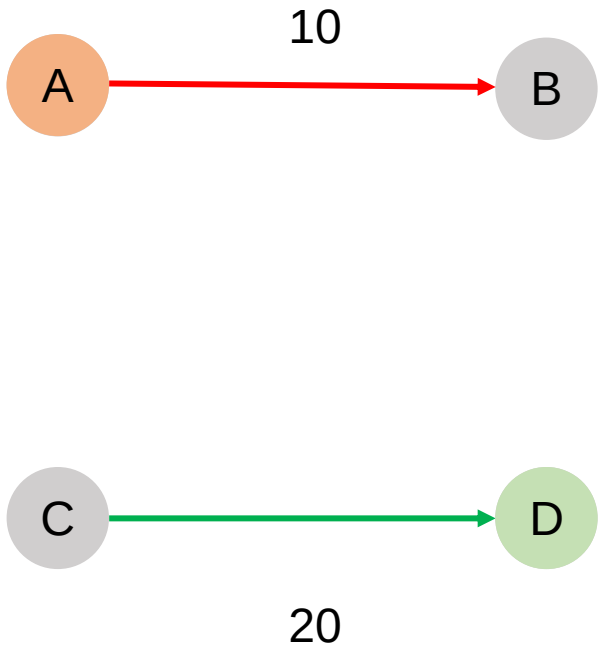
Ví dụ 2

Input

```
2
Y U A B 10
M U C D 20
A D
```

Output

You will never meet.



Không có cách nào để 2 người gặp nhau

Hướng dẫn giải

Nhận xét

Xét tất cả các địa điểm, với mỗi địa điểm u, tính xem nếu chọn u làm địa điểm gặp nhau thì tổng năng lượng tối thiểu là bao nhiêu?

Để tổng năng lượng là tối thiểu:

- Bạn xuất phát từ S, đi qua các con đường "Y", đến u với ít năng lượng nhất.
- Giáo sư xuất phát từ T, đi qua các con đường "M", đến u với ít năng lượng nhất.

→ Việc di chuyển của 2 người không ảnh hưởng tới nhau.

→ Ta có thể tìm riêng quãng đường cho từng người.

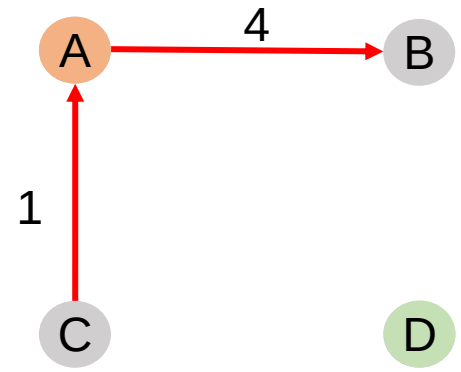
Nhận xét

Biểu diễn thành phố thành 2 đồ thị, tương ứng với 2 loại con đường “Y” và “M”:

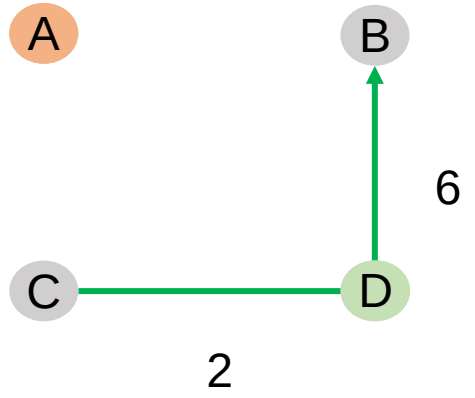
- Địa điểm “A” là đỉnh 0, “B” là đỉnh 1, …, “Z” là đỉnh 25.
- Các cạnh là các con đường (một chiều hoặc hai chiều)

Trên đồ thị gồm các cạnh “Y”, tìm đường đi ngắn nhất từ S tới mọi đỉnh.

Trên đồ thị gồm các cạnh “M”, tìm đường đi ngắn nhất từ M tới mọi đỉnh.



Đồ thị chỉ chứa các cạnh ‘Y’



Đồ thị chỉ chứa các cạnh ‘M’

Các bước giải

Bước 1: Đọc dữ liệu, xây dựng 2 đồ thị với 2 loại con đường "Y" và "M".

Bước 2: Sử dụng thuật toán Dijkstra / Bellman-Ford / Floyd-Warshall:

+ Trên đồ thị các cạnh "Y", tìm đường đi ngắn nhất từ S tới mọi đỉnh (lưu vào mảng $distS$)

+ Trên đồ thị các cạnh "M", tìm đường đi ngắn nhất từ T tới mọi đỉnh (lưu vào mảng $distT$)

Bước 3: Chuẩn bị:

+ Tổng chi phí tối ưu: $total\_energy = +\infty$

+ Danh sách các thành phố được chọn: $cities = [\,]$

Bước 4: Duyệt qua từng đỉnh u từ $0 \rightarrow 25$:

+ Tính chi phí: $E = distS[u] + distT[u]$

+ Nếu $E = total\_energy$

→ Thêm u vào danh sách $cities$.

+ Nếu $E < total\_energy$:

→ Gán $total\_energy = E$ và gán $cities = [char(u + 65)]$

Bước 5: Nếu $total\_energy = +\infty$**,** thông báo không có cách gặp nhau.

Ngược lại, in ra $total\_energy$ và danh sách $cities$.

Độ phức tạp: O(T * N^3) với T là số lượng testcase và N là số lượng đỉnh (26)

Mã giả

Mã giả

```
function Floyd_warshall(dist):
        for k = 0 to 25:
            for u = 0 to 25:
                for v = 0 to 25:
                    dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v])
Function main:
    while True:
        read(N)
        if (N == 0)
            break
        distS = [[0 if i == j else INF for i in range(26)] for j in range(26)]
        distT = [[0 if i == j else INF for i in range(26)] for j in range(26)]
        for i = 1 to N:
            read(type, direction, u, v, c)
            u = ord(u) - ord('A')
            v = ord(v) - ord('A')
            if type == 'Y':
                distS[u][v] = min(distS[u][v], c)
                if direction == 'B':
                    distS[v][u] = min(distS[v][u], c)
            else:
                distT[u][v] = min(distT[u][v], c)
                if direction == 'B':
                    distT[v][u] = min(distT[v][u], c)
```

```
    read(S, T)
    S = ord(S) - ord('A')
    T = ord(T) - ord('A')
    Floyd_warshall(distS)
    Floyd_warshall(distT)
    total_energy = INF
    cities = []
    for u = 0 to 25:
        if distS[S][u] != INF and distT[T][u] != INF:
                E = distS[S][u] + distT[T][u]
                if E == total_energy:
                    city = chr(u + ord('A'))
                    cities.add(city)
                if E < total_energy:
                    total_energy = E
                    city = char(u + ord('A'))
                    cities = [city]

    if total_energy == INF:
        print("You will never meet.")
    else:
        print(total_energy)
        for city in cities:
            print(city)
```