Asterix and Obelix

Big-O Blue - Lecture 15: Floyd Warshall

Tóm tắt đề bài

Tóm tắt đề bài

Có **C** thành phố và **R** con đường nối giữa các thành phố đó. Khi đi qua mỗi con đường sẽ tốn *chi phí đi đường* **d**.

Asterix sẽ tổ chức một bữa tiệc tại một trong các thành phố đi qua trên đường về nhà. Nếu tổ chức bữa tiệc tại thành phố **i** sẽ tốn *chi phí tổ chức tiệc* **cost[i]**.

Trên con đường ông chọn để đi về nhà, ông sẽ tổ chức bữa tiệc tại thành phố có *chi phí tổ chức tiệc* đắt nhất.

? Tính tổng chi phí nhỏ nhất (*chi phí đi đường* + *chi phí tổ chức tiệc*) cần bỏ ra để đi về nhà (từ thành phố **C1** về thành phố **C2**)

Mô tả Input/Output

Input:

- Dòng đầu tiên chứa **C** là số thành phố, **R** là số con đường, **Q** là số truy vấn.
- Dòng tiếp theo chứa **C** số nguyên là chi phí tổ chức tiệc của **C** thành phố.
- **R** dòng tiếp theo là thông tin **R** con đường hai chiều.
- **Q** dòng tiếp theo chứa {điểm bắt đầu, đích đến} của hành trình.

(C ≤ 80, R ≤ 1000, Q ≤ 6320)

Output:

Case #**i:**

totalCost1

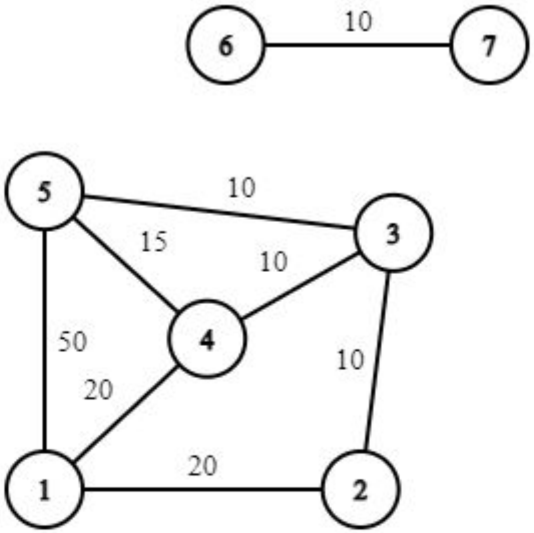
....

totalCostQ

Nếu không tồn tại đường đi từ C1 → C2: “-1”

Giải thích ví dụ

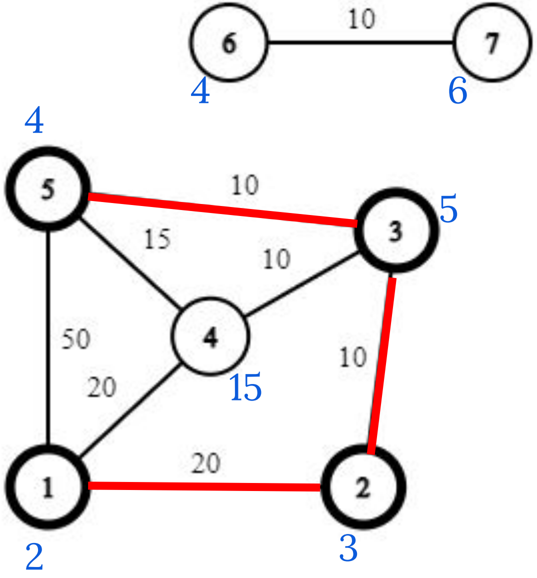
Ví dụ 1



Input:
7 8 5
2 3 5 15 4 4 6
1 2 20
1 4 20
1 5 50
2 3 10
3 4 10
3 5 10
4 5 15
6 7 10
1 5
1 6
5 1
3 1
6 7

Output:
Case #**1**
45
-1
45
35
16

Ví dụ 1



Input:

7 8 5
2 3 5 15 4 4 6
1 2 20
1 4 20
1 5 50
2 3 10
3 4 10
3 5 10
4 5 15
6 7 10
1 5
1 6
5 1
3 1
6 7

