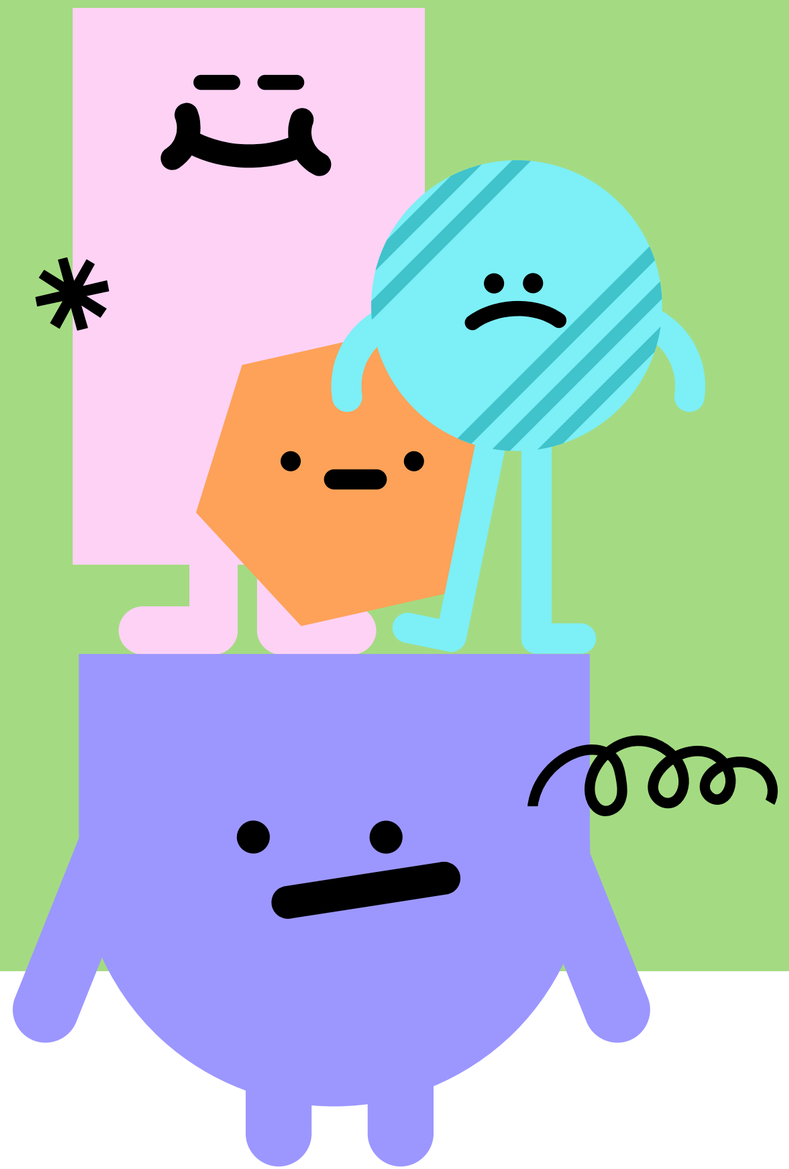
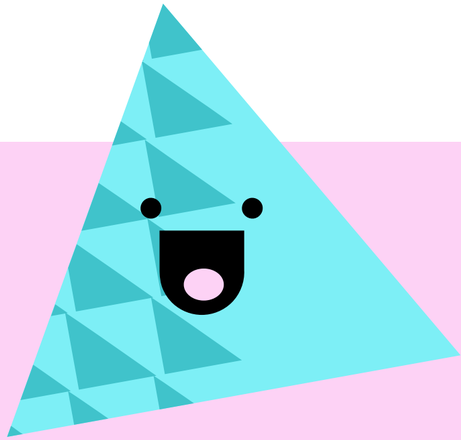
NHÓM 5

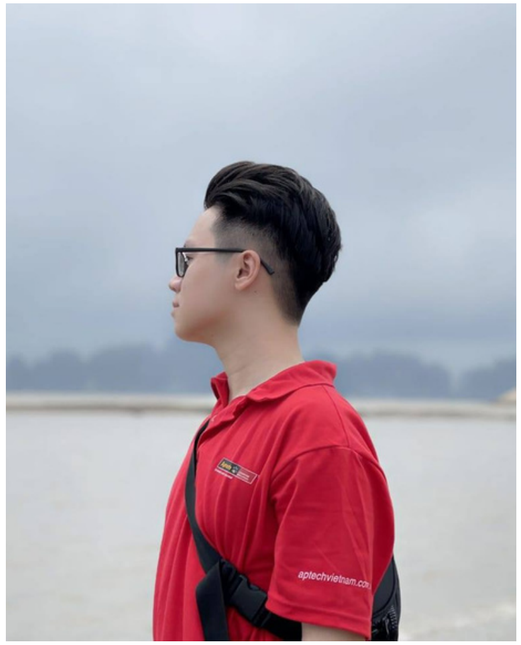


Ly



XIN CHÀO,
CÁC BẠN CÙNG LỚP!
CHÚNG TÔI LÀ NHÓM 5!

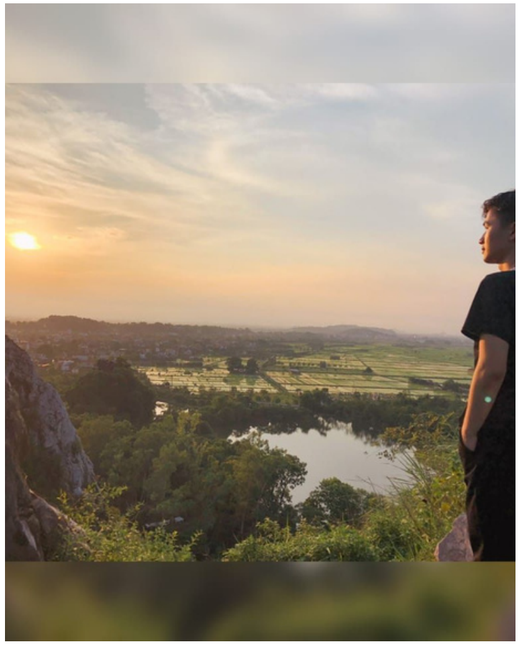
THÀNH VIÊN NHÓM



Quang Phúc

Hoàng Vũ

Quang Phú



Đức Trung

MỤC TIÊU

- BỐI CẢNH RA ĐỜI

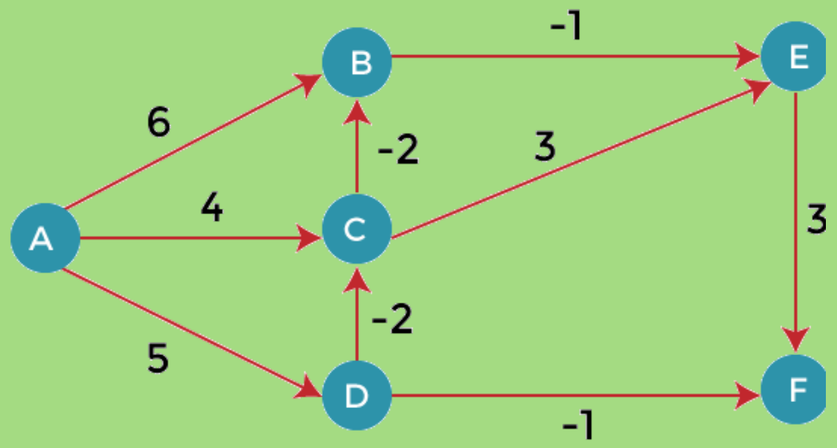
- CÁC BƯỚC THUẬT TOÁN VÀ CÁCH HOẠT ĐỘNG
- MÃ GIẢ
- CÀI ĐẶT BẰNG C++
- ĐỘ PHỨC TẠP THUẬT TOÁN
- TÍNH ĐÚNG ĐẮN

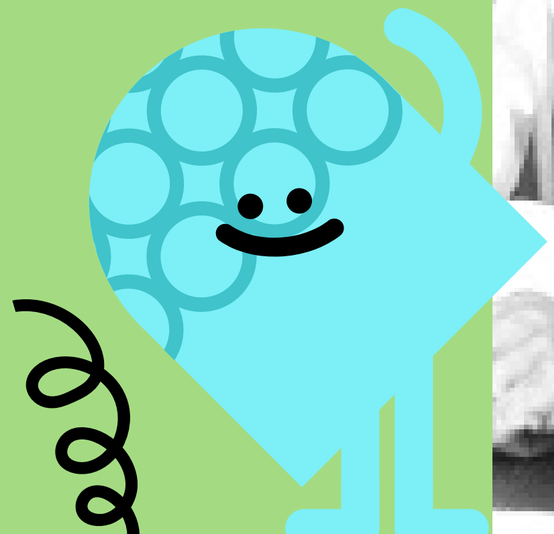
- HÌNH MINH HOẠ
- VÍ DỤ ÁP DỤNG

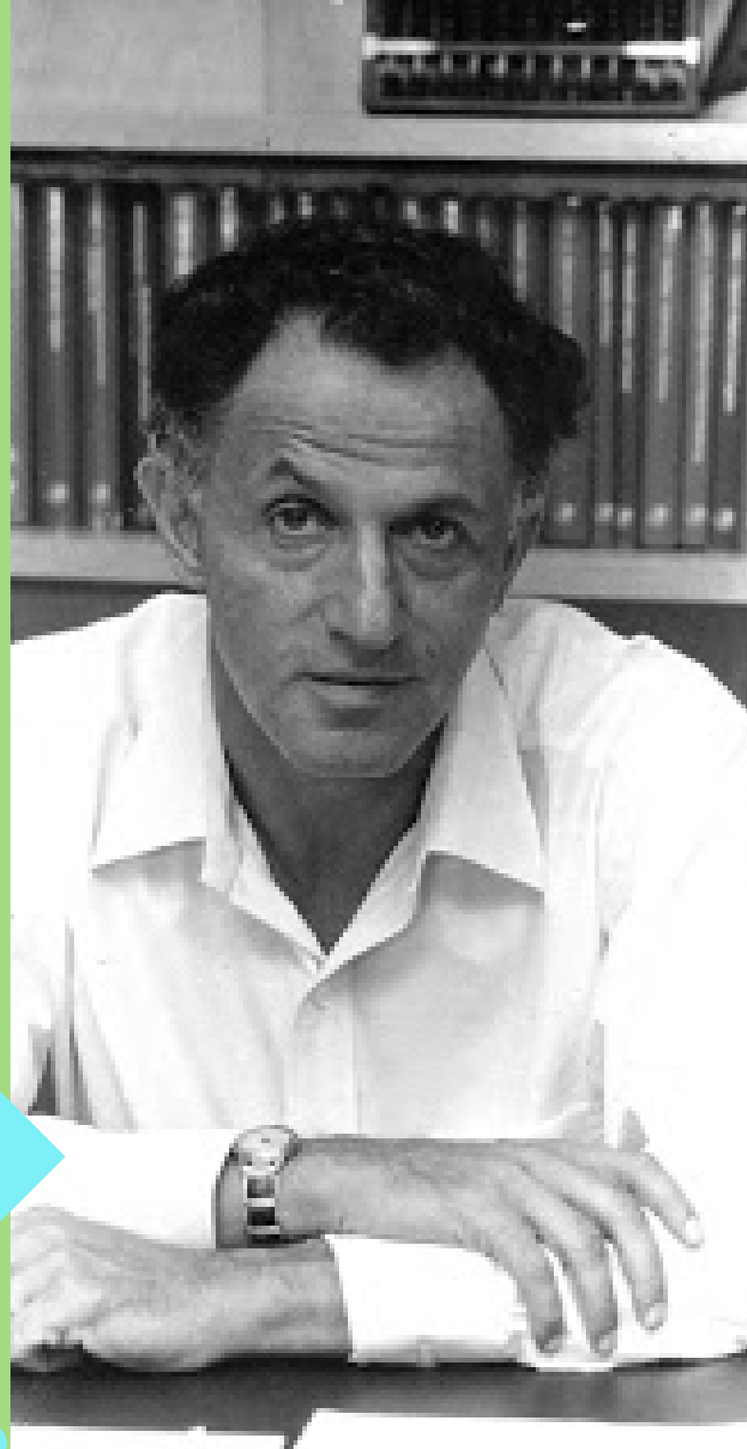
BỐI CẢNH XUẤT XỨ

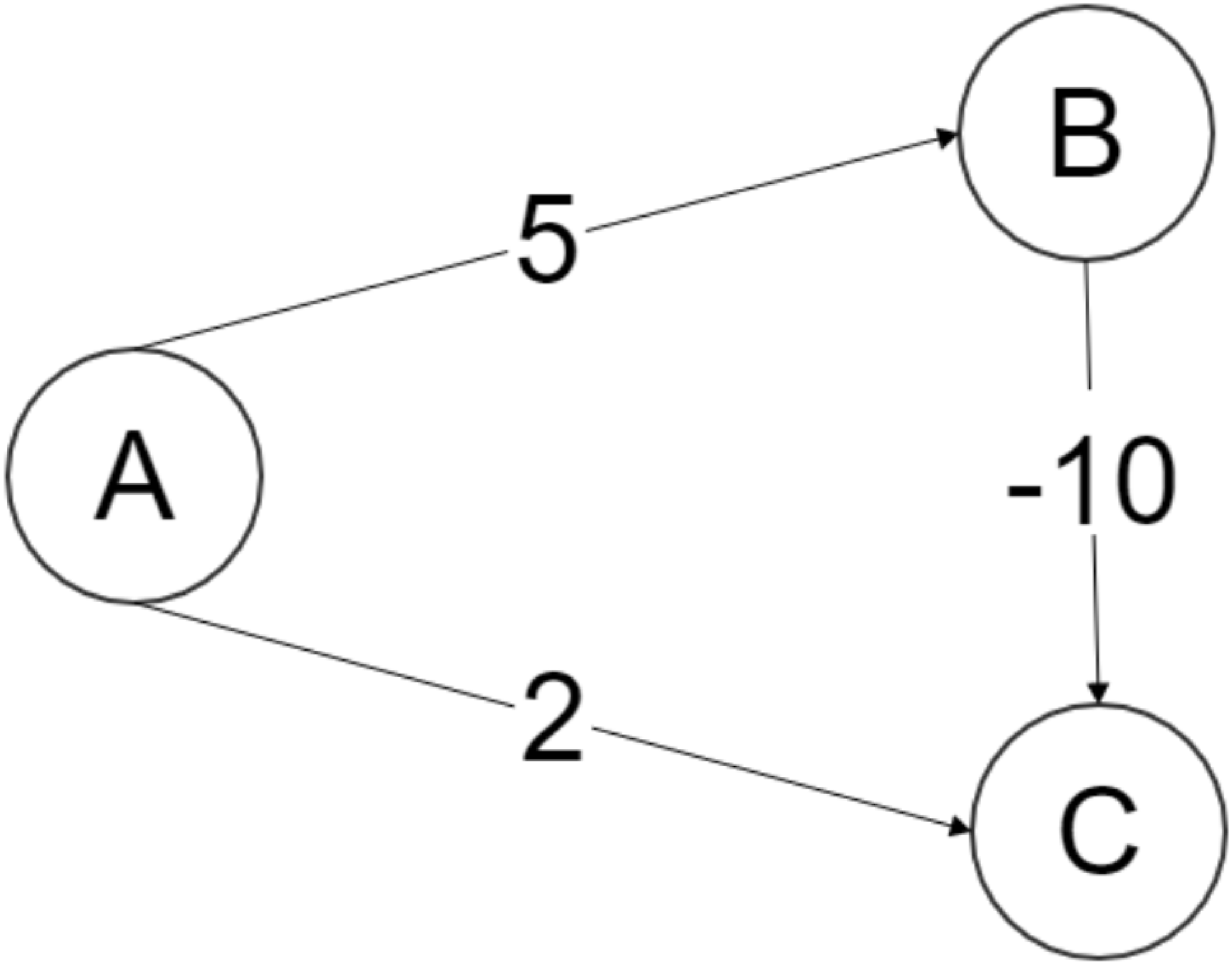
TỔNG QUAN

- **TÁC DỤNG**
- **ƯU ĐIỂM**









Tại sao

Dijktra không thể sử dụng được với trọng số âm?

