Commandos

Big-O Blue - Lecture 8: Dijkstra

Tóm tắt đề bài

Tóm tắt đề bài

- Gồm một quân đoàn đang tiến hành làm nhiệm vụ phá các trụ sở của kẻ địch.
- Tất cả quân đều di chuyển đồng loạt với tốc độ như nhau (1 đơn vị thời gian trên mỗi cạnh).
- Lộ trình di chuyển đều như nhau: bắt đầu từ vị trí S đi đến các mục tiêu, sau đó quay về vị trí D.
- Nhiệm vụ kết thúc khi quân cuối cùng hoàn thành và về đến vị trí D.
- Yêu cầu: Thời gian ít nhất để hoàn thành nhiệm vụ phá trụ sở của kẻ địch.

Mô tả Input/Output

Input:

Gồm T $(1 \leq T \leq 50)$ test case ở dòng đầu tiên, với mỗi test case:

- Hai dòng đầu chứa số N và R $(1 \leq N \leq 100)$ với N là số tòa nhà, R là số đường nối trực tiếp giữa các tòa.
- R dòng kế tiếp chứa các chỉ số u, v $(0 \leq u, v < N)$ là chỉ số các tòa.
- Dòng cuối cùng của test case sẽ là chỉ số vị trí S và D.

Output:

Với mỗi test case:

- In ra "Case {**id**}: {**answer**}" với **id** là chỉ số test case và **answer** là đáp án của test case tương ứng.

Giải thích ví dụ

Ví dụ 1

Input:

4

3

0 1

2 1

1 3

0 3

Output:

Case 1: 4

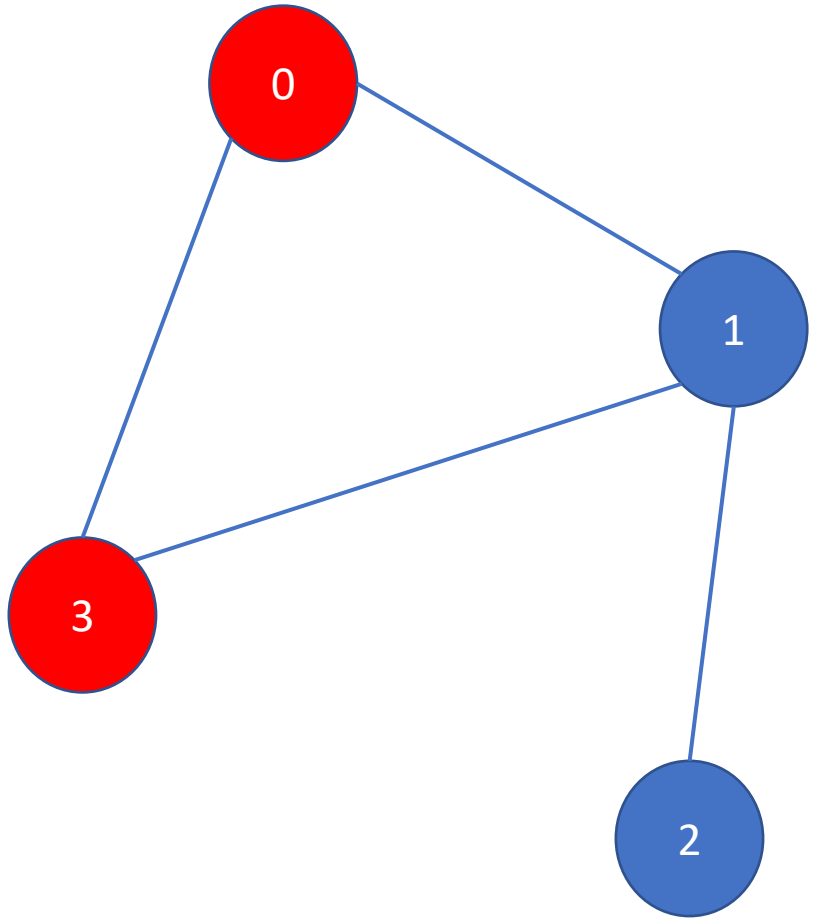
Giải thích:

Quân cuối cùng về vị trí 3 là quân phải đi phá tòa số 2.