Đường về nhà: 1 → 5

Đường đi: 1 → 2 → 3 → 5:

- dist = 20 + 10 + 10 = 40
- maxCost = 5

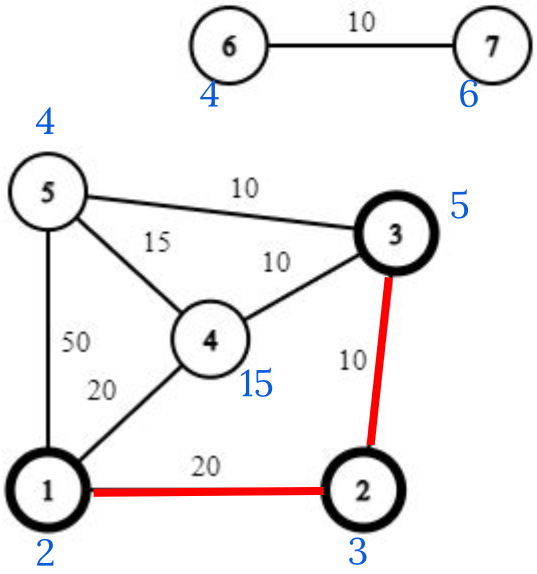
→ TotalCost = dist + maxCost = 40 + 5 = **45**

Đường đi: 1 → 4 → 5:

- dist = 20 + 15 = 35
- maxCost = 15

→ TotalCost = dist + maxCost = 35 + 15 = 50

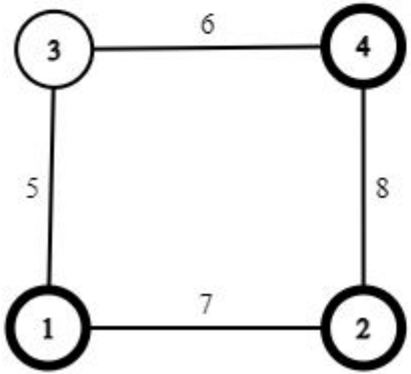
Ví dụ 1



Input:
7 8 5
2 3 5 15 4 4 6
1 2 20
1 4 20
1 5 50
2 3 10
3 4 10
3 5 10
4 5 15
6 7 10
1 5
1 6
5 1
3 1
6 7

Đường về nhà: 3 → 1
Đường đi: 3 → 2 → 1 • dist = 10 + 20 = 30 • maxCost = 5 → TotalCost = dist + maxCost = 30 + 5 = **35**
Đường đi: 3 → 4 → 1 • dist = 10 + 20 = 30 • maxCost = 15 → TotalCost = dist + maxCost = 30 + 15 = 45

Ví dụ 2



Input:

4 4 2

2 1 8 3

1 2 7

1 3 5

2 4 8

3 4 6

1 4

2 3

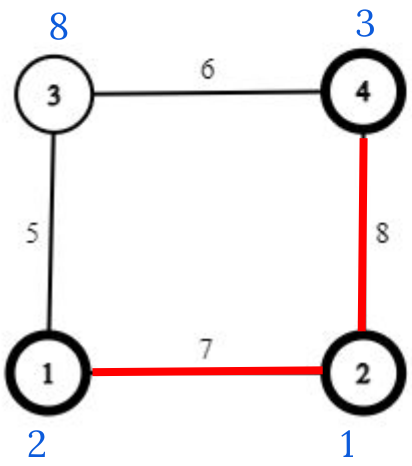
Output:

Case #**2**

18

20

Ví dụ 2



Input:
4 4 2
2 1 8 3
1 2 7
1 3 5
2 4 8
3 4 6
1 4
2 3
Output:
Case #**2**
18
20

Đường về nhà: 1 → 4

Đường đi: 1 → 2 → 4

- dist = 7 + 8 = 15
- maxCost = 3

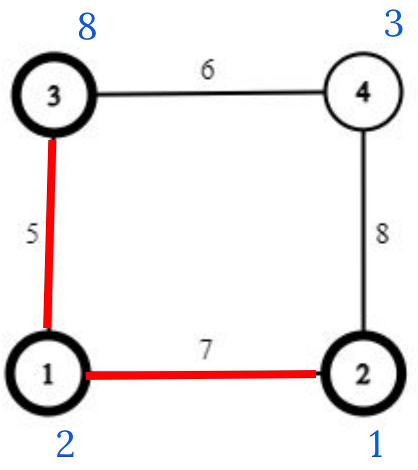
→ TotalCost = dist + maxCost = 15 + 3 = **18**