CÁC BƯỚC THUẬT TOÁN

BƯỚC 1

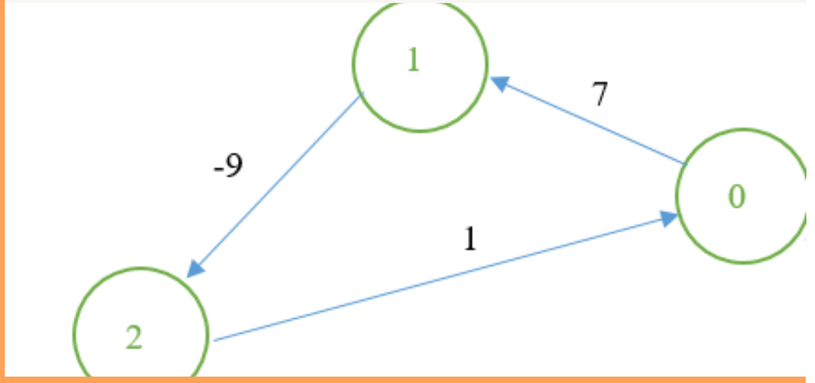
```
function bellmanFord(G, S)
  for each vertex V in G
    distance[V] <- infinite
      previous[V] <- NULL
  distance[S] <- 0
```

BƯỚC 2

```
for each vertex V in G
check <- false
for each edge (U,V) in G
tempDistance <- distance[U] + edge_weight(U, V)
if tempDistance < distance[V]
distance[V] <- tempDistance
previous[V] <- U
check <- true
endif
if (!check) then break the loop
```

BƯỚC 3

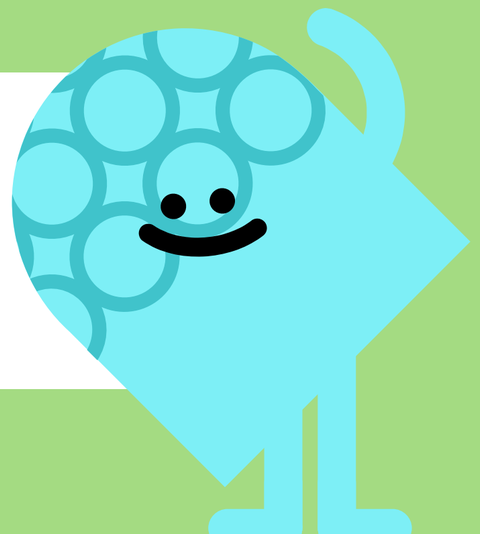
```
 for each edge (U,V) in G
If distance[U] + edge_weight(U, V) < distance[V}
  Error: Negative Cycle Exists
```

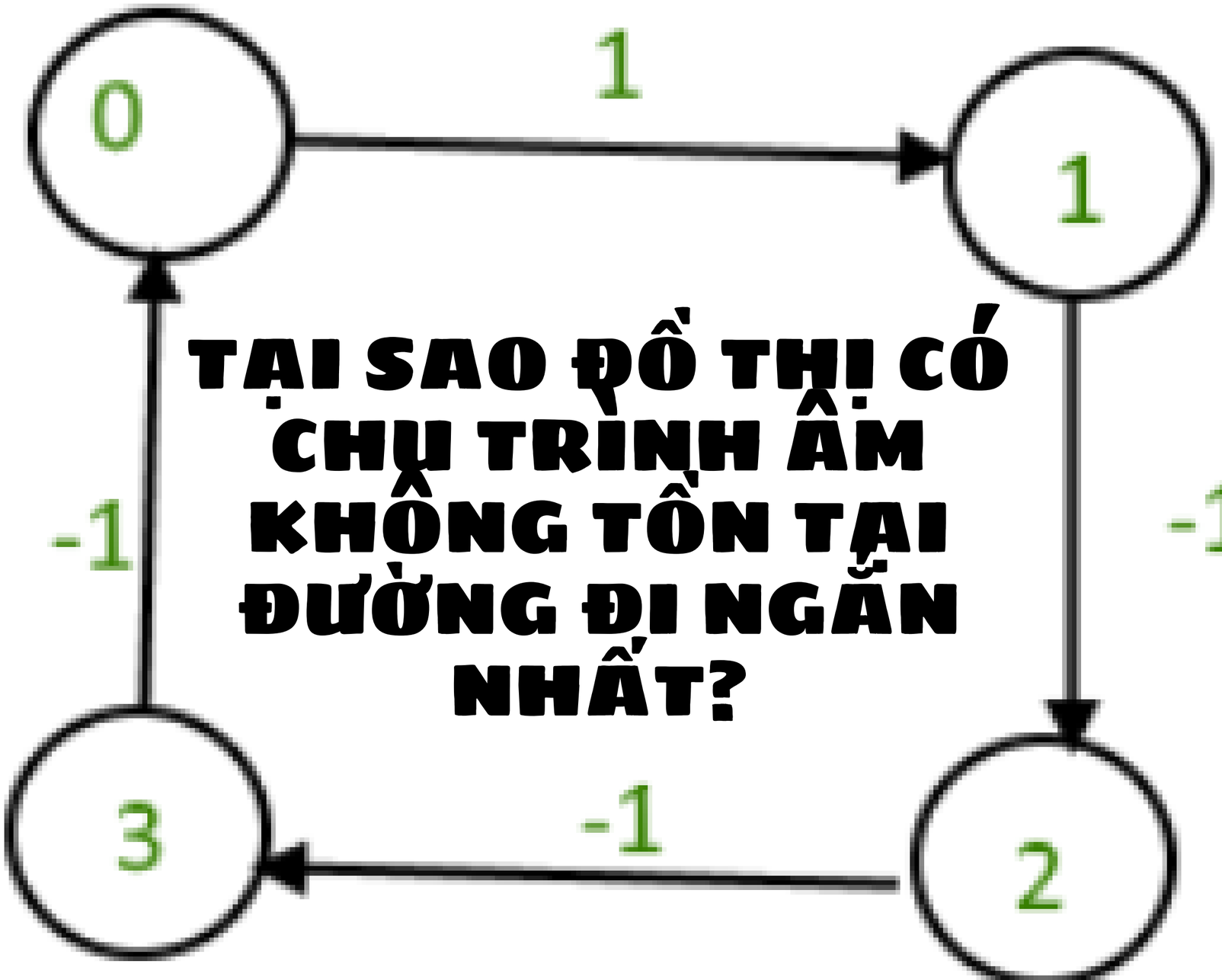


VẤN ĐỀ THUẬT TOÁN

TẠI SAO ĐỒ THỊ CÓ CHU TRÌNH ÂM KHÔNG TỒN TẠI ĐƯỜNG ĐI NGẮN NHẤT?

CẢI THIỆN THUẬT TOÁN





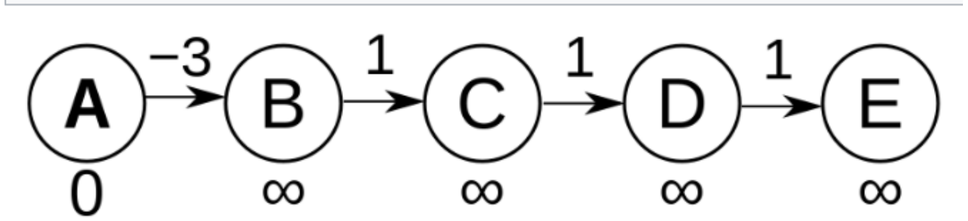
TẠI SAO ĐỒ THỊ CÓ CHU TRÌNH ÂM KHÔNG TỒN TẠI ĐƯỜNG ĐI NGẮN NHẤT?

Cải tiến thuật toán Bellman Ford

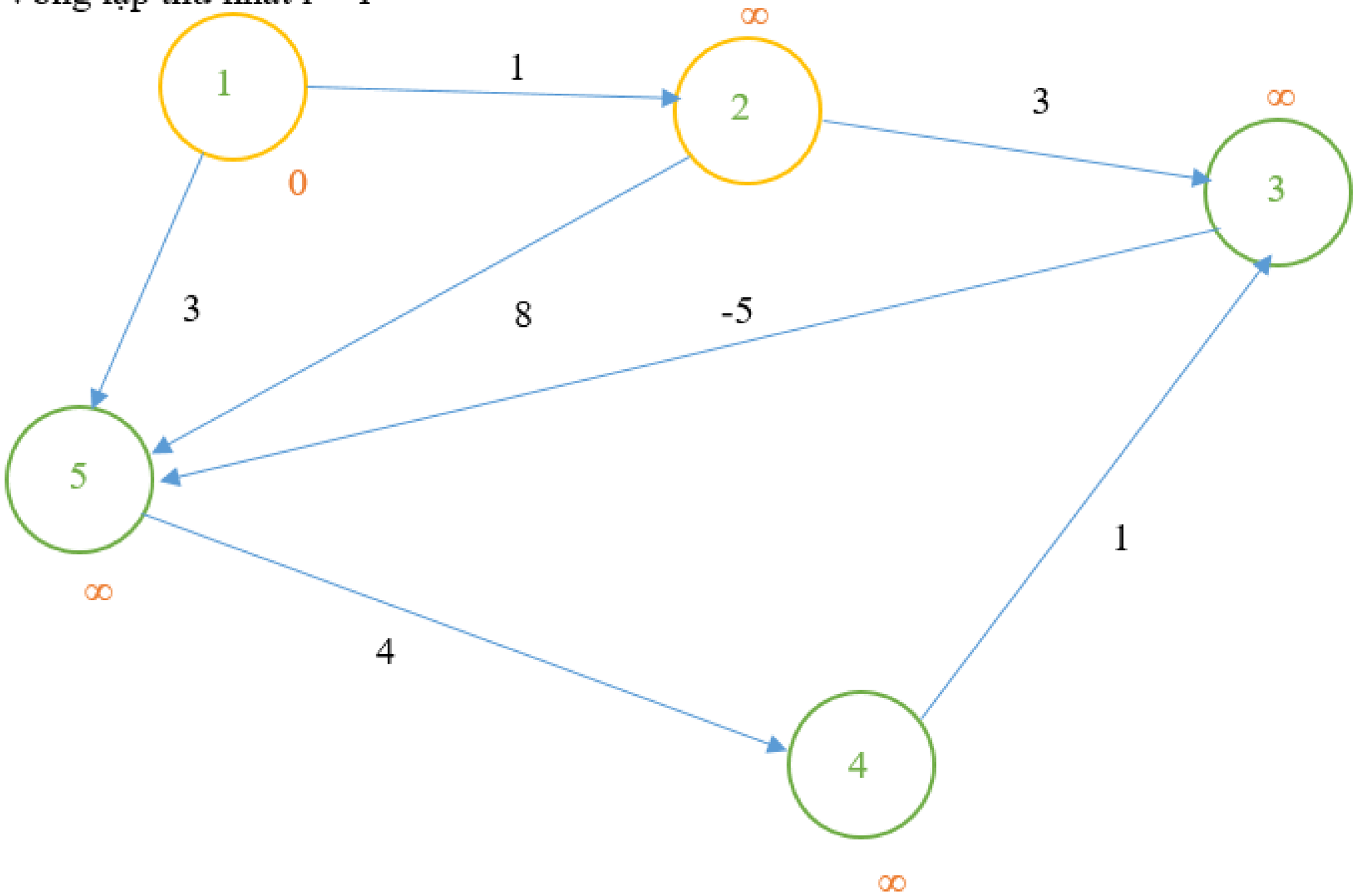
```
for each vertex V in G
    check <- false
    for each edge (U,V) in G
      tempDistance <- distance[U] + edge_weight(U, V)
      if tempDistance < distance[V]
        distance[V] <- tempDistance
        previous[V] <- U
        check <- true
      endif
    if (!check) then break the loop
```