Từ 0 → 2 tốn 2 (đvtg)

Từ 2 → 3 tốn 2 (đtvg)

➔ Tốn tổng cộng là 4 (đvtg)



Ví dụ 2

Input:

2

1

0 1

1 0

Giải thích:

Trong test case này không có tòa cần phải phá.

➔ Từ 0 → 1 tốn 1 (đvtg)

Output:

Case 2: 1

Hướng dẫn giải

Nhận xét

- Mình nhận xét từ đề cho rằng thời gian của nhiệm vụ là bằng với thời gian của quân cuối cùng hoàn thành nhiệm vụ của mình và về đến vị trí D.

➔ Chúng ta phải tìm ra người hoàn thành trễ nhất.

➔ Gọi **u** là tòa nhà của người hoàn thành trễ nhất phải phá, như vậy thời gian hoàn thành của người này sẽ bằng :

Thời gian **S → u → D** = thời gian từ **S đến tòa u +** thời gian từ **tòa u đến D.**

- Và để người trễ nhất này hoàn thành sớm nhất có thể.

➔ Chúng ta chọn ra đường đi có tổng thời gian nhỏ nhất từ **S** → mục tiêu **u → D**.

Nhận xét

- Ta đặt thời gian **S đến tòa u** = DistFromS[u].
- Tương tự, đặt thời gian **tòa u đến D** = DistFromD[u].
- Như vậy để tổng thời gian nhỏ nhất, ta phải có DistFromS[u] và DistFromD[u] phải là nhỏ nhất.

➔ Áp dụng thuật toán Dijkstra cho hai điểm bắt đầu **S** và **D** tìm đường đi ngắn nhất đến mọi tòa nhà **u**.

➔ Sau đó duyệt mỗi tòa u, ta chọn Max(DistFromS[u] + DistFromD[u]) làm đáp án.

- Ngoài ra, bài này cũng có thể không dùng Dijkstra, do mỗi cạnh có trọng số như nhau nên chúng ta cũng có thể **BFS** từ hai đỉnh **S** và **D** để tính.

Chạy ý tưởng thuật toán.

Input:

4

3

0 1

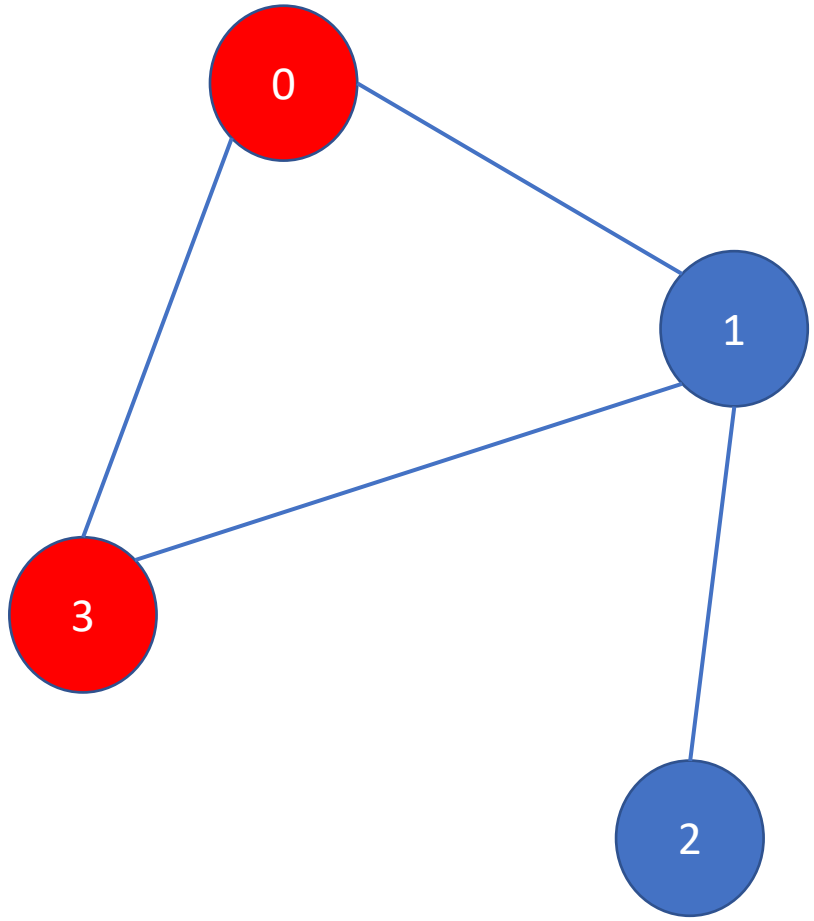
2 1

1 3

0 3

Output:

i	0	1	2	3
DistFromS[i]	0	INF	INF	INF
DistFromD[i]	INF	INF	INF	0



Chạy ý tưởng thuật toán (Sau khi chạy Dijkstra)