Đường đi: 1 → 3 → 4:

- dist = 5 + 6 = 11
- maxCost = 8

→ TotalCost = dist + maxCost = 11 + 8 = 19

Ví dụ 2



Input:
4 4 2
2 1 8 3
1 2 7
1 3 5
2 4 8
3 4 6
1 4
2 3
Output:
Case #2
18
20

Đường về nhà: 2 → 3
Đường đi: 2 → 1 → 3 • dist = 7 + 5 = 12 • maxCost = 8 → TotalCost = **20**
Đường đi: 2 → 4 → 3 • dist = 8 + 6 = 14 • maxCost = 8 → TotalCost = 22

Hướng dẫn giải

Nhận xét

1. Chi phí đi từ **u** đến **v** không chỉ bằng tổng trọng số của các cạnh trên đường đi (*chi phí đi đường*), mà còn phụ thuộc vào chi phí lớn nhất trong các đỉnh đi qua (*chi phí tổ chức tiệc*).
   → Sử dụng thêm ma trận **maxCost** với **maxCost[u][v]** là chi phí của đỉnh lớn nhất trên đường đi của **dist[u][v]**.
2. Tổng chi phí đi từ **i** đến **j**: **dist[i][j] + maxCost[i][j]**

Ý tưởng

Tổng chi phí đi từ **i** đến **j** = **dist[i][j] + maxCost[i][j]**

Tổng chi phí đi từ **i** đến đỉnh trung gian **k**, rồi từ đỉnh trung gian **k** đến đỉnh **j**:

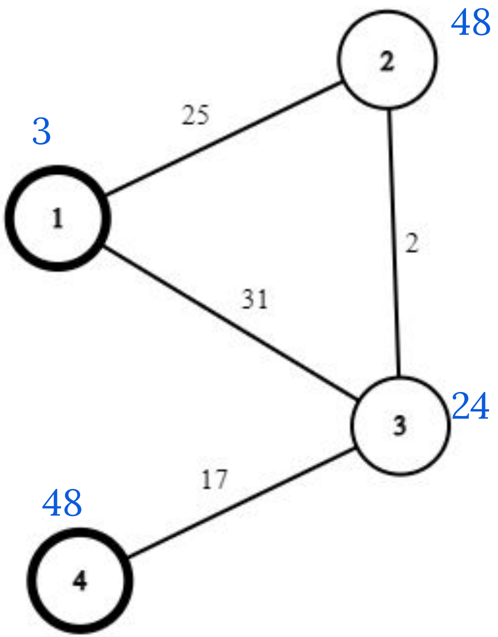
dist[i][k] + dist[k][j] + max(maxCost[i][k], maxCost[k][j])

Nếu **dist[i][j] + maxCost[i][j] > dist[i][k] + dist[k][j] + max(maxCost[i][k], maxCost[k][j]):**

Cập nhật **dist[i][j] = dist[i][k] + dist[k][j]**

maxCost[i][j] = max(maxCost[i][k], maxCost[k][j])

Minh họa ý tưởng



Input:

4 4 1

3 48 24 48

1 2 25

2 3 2

1 3 31

3 4 17

1 4

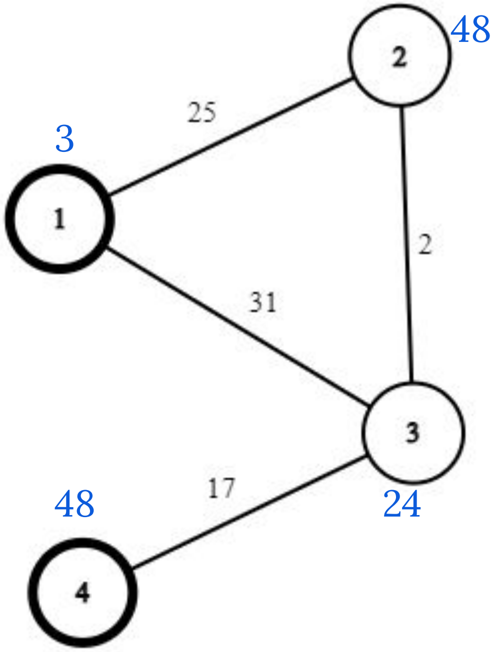
0 0 0

Output:

Case #1

92

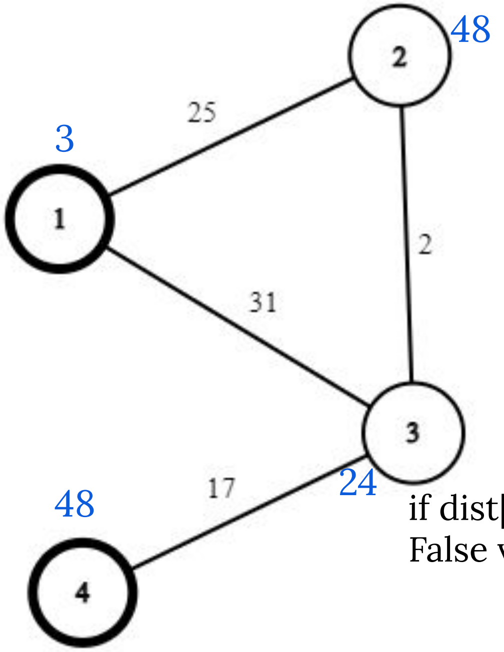
Khởi tạo dist[][], maxCost[][]



dist	1	2	3	4
1	0	25	31	oo
2	25	0	2	oo
3	31	2	0	17
4	oo	oo	17	0

maxCost	1	2	3	4
1	3	48	24	48
2	48	48	48	48
3	24	48	24	48
4	48	48	48	48

Floyd lần 1, k = 2, i = 1, j = 3

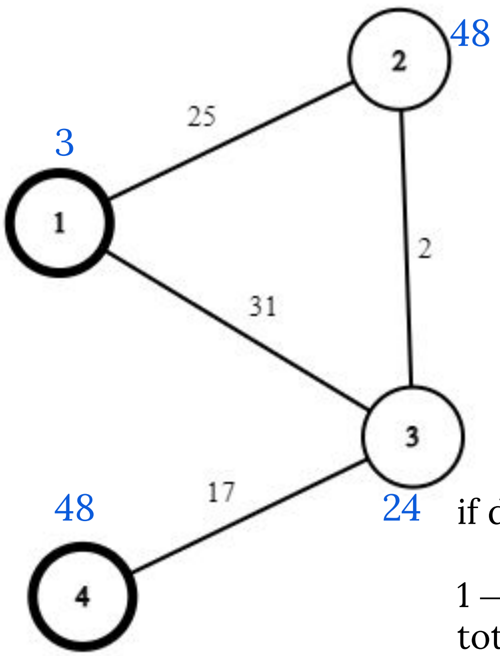


dist	1	2	3	4
1	0	25	31	oo
2	25	0	2	oo
3	31	2	0	17
4	oo	oo	17	0

maxCost	1	2	3	4
1	3	48	24	48
2	48	48	48	48
3	24	48	24	48
4	48	48	48	48

if dist[1][3] + maxCost[1][3] > dist[1][2] + dist[2][3] + max(maxCost[1][2], maxCost[2][3])
False vì 31 + 24 < (25 + 2) + 48