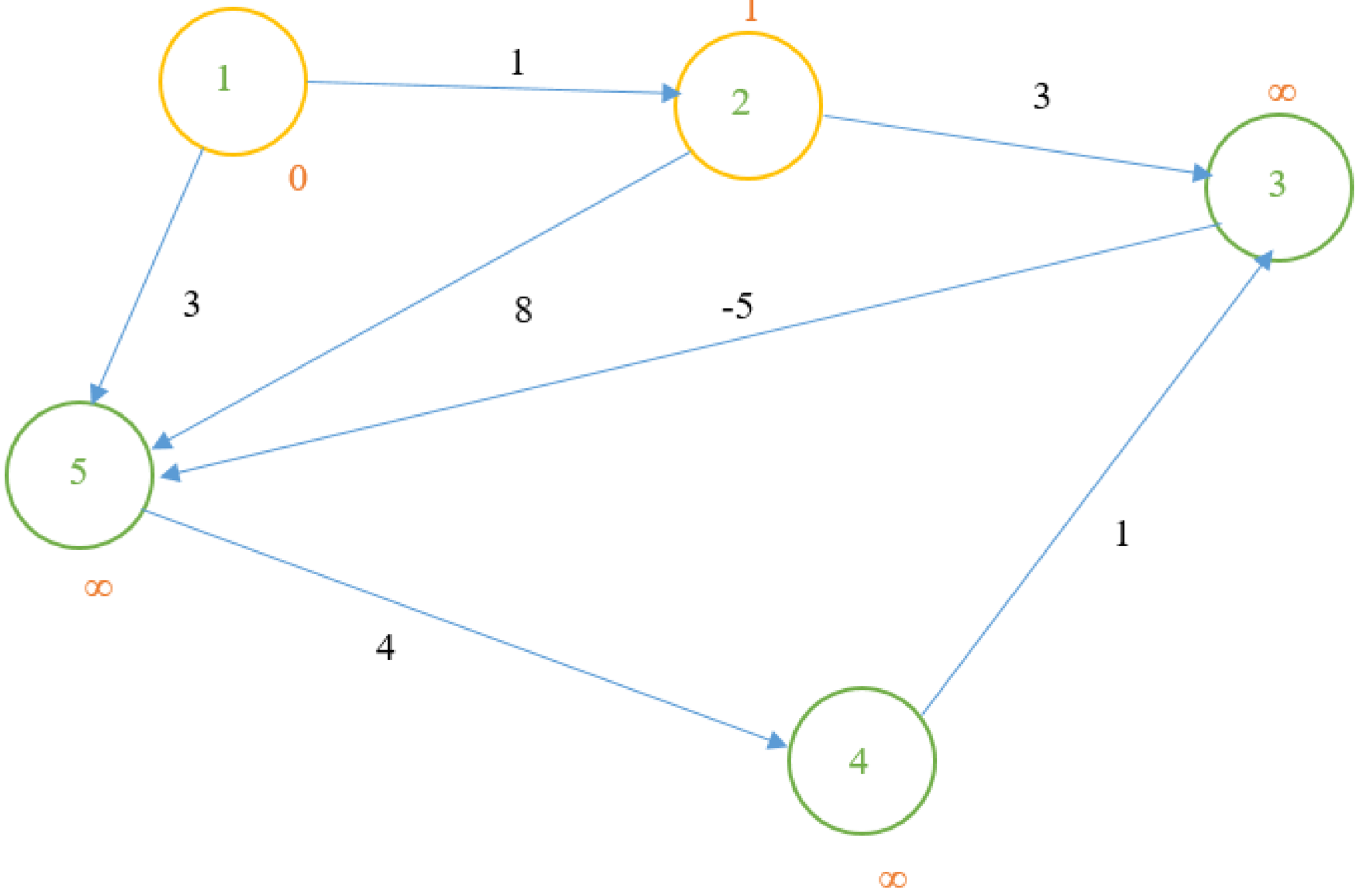
ĐỘ PHỨC TẠP THUẬT TOÁN

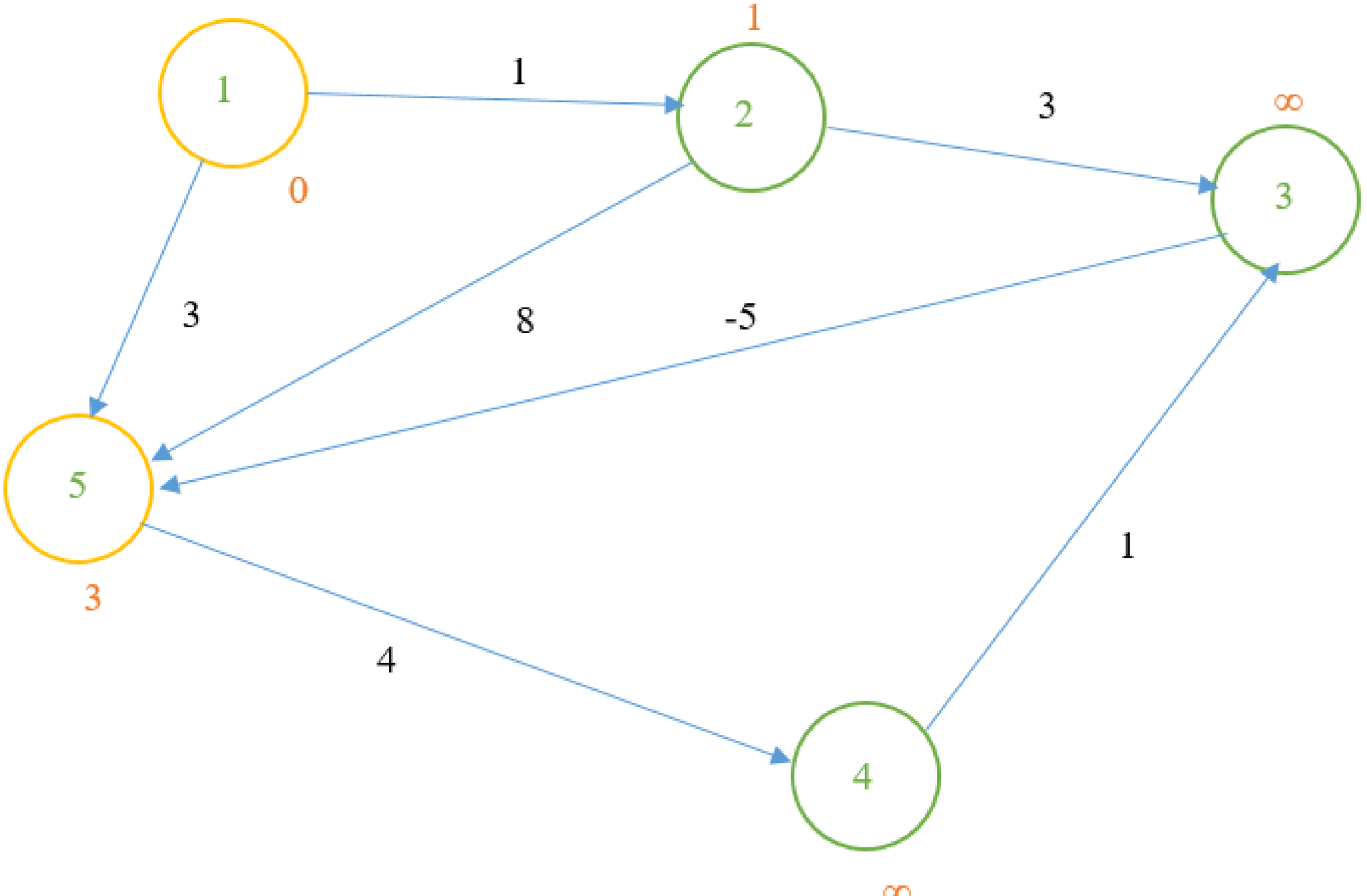
* Trường hợp tốt nhất O(2E)