Input:

4

3

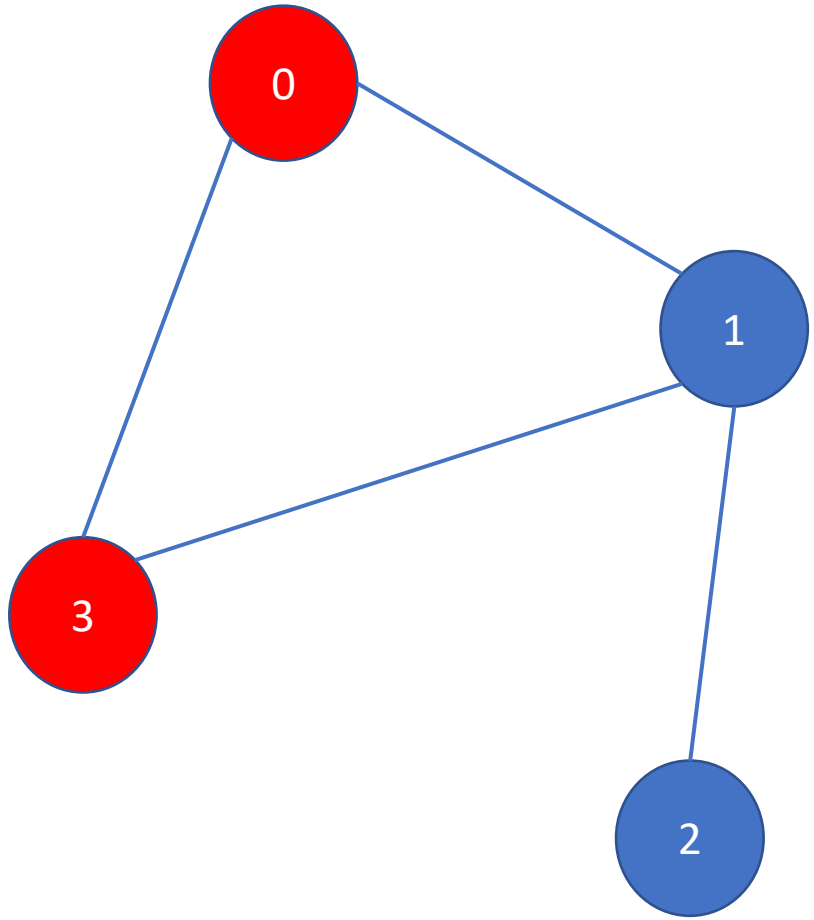
0 1

2 1

1 3

0 3

i	0	1	2	3
DistFromS[i]	0	1	2	1
DistFromD[i]	1	1	2	0



Output:

Chạy ý tưởng thuật toán (Sau khi chạy Dijkstra)

Input:

4

3

0 1

2 1

1 3

0 3

Output:

4

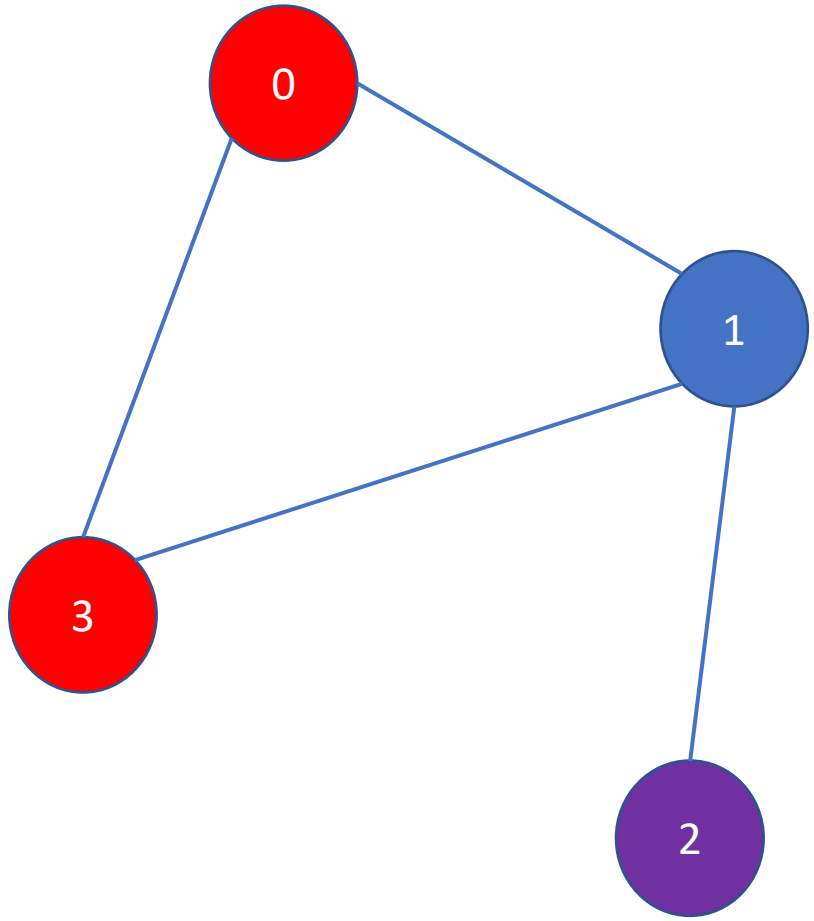
i	0	1	2	3
DistFromS[i]	0	1	2	1
DistFromD[i]	1	1	2	0

Ta thấy được quân đến tòa 2 là người có tổng thời gian lâu nhất.

➔ 0 → 2 = 2.

➔ 2 → 3 = 2.

➔ 0 → 2 → 3 = 4.



Các bước giải

Bước 1: Đọc vào dữ liệu đề cho, số đỉnh N, số cạnh R và hai đỉnh S và D.

Bước 2: Chạy thuật toán Dijkstra từ đỉnh S, lưu mảng DistFromS. Tương tự chạy Dijkstra từ đỉnh D, lưu lại mảng DistFromD.

Bước 3: Khởi tạo biến answer = 0, duyệt mỗi tòa nhà i : 0 → N – 1

Nếu answer < DistFromS[i] + DistFromD[i] thì

cập nhật đáp án answer = DistFromS[i] + DistFromD[i]

Bước 4: Ghi ra đáp án answer của test case tương ứng.

Độ phức tạp: $\boldsymbol{O(T \times (Rlog(N) + N))}$

Mã giả

Mã giả

```
Read(T)
INF = 1e9
For test in range(1, T + 1):
  read(N)
  read(R)
  Graph = [[] for i in range(N)]
  for i in range(R):
    read(u, v)
    Graph[u].append(v)
    Graph[v].append(u)
  read(S, D)
  DistFromS = Dijkstra(S)
  DistFromD = Dijkstra(D)
  # To be continue
```

Mã giả

```
answer = 0
for i in range(N):
  if answer < (DistFromS[i] + DistFromD[i]):
    answer = DistFromS[i] + DistFromD[i]
print('Case {test}: {answer}')
```

Mã giả

```
Func Dijkstra(start):
 dist = [INF, INF, INF, …]
 dist[start] = 0
 Heap = []
 Heap.push((0, start)) // Heap lưu một pair thông tin (dist, node)
 while (Heap.empty() == false):
  DistToU, u = Heap.get()
  if (DistToU != dist[u]):
   continue
  for v in graph[u]:
   if (dist[v] > dist[u] + 1):
    dist[v] = dist[u] + 1
    Heap.push((dist[v], v))
 return dist; // trả về là một mảng kết quả của Dijkstra.
```

Mã giả (nếu thay Dijkstra bằng BFS)

```
Func BFS(start): // Độ phức tạp của thuật toán sẽ là O(T * (R + N))
 dist = [INF, INF, INF, …]
 dist[start] = 0
 queue = []
 queue.push(start) // Queue lưu thông tin đỉnh.
 while (queue.empty() == false):
  u = queue.get()
  for v in graph[u]:
   if (dist[v] > dist[u] + 1):
    dist[v] = dist[u] + 1
    queue.push(v)
 return dist;
```

Thank you