Floyd lần 1, k = 3



dist	1	2	3	4
1	0	25	31	oo→ 48
2	25	0	2	00→ 19
3	31	2	0	17
4	oo→ 48	00→ 19	17	0

maxCost	1	2	3	4
1	3	48	24	48
2	48	48	48	48
3	24	48	24	48
4	48	48	48	48

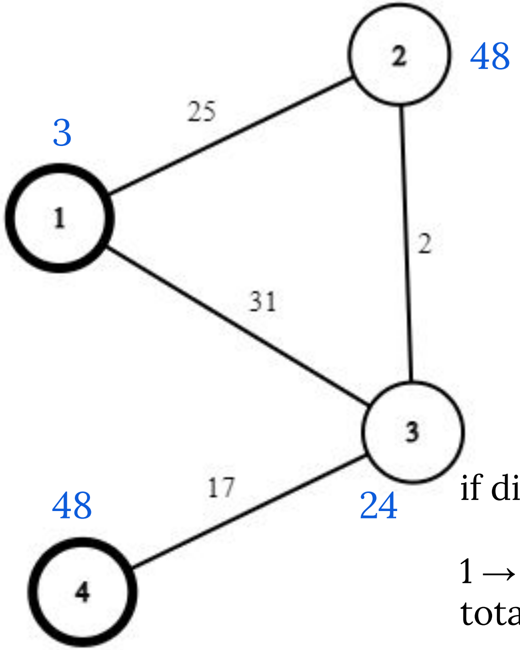
if dist[1][4] + maxCost[1][4] > dist[1][3] + dist[3][4] + max(maxCost[1][3], maxCost[3][4]):

dist[1][4] = dist[1][3] + dist[3][4] = 31 + 17 = 48

1 → 3 → 4

totalCost[1][4] = dist[1][4] + maxCost[1][4] = 48 + 48 = 96

Floyd lần 2, k = 2



dist	1	2	3	4
1	0	25	31	44
2	25	0	2	19
3	31	2	0	17
4	44	19	17	0

maxCost	1	2	3	4
1	3	48	24	48
2	48	48	48	48
3	24	48	24	48
4	48	48	48	48

if dist[1][4] + maxCost[1][4] > dist[1][2] + dist[2][4] + max(maxCost[1][2], maxCost[2][4]):

dist[1][4] = dist[1][2] + dist[2][4] = 25 + 19 = 44

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

totalCost[1][4] = dist[1][4] + maxCost[1][4] = 44 + 48 = **92**

Lời giải

Bước 1: Khởi tạo mảng **dist[][]**, **maxCost[][]** với **maxCost[i][j] = max(f[i],f[j]).**

Bước 2: Chạy thuật toán Floyd

Nếu **dist[i][j] + maxCost[i][j] > dist[i][k] + dist[k][j] + max(maxCost[i][k], maxCost[k][j]):**

Cập nhật **dist[i][j] = dist[i][k] + dist[k][j]**

maxCost[i][j] = max(maxCost[i][k], maxCost[k][j])

Bước 3: Chạy Floyd *nhiều lần* để đảm bảo 2 đại lượng **dist**, **maxCost** đều tối ưu.

Bước 4: In kết quả cho chi phí từ thành phố **i** đến **j** là **dist[i][j] + maxCost[i][j]**.

Mã giả

Mã giả

```
MAX = 85
while True:
    read C,R,Q
    read cost[]
    dist = [[INF]*MAX]*MAX
    for i = 1 → R:
        read u,v,w
        dist[u][v] = w
        dist[v][u] = w
    maxCost = [[0]*MAX]*MAX
    for i = 1 → C:
        for j = 1 → C:
            maxCost[i][j] = max(cost[i],cost[j])
    updated = Floyd()
    while (updated == True):
        updated = Floyd()
```

```
for i = 1 → Q:
    read u,v
    if dist[u][v] == INF:
        print(-1)
    else:
        print(dist[u][v] + maxCost[u][v])
```

Độ phức tạp: O(T*(k*C^3 + Q)) = O(T*C^3)

Mã giả

```
function Floyd():
    updated = False
    for k = 1 → C:
        for i = 1 → C:
            if dist[i][k] == INF: continue
            for j = 1 → C:
                if dist[k][j] == INF: continue
                if dist[i][j]+maxCost[i][j] > dist[i][k]+dist[k][j]+
max(maxCost[i][k], maxCost[k][j]):
                    dist[i][j] = dist[i][k]+dist[k][j]
                    maxCost[i][j] = max(maxCost[i][k], maxCost[k][j])
                    updated = True
    return updated

True: lần chạy vừa rồi có sự cập nhật, -> tiếp tục chạy Floyd

False: không có sự cập nhật, kết quả đã tối ưu, dừng
```