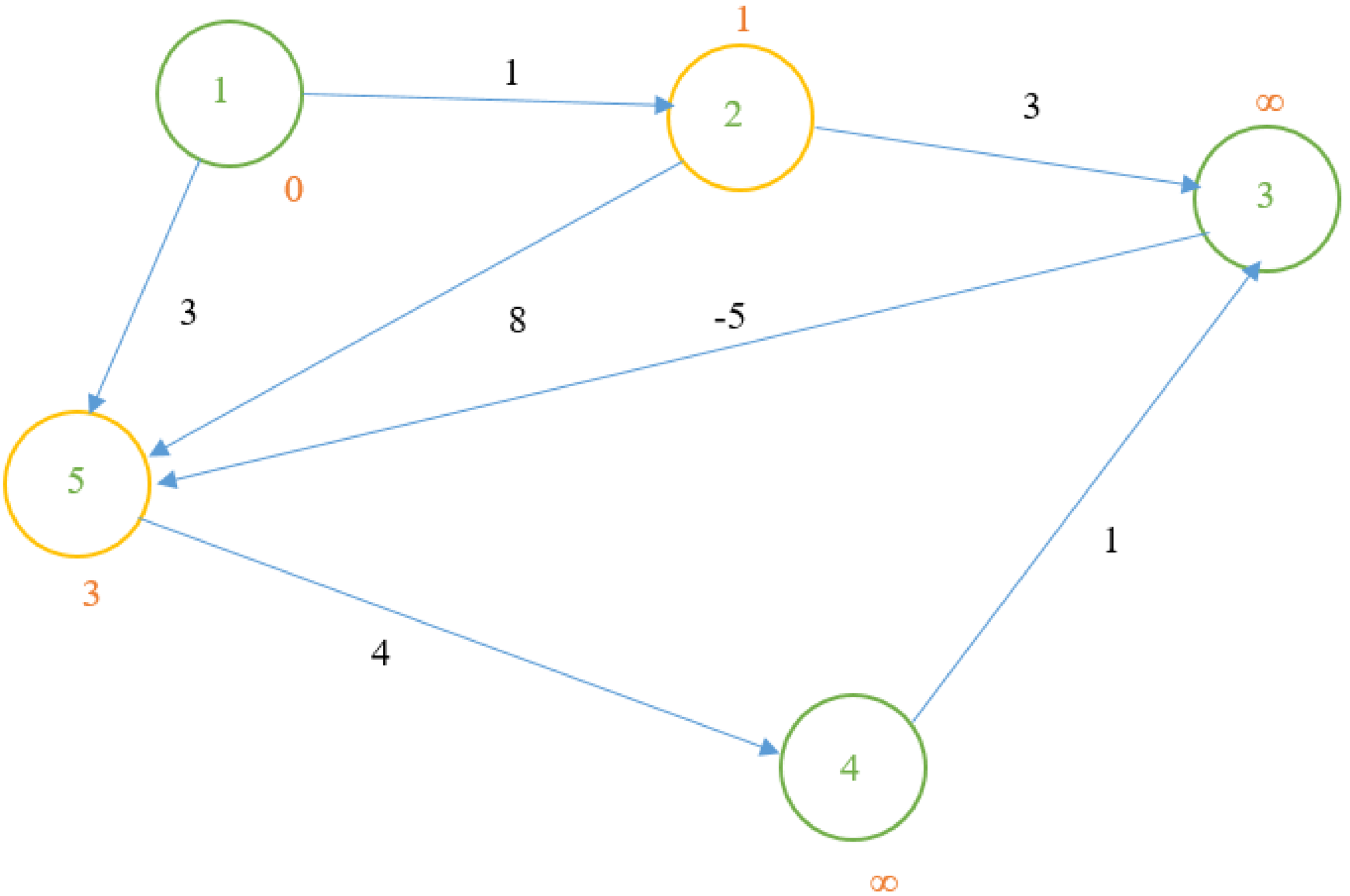


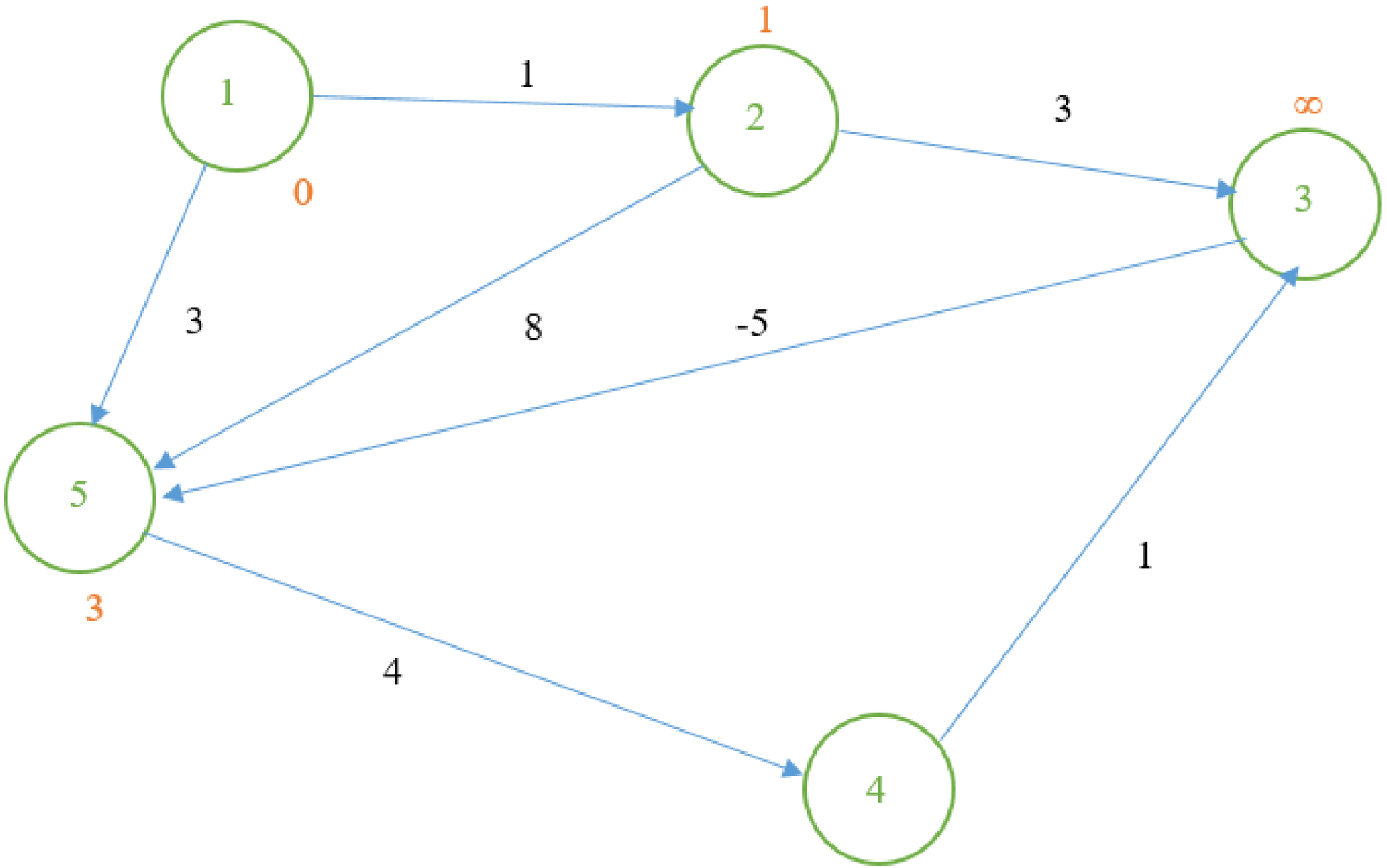
- Vòng lặp thứ nhất i = 1

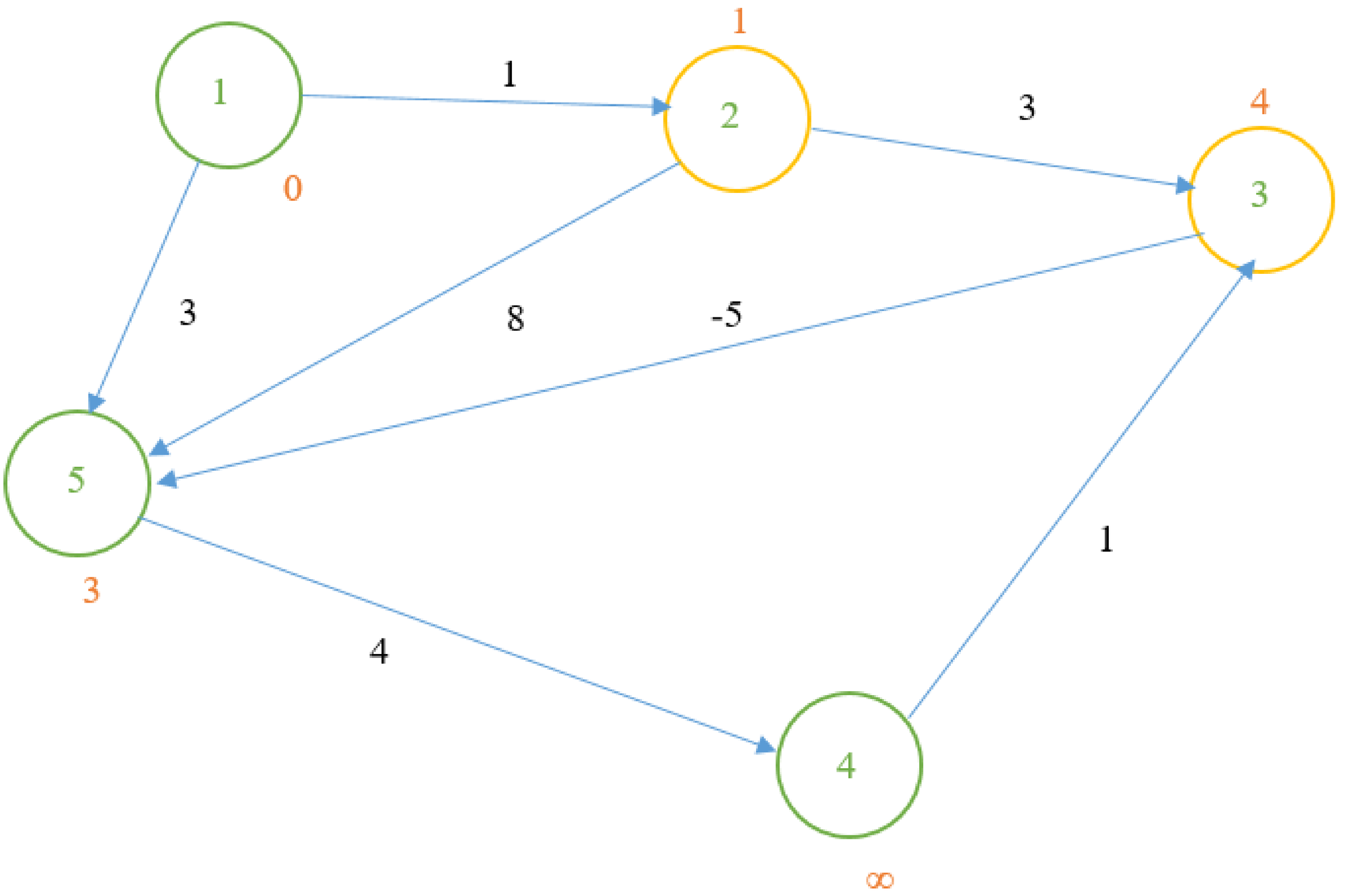


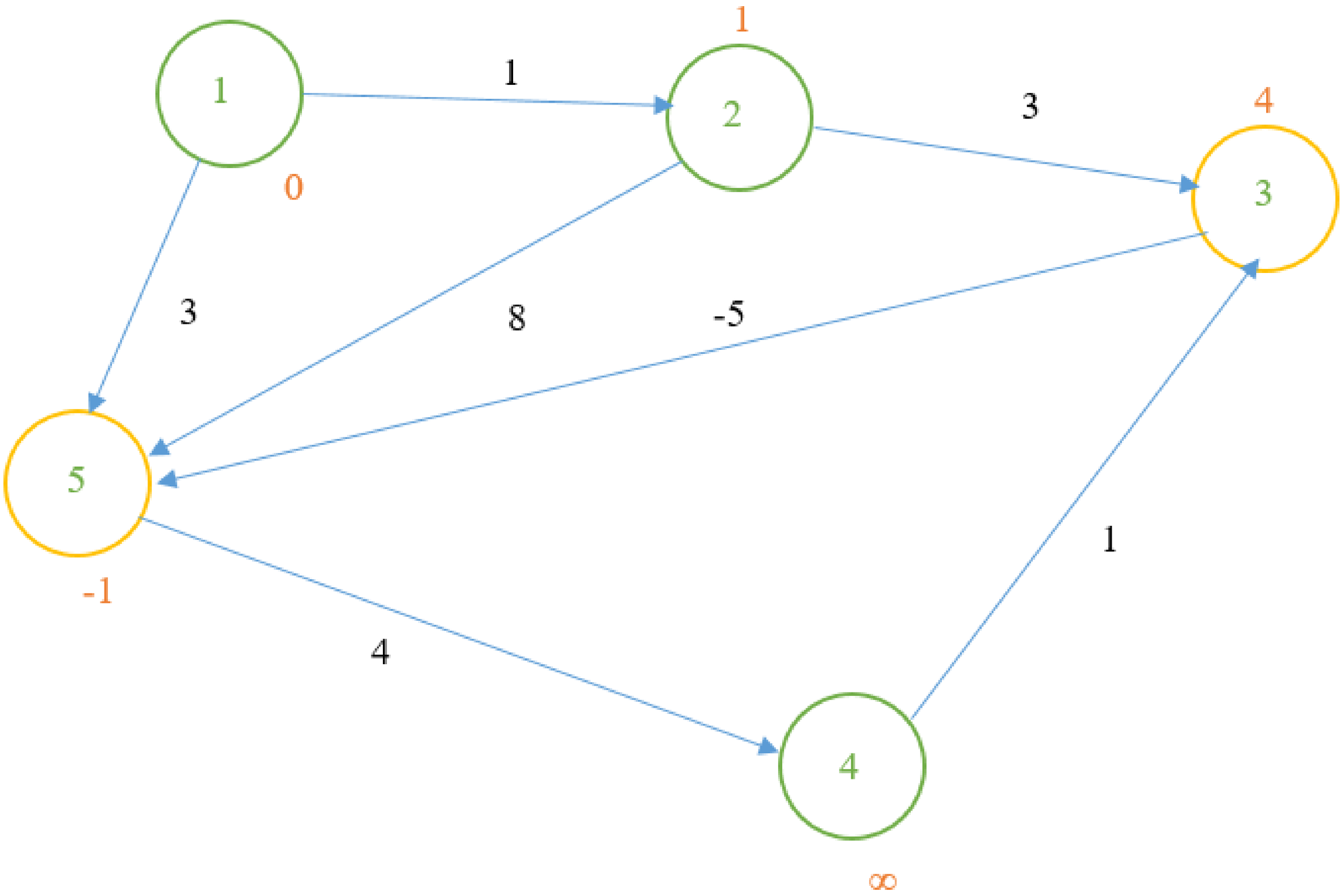

1
2
3
4
5
1
0
∞
∞
∞
1
3
8
-5
3
4
1

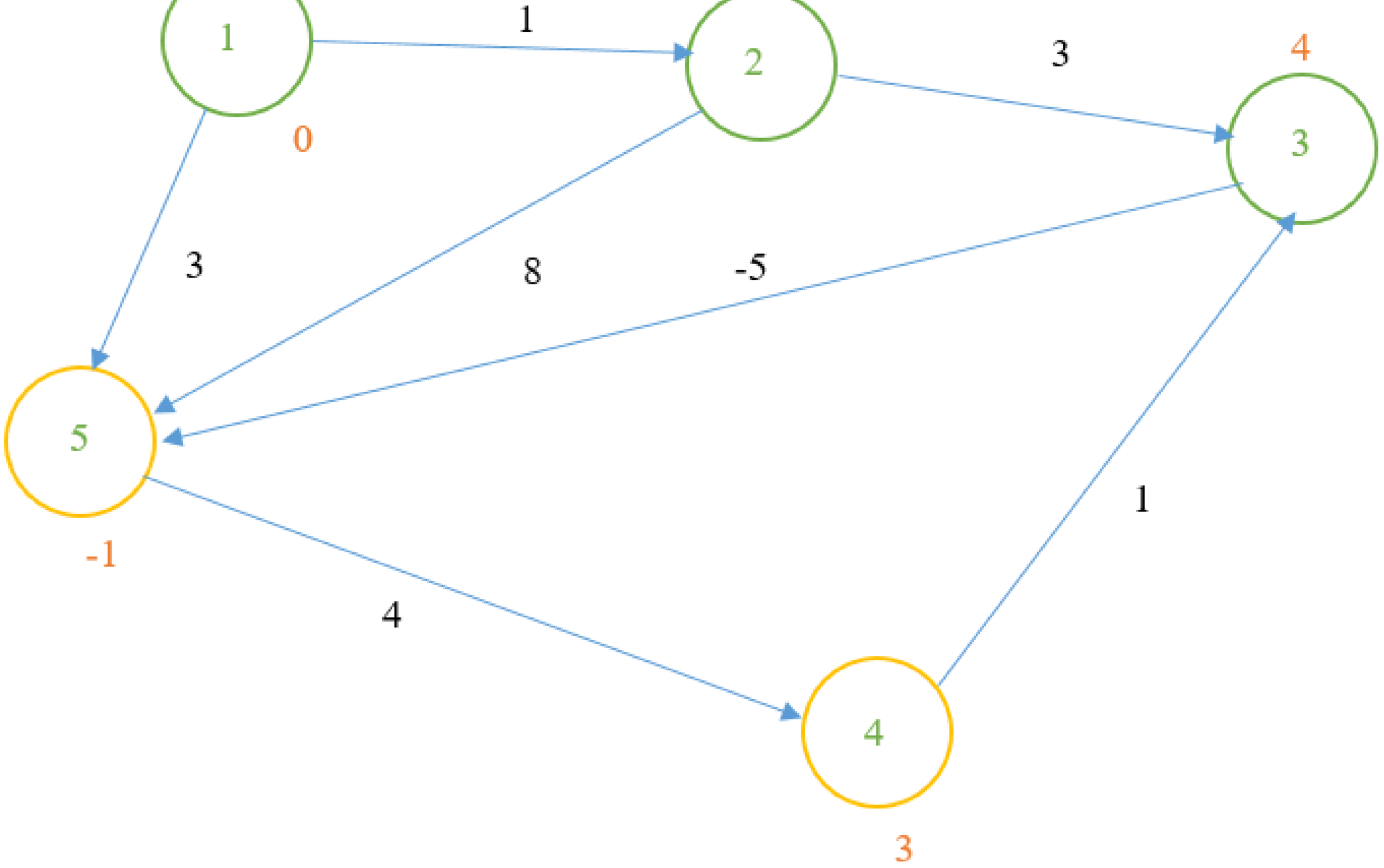

1
2
3
4
5
1
0
∞
∞
3
1
3
8
-5
4
3

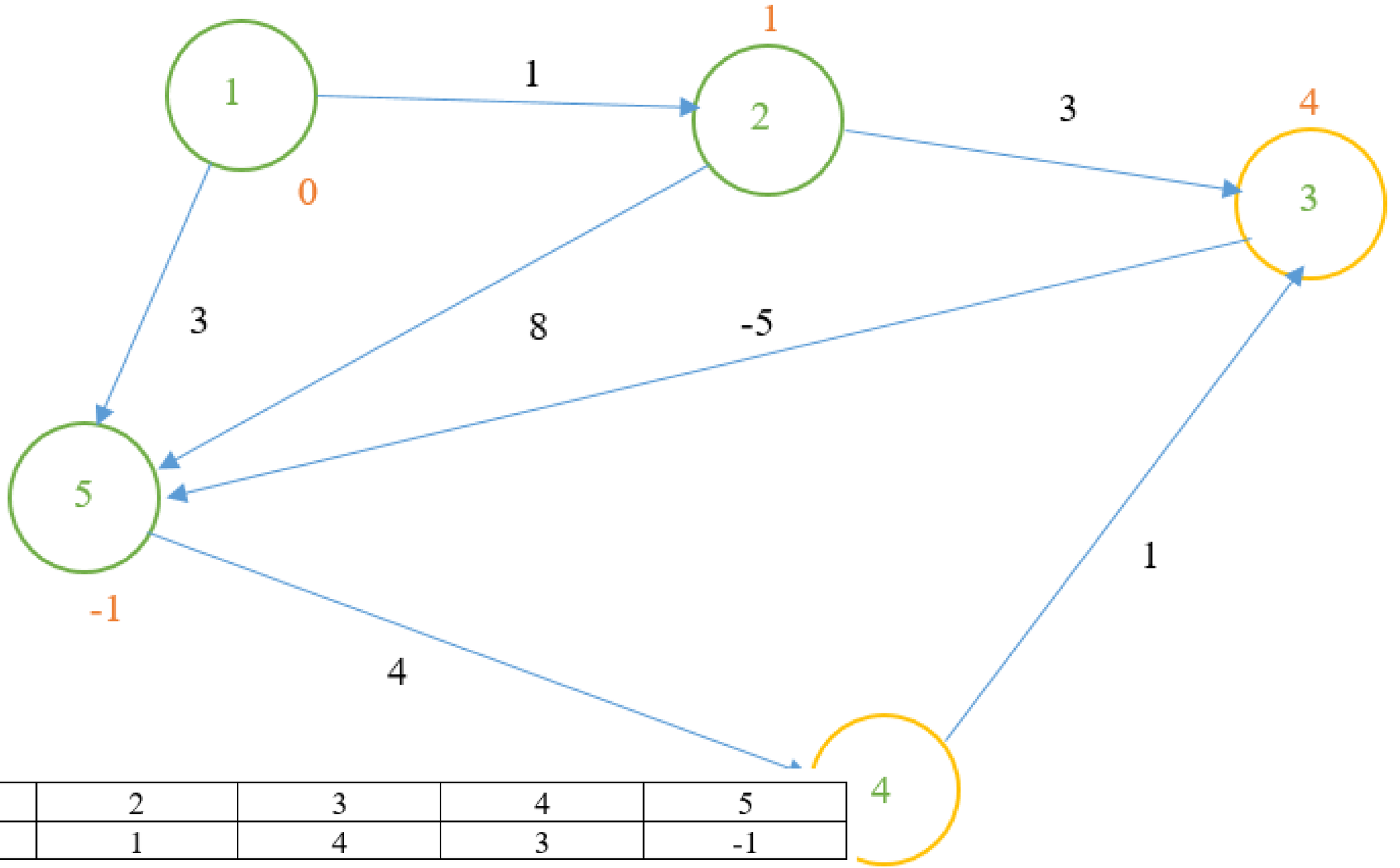

1
2
3
4
5
1
0
∞
∞
3
1
3
1
3
8
-5
4


1
2
3
4
5
1
0
∞
3
1
3
8
-5
4
1


1
2
3
4
5
1
0
1
3
4
3
8
-5
3
4
1
∞

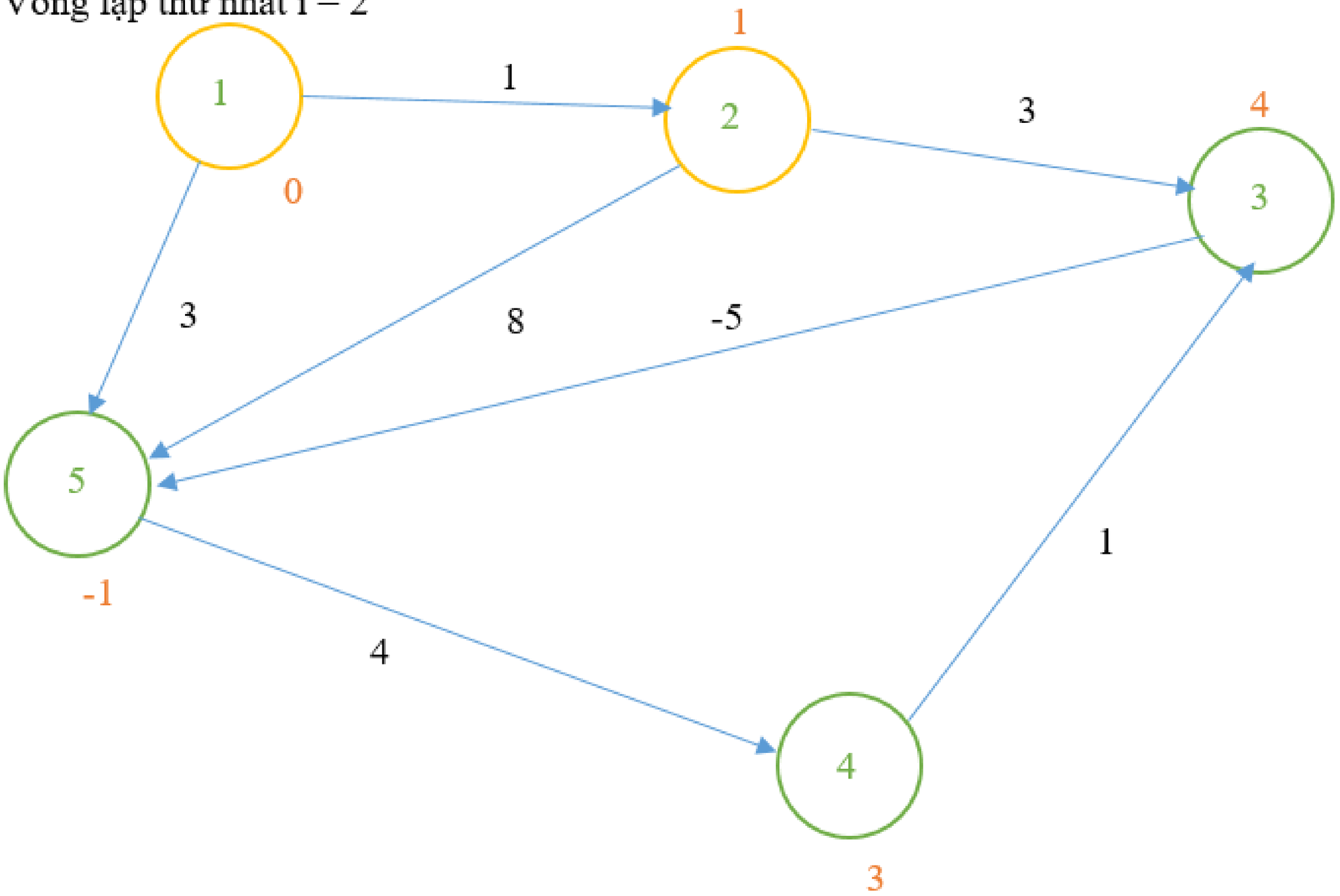

1
2
3
4
5
1
0
4
∞
-1
1
3
3
8
-5
4
1


1
2
3
4
5
1
3
8
-5
3
4
1
0
4
-1
3

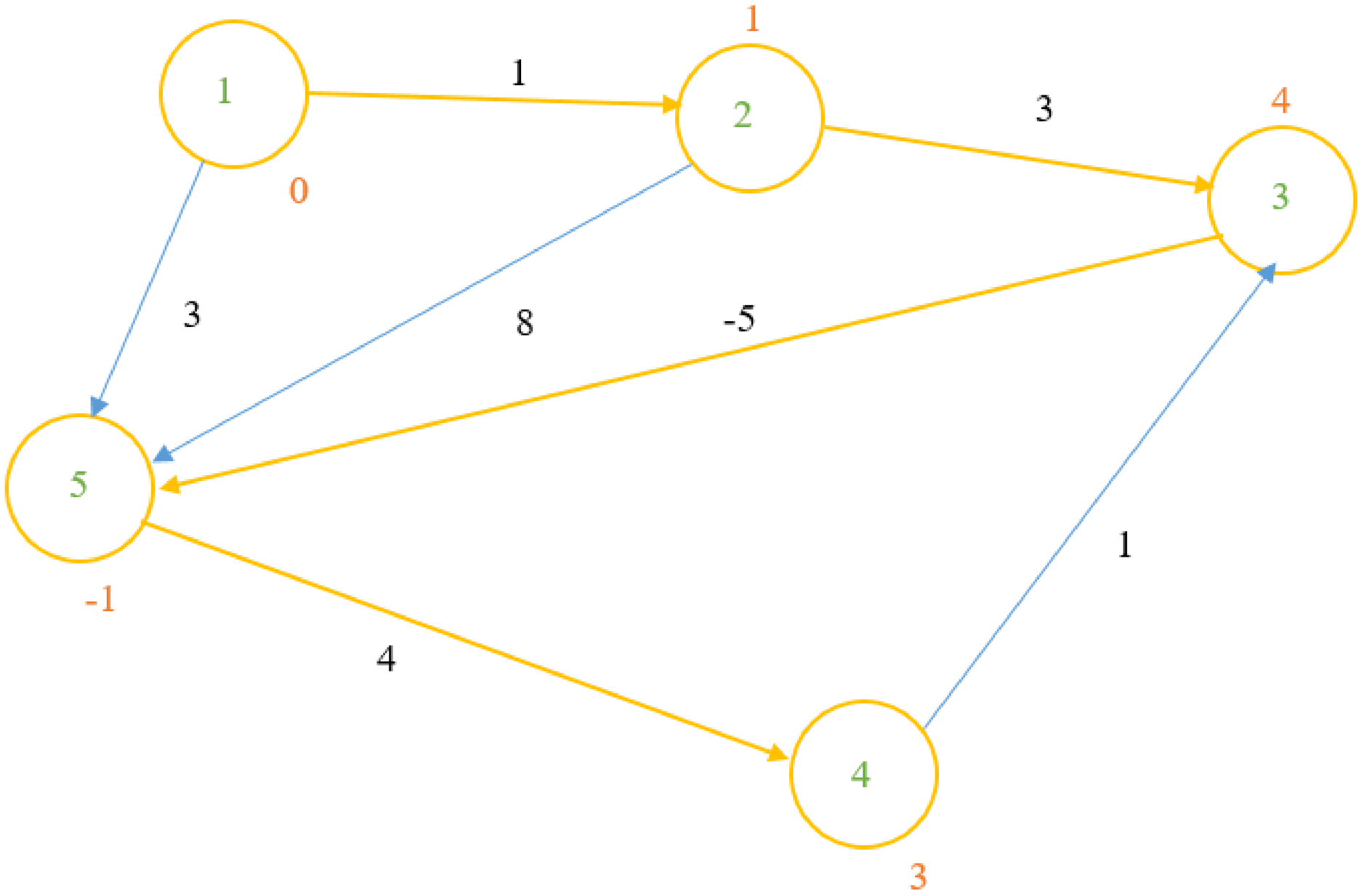


1	2	3	4	5
0	1	4	3	-1

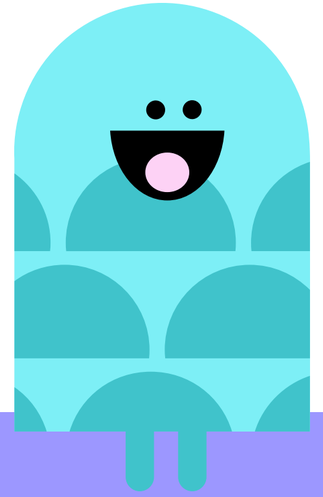
- Vòng lặp thứ nhất i = 2



Có thể thấy kết quả của vòng lặp thứ nhất với vòng lặp thứ hai -> có thể dừng vòng lặp lại và kết luận có đường đi ngắn nhất theo thuật toán Bellman Ford như hình:



TRƯỜNG HỢP XẤU NHẤT *

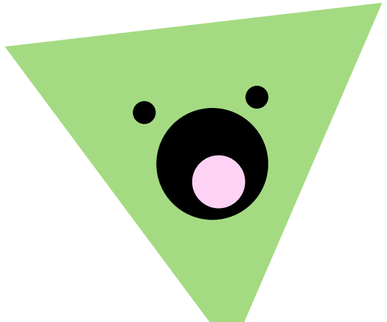


Disjoint Graph

Negative Cycle

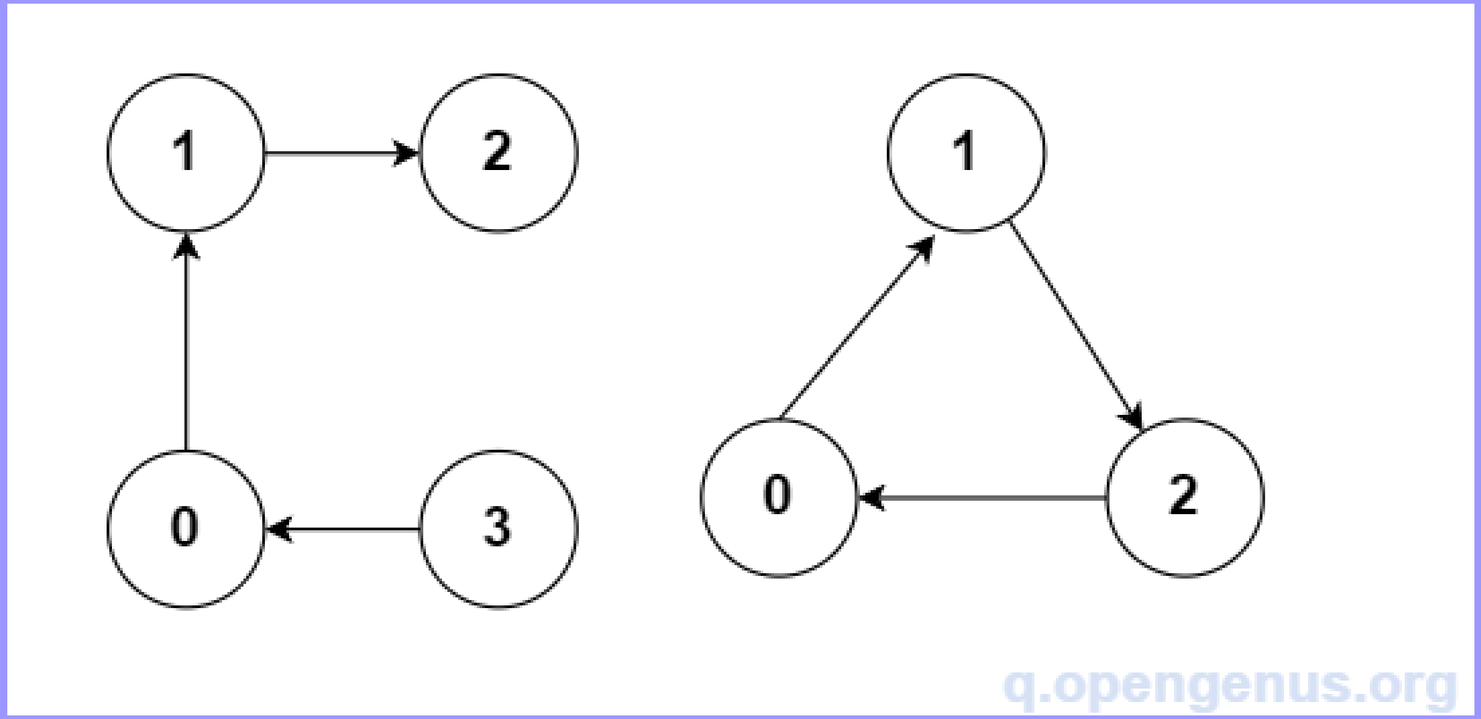
Order of edge list is reversed

Complete graph



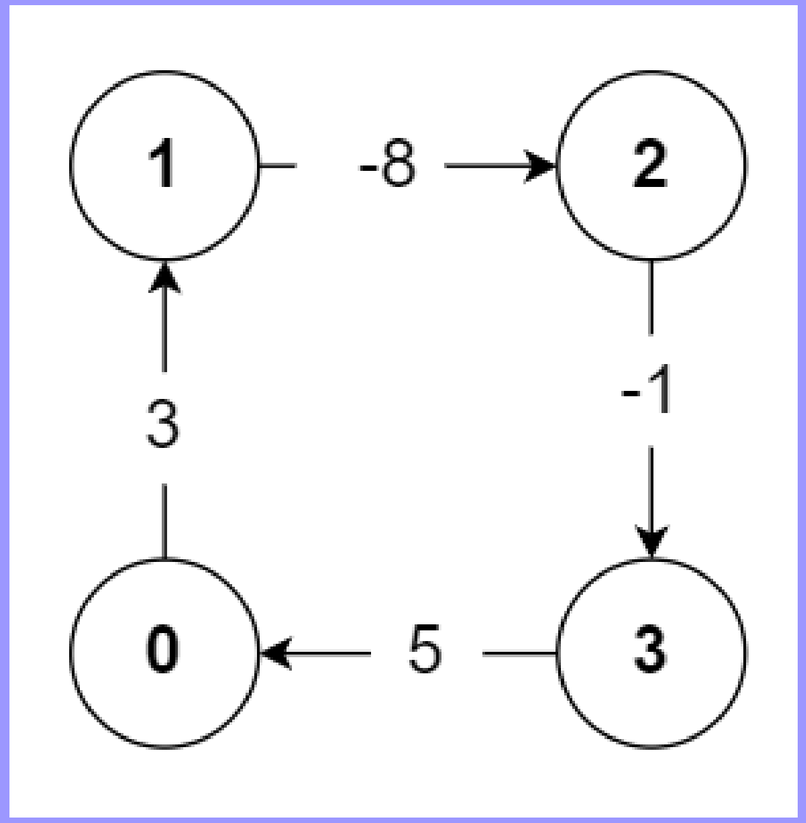
DISJOINT GRAPH O(V*E)

*

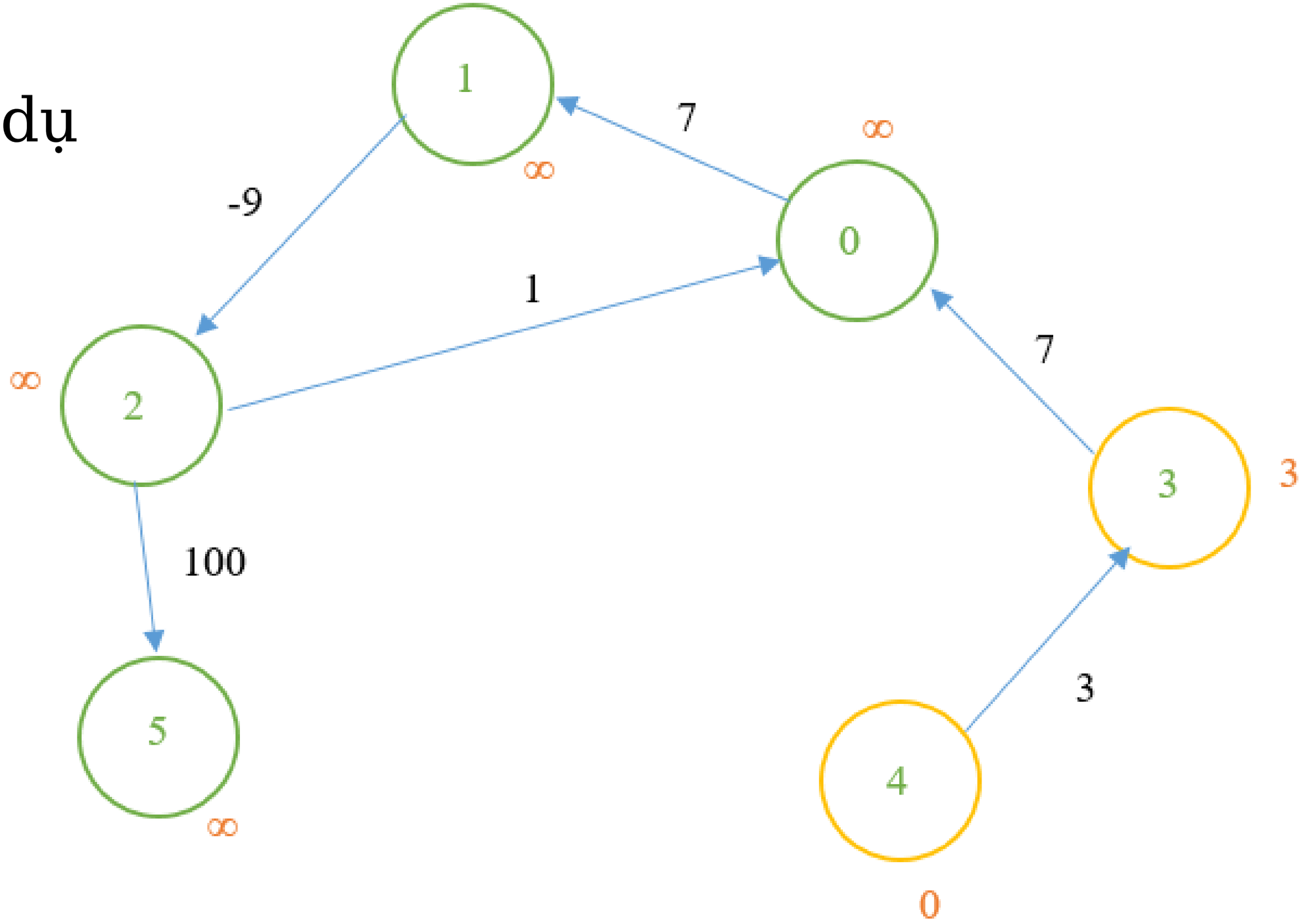


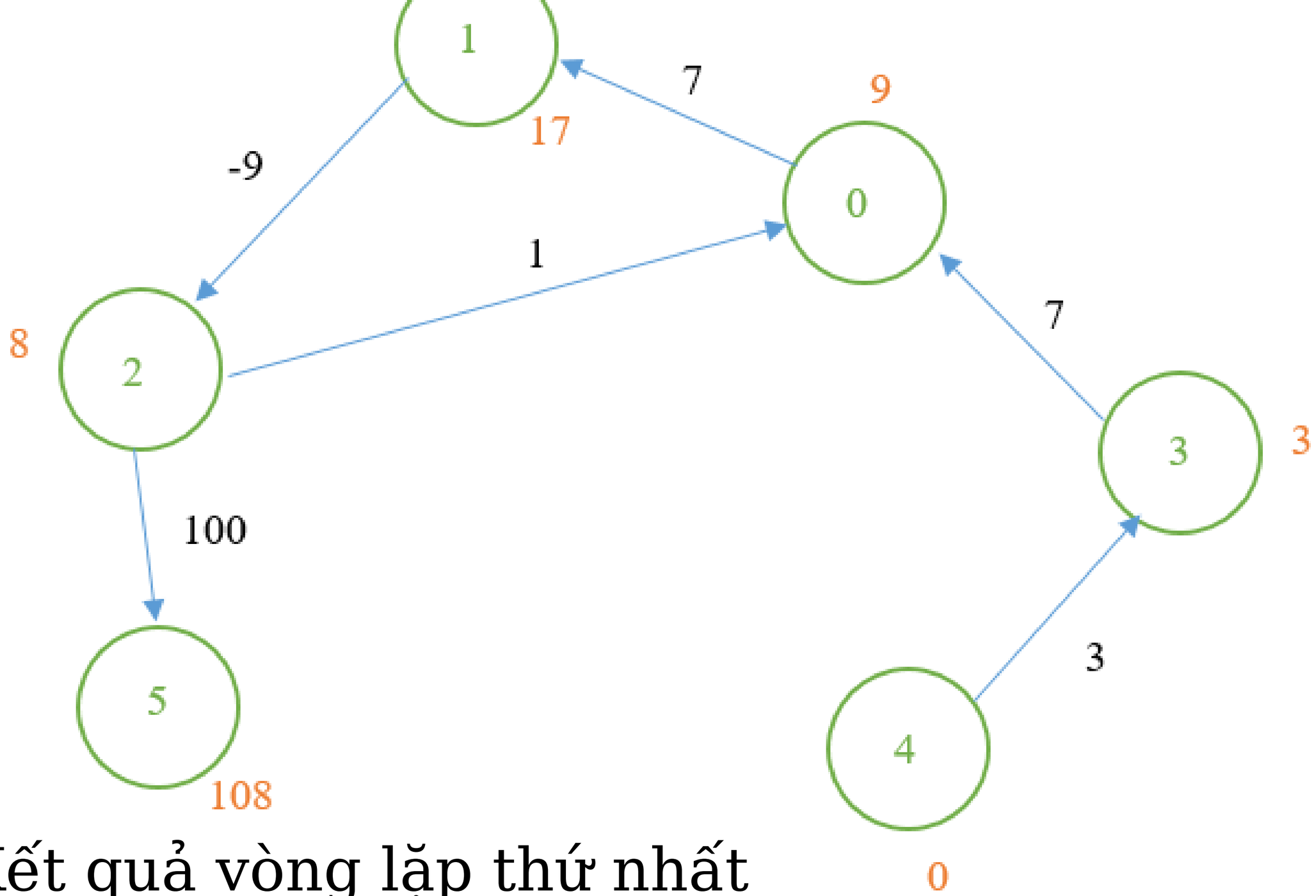
NEGATIVE CYCLE O(V*E)

*

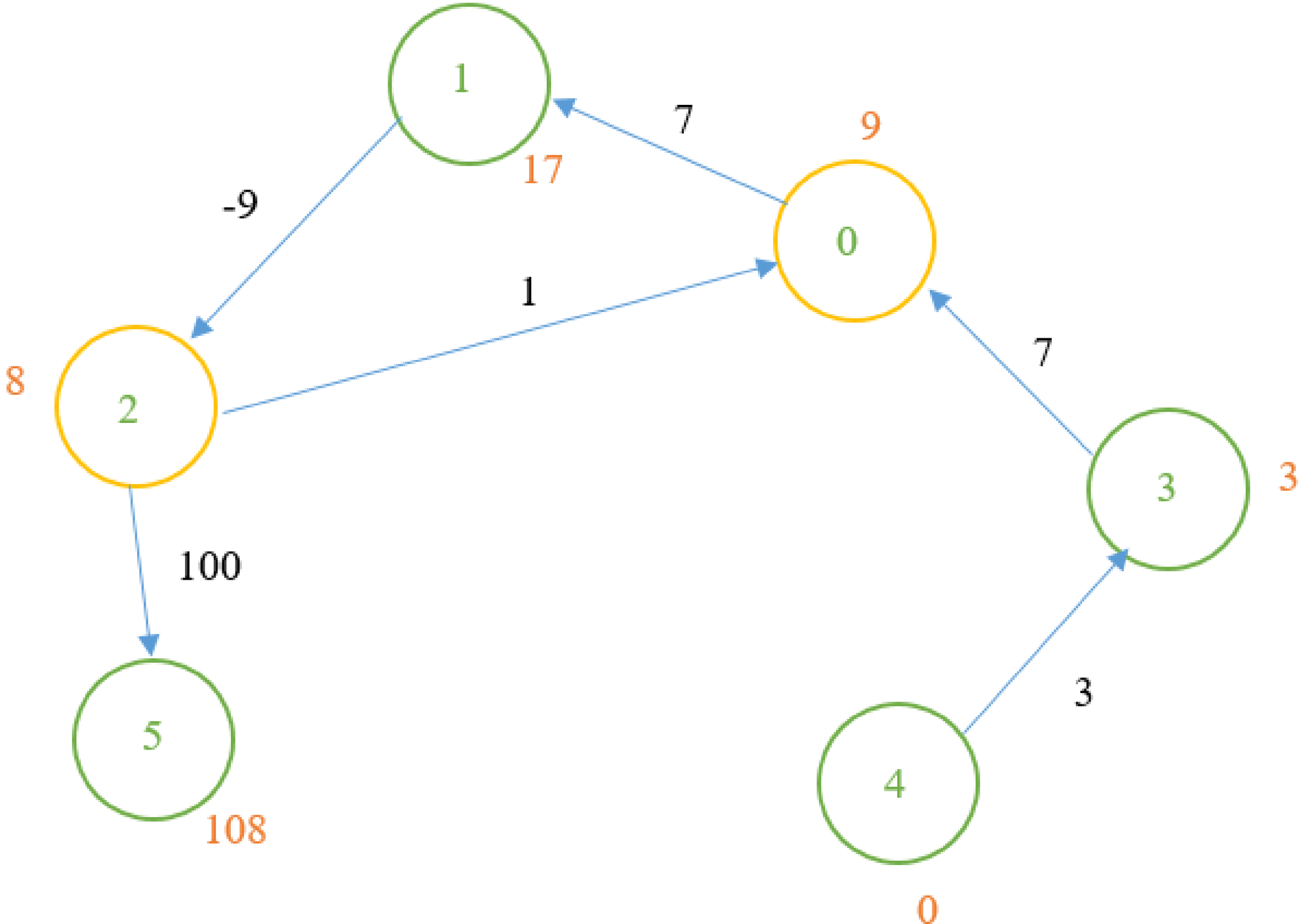


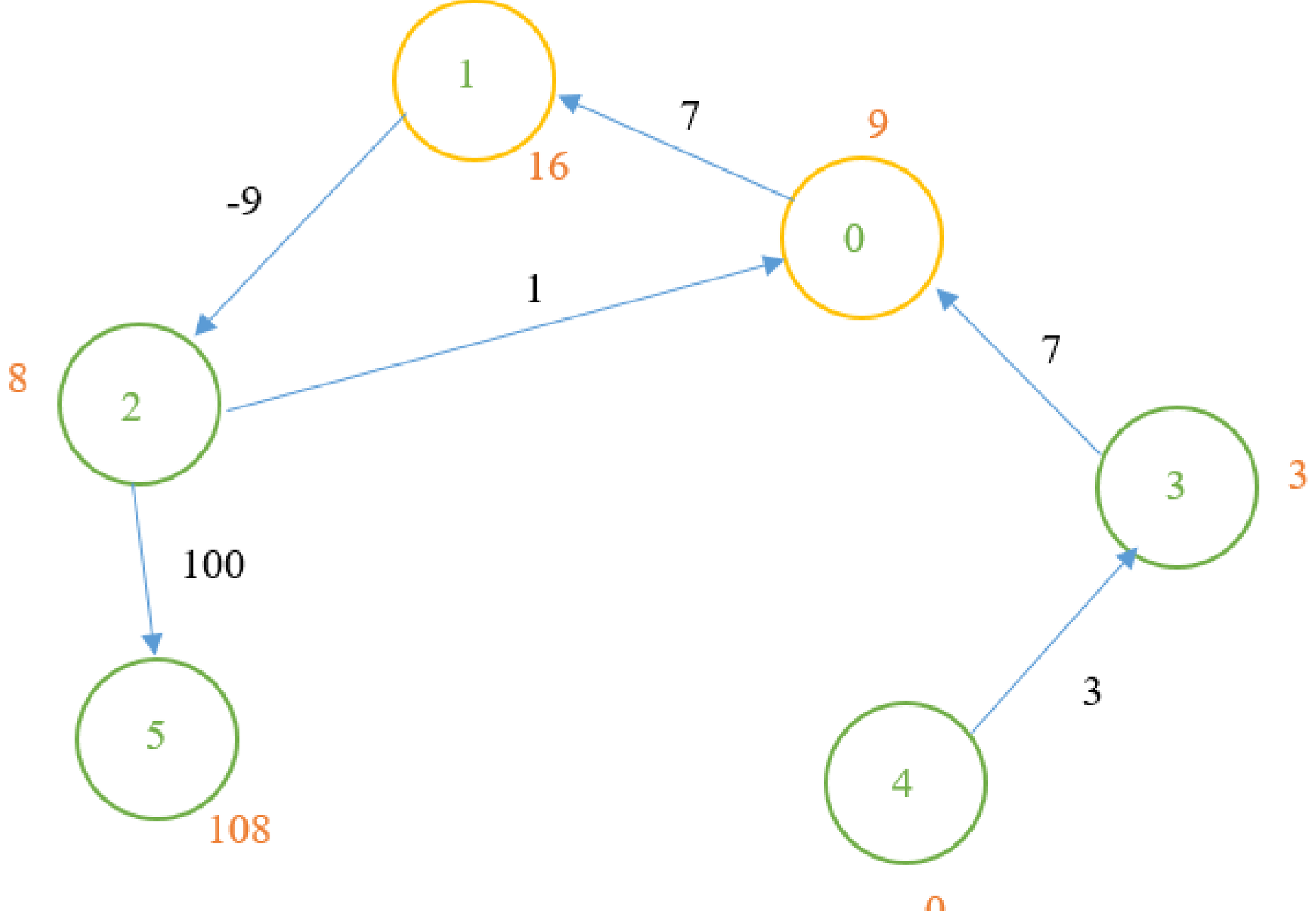
Ví dụ

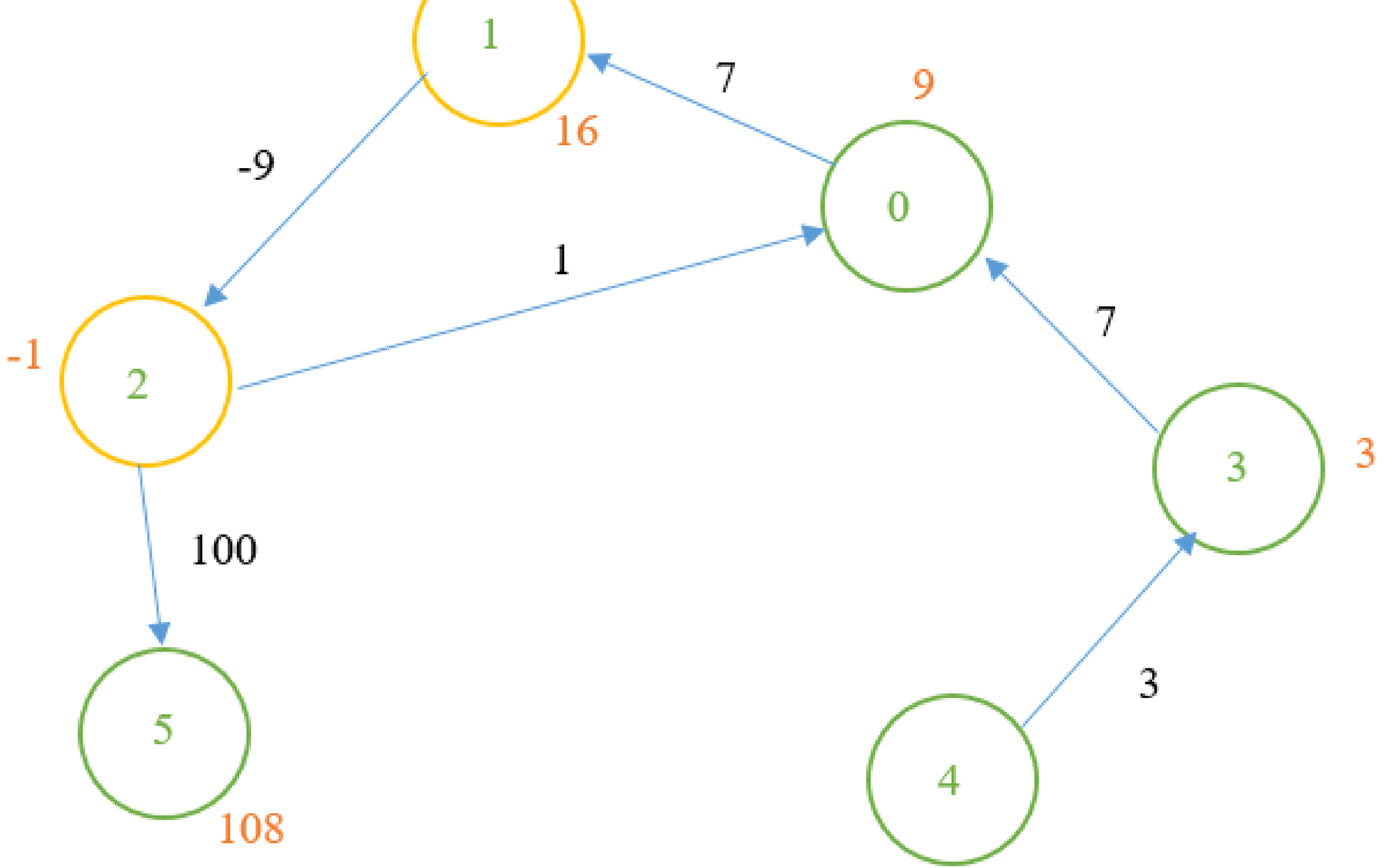




Kết quả vòng lặp thứ nhất

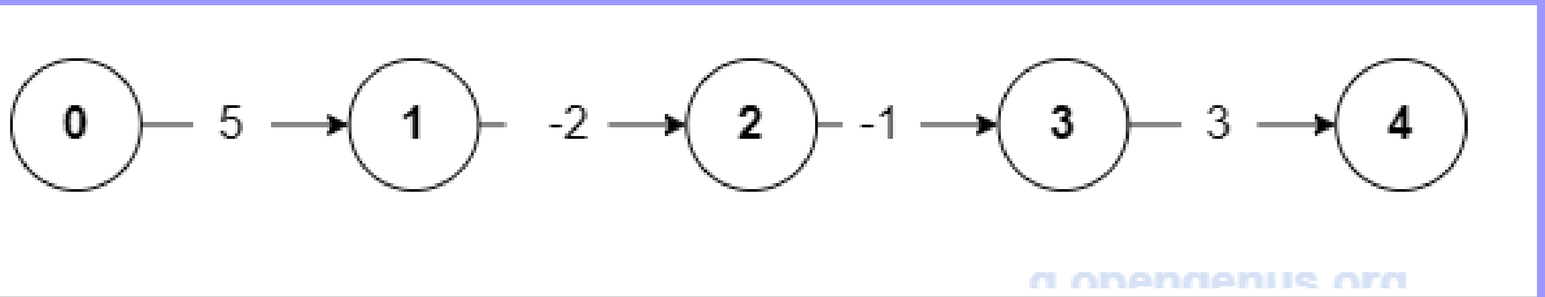

1
17
7
9
-9
0
1
8
2
7
3
3
100
5
108
4
3
0


1
16
7
9
-9
0
1
8
2
7
3
3
100
5
108
4
3
0


1
16
7
9
-9
0
1
-1
2
7
3
3
100
5
108
4
3

ORDER OF EDGE LIST IS REVERSED O(V*E)

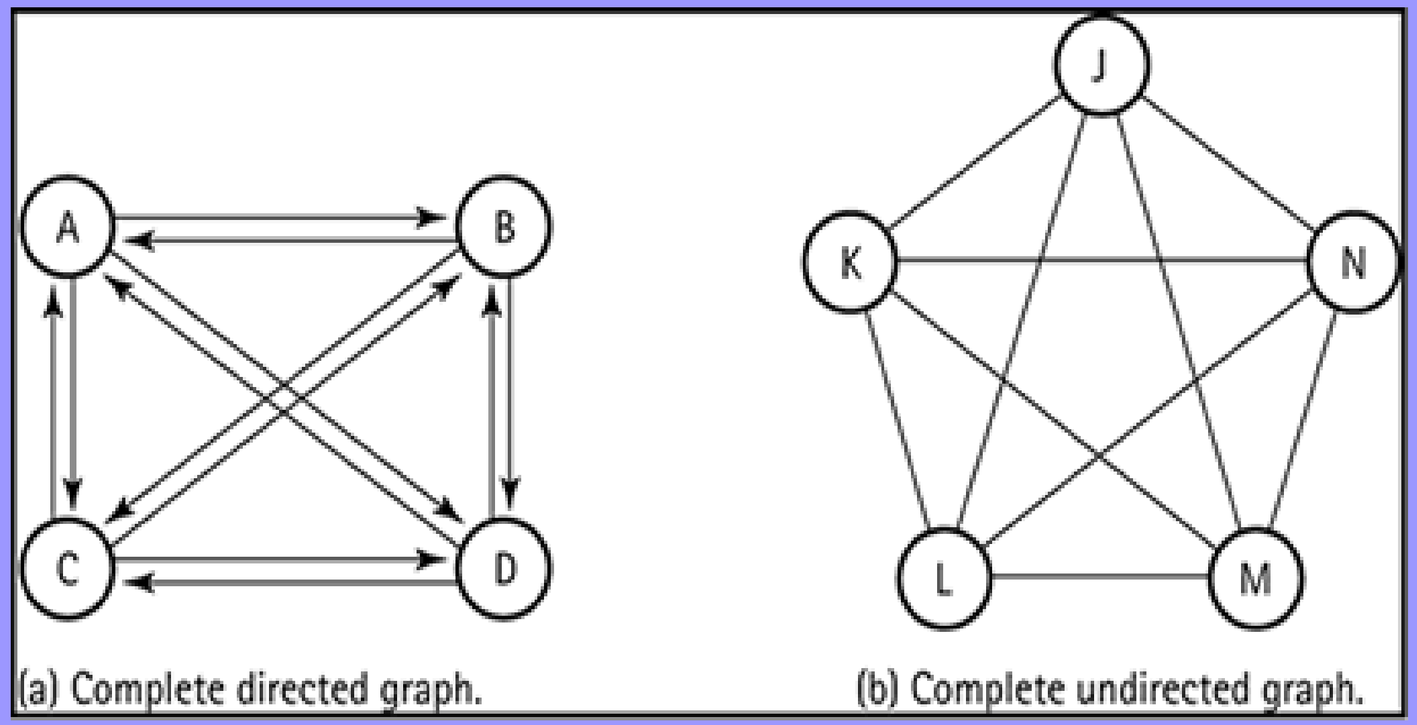
*



u	v	w
3	4	3
2	3	1
1	2	2
0	1	5

COMPLETE GRAPH O(V^3)

*

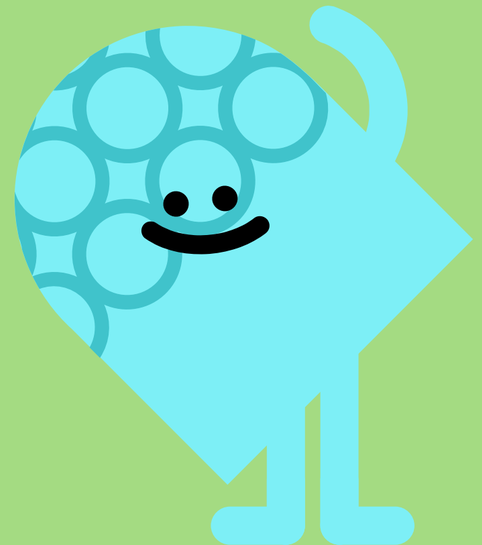


(a) Complete directed graph. (b) Complete undirected graph.

VÍ DỤ ÁP DỤNG *

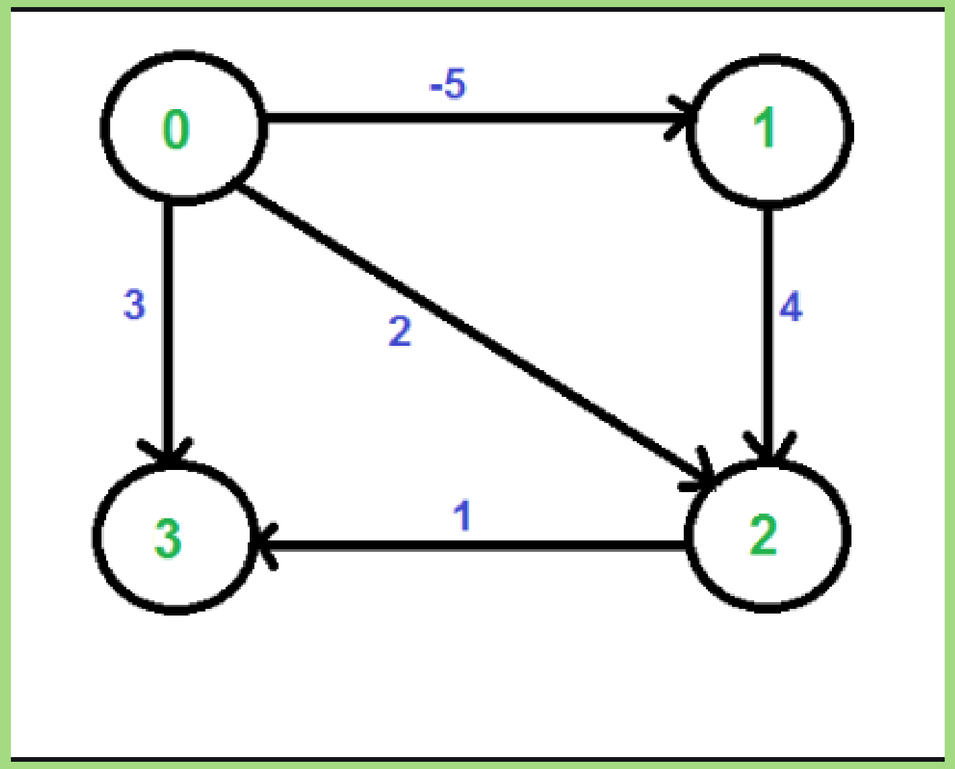
JOHNSON ALGORITHM

***ROUTING INFORMATION PROTOCOL*(*RIP*)**

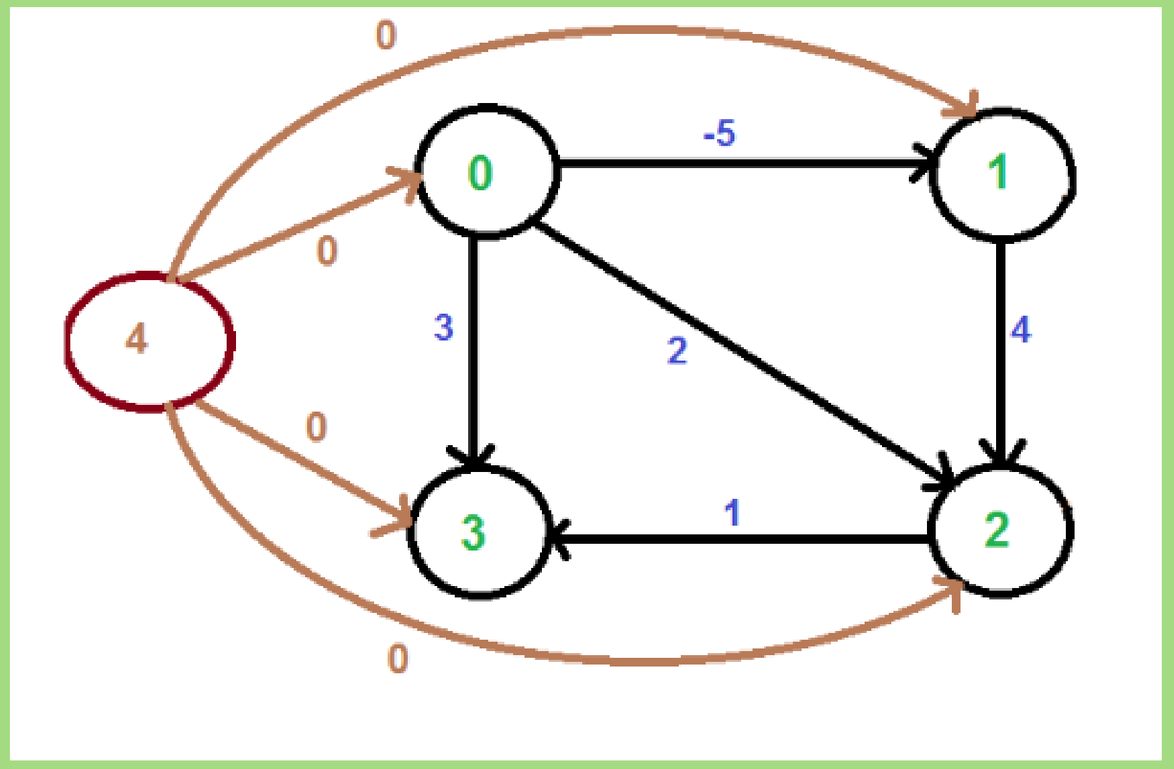


JOHNSON ALGORITHM

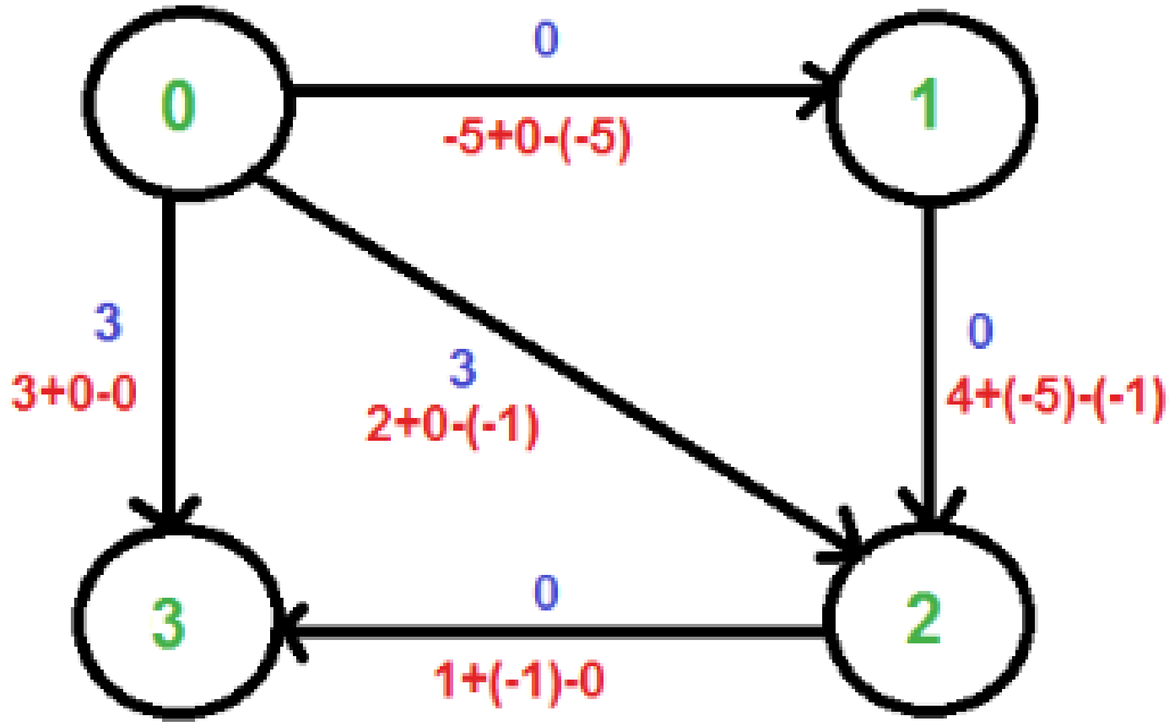
Đồ thị ban đầu



Tạo 1 đỉnh mới S



SỬ DỤNG BELLMAN



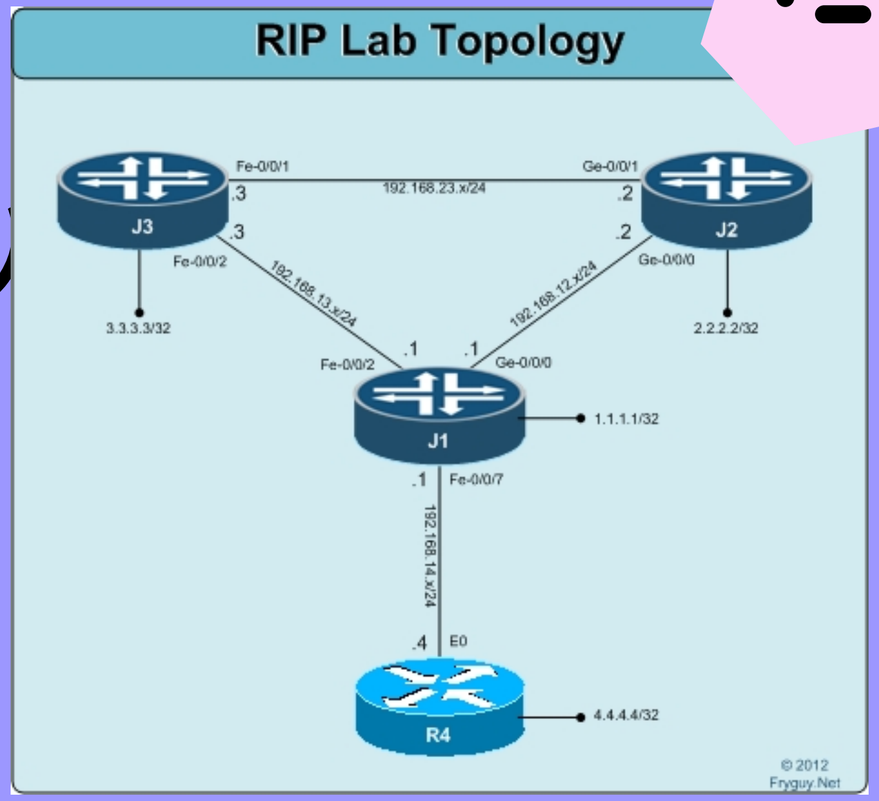
Distances from 4 to 0, 1, 2 and 3 are 0, -5, -1 and 0 respectievely.

RIP

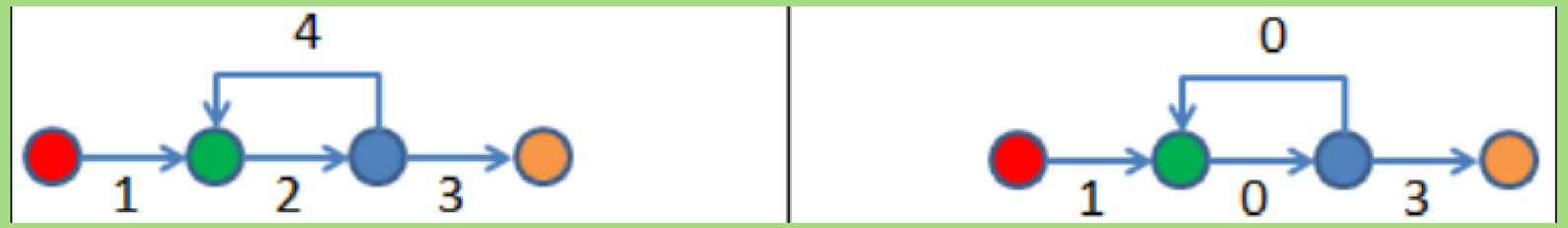
*

RIP LÀ GIAO THỨC ĐỊNH TUYẾN VECTOR KHOẢNG CÁCH ĐIỂN HÌNH

NÓ ĐỀU ĐẶN GỬI TOÀN BỘ ROUTING TABLE RA CÁC ROUTER HÀNG XÓM VÀ CÁC ROUTER NÀY SẼ PHÁT TÁN RA TẤT CẢ ROUTER BÊN CẠNH ĐỀU ĐẶN THEO CHU KỲ LÀ 30 GIÂY

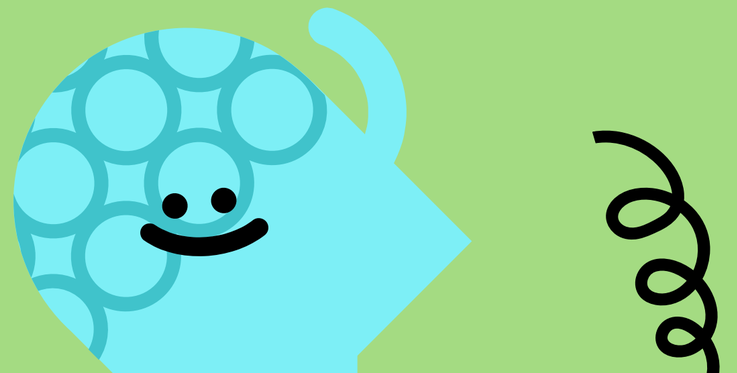


Tính đúng đắn của thuật toán



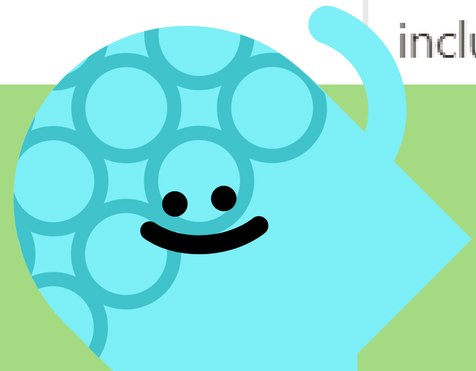
SO SÁNH CÁC THUẬT TOÁN LIÊN QUAN*

TYPE	DIJKTRA	BELLMAN FORD	SHORTEST PATH FASTER ALGORITHM	FLOYD WARSHALL	JOHNSON
Usage	Shortest path from one node to all nodes	Shortest path from 1 node to all nodes	Shortest path from 1 node to all nodes	Shortest path between all pairs	Shortest path between all pairs



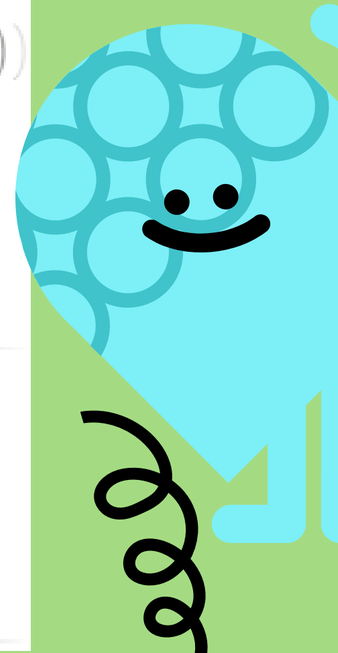
SO SÁNH CÁC THUẬT TOÁN LIÊN QUAN*

TYPE	DIJKTRA	BELLMAN FORD	SHORTEST PATH FASTER ALGORITHM	FLOYD WARSHALL	JOHNSON
Negative edge handling	No	Yes	Yes	Yes	Yes
Negative cycle detection	No	Yes	Yes	Yes	Yes (bc it has included BF)



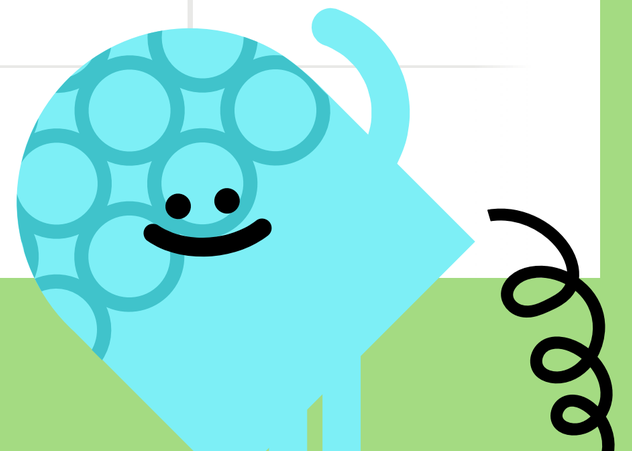
SO SÁNH CÁC THUẬT TOÁN LIÊN QUAN*

TYPE	DIJKTRA	BELLMAN FORD	SHORTEST PATH FASTER ALGORITHM	FLOYD WARSHALL	JOHNSON
Time complexity	$O(E + VlogV)$ w/ Fibonacci Heap	• Best case: O(E) • Medium case: O(VE) • Worst case: O(V^3)	• Best case: O(E) • Medium case: O(E) • Worst case: O(VE)	O(V^3)	$O(V^2 + logV + VE))$
Space complexity (without graph representation)	O(V)	O(V), (size of the array of distances in N); O(V)	O(E)	O(V^2)	O(E + 2VE)



SO SÁNH CÁC THUẬT TOÁN LIÊN QUAN*

TYPE	DIJKTRA	BELLMAN FORD	SHORTEST PATH FASTER ALGORITHM	FLOYD WARSHALL	JOHNSON
When to use	No negative edges, dense graph	Negative edges in graph	Negative edges in graph, dense graph	Dense graphs aka many edges	Sparse graphs aka few edges
Application in networking	Open Shortest Path First (OSPF)	Routing Information Protocol (RIP)			



CẢM ƠN!

Các bạn có bất kỳ câu hỏi nào cho chúng tôi không?