Bellman-Ford / Floyd-Warshall

Wednesday, 1 June, 2022 15:08

best

dijkstra	bellman-ford	floyd-warshall
× neg edges	✓ neg edges	✓ neg edges
$O(E \log V)$ time	$O(EV)$ time	$O(V^3)$ time
$O(V)$ aux space	$O(V^2)$ aux space ↓ optimize $O(V)$ aux space	$O(V^2)$ aux space
greedy	dynamic prog	dynamic prog
		transitivity

if no neg cycle

diagonal is 0 or not neg

Shortest distance from A to B using up to a maximum of i edges.

i+1 (edge)

Bellman-Ford Questions

Wednesday, 1 June, 2022 19:10

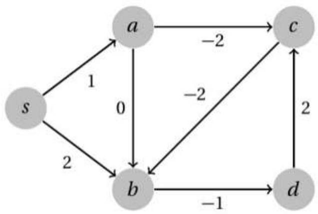
Question 20

Consider the following graph.

After some number of iterations of **Bellman-Ford**, the distance values to each vertex are as shown in the table below.

What would the distance values be **after one more iteration** of Bellman Ford? Please **choose** the appropriate value for each vertex from the **drop-down options**.

Assume that the **edges are being processed in the following order:** a->b, a->c, b->d, c->b, d->c, s->a, s->b



Vertex	Iteration		
	0	1	2
s	0	0	
a	∞	1	
b	∞	2	
c	∞	∞	
d	∞	∞	

What is the distance to **s** after iteration 2?

What is the distance to **a** after iteration 2?

What is the distance to **b** after iteration 2?

What is the distance to **c** after iteration 2?

What is the distance to **d** after iteration 2?

The correct answer is:

- What is the distance to **s** after iteration 2? → 0,
- What is the distance to **a** after iteration 2? → 1,
- What is the distance to **b** after iteration 2? → -3,
- What is the distance to **c** after iteration 2? → -1,
- What is the distance to **d** after iteration 2? → 0

Graph and Shortest Distance

Question 17

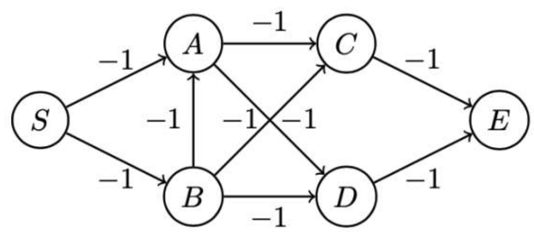
1 Mark

Consider the following version of the **Bellman-Ford algorithm**:

Algorithm 59 Bellman-Ford

```
function BELLMAN_FORD(G = (V, E), s)
    dist[1..n] = ∞
    pred[1..n] = null
    dist[s] = 0
    for k = 1 to n − 1 do
        for each edge e in E do
            RELAX(e)
    return dist[1..n], pred[1..n]
```

and the following **directed graph**



Let S be the source node for the execution of the Bellman-Ford algorithm.

If the edges are relaxed in the following order (S,B), (A,C), (B,C), (S,A), (B,A), (C,E), (D,E), (A,D), (B,D).

What is the value of dist[C] **after the first iteration of the outer loop** is done?

Select one:

- a. dist[C]=∞
- b. dist[C]=-3
- c. dist[C]=-2

Question 18

1 Mark

What is the value of dist[D] **after the first iteration of outer loop** of Bellman-Ford in the scenario above?

Select one:

- a. dist[D]=∞
- b. dist[D]=-3
- c. dist[D]=-2

Question 19

1 Mark

What is the value of dist[E] **after the first iteration of outer loop** of Bellman-Ford in the scenario above?

Select one:

- a. dist[E]=∞
- b. dist[E]=-4
- c. dist[E]=-3

Floyd-Warshall

Wednesday, 1 June, 2022 20:12

Question 20

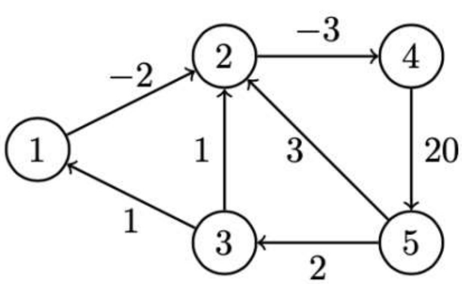
Consider the **Floyd-Warshall algorithm**

2
Marks

Algorithm 63 Floyd-Warshall

function FLOYD_WARSHALL(G = (V, E))
  $dist[1..n][1..n] = \infty$
  $dist[v][v] = 0$ **for all** vertices v
  $dist[u][v] = w(u, v)$ **for all** edges $e = (u, v)$ **in** E
  for each vertex $k = 1$ **to** n **do**
    for each vertex $u = 1$ **to** n **do**
      for each vertex $v = 1$ **to** n **do**
        $dist[u][v] = \min(dist[u][v], dist[u][k] + dist[k][v])$
  return $dist[1..n][1..n]$

and the following **directed graph**



After the outer loop of the algorithm **finished two iterations**, what is the sum of all values in the array *dist* that are **not equal to infinity**? Just type the numerical answer without punctuation or spaces.

K	1	2	3	4	5
U	1	2	3	4	5
V	1	2	3	4	5

dis[u][v], dis[u][k] + dist[k][v]

	1	2	3	4	5
1	0	-2	inf	inf	inf
2	inf	0	inf	-3	inf
3	1	1	0	inf	inf
4	inf	inf	inf	0	20
5	inf	3	2	inf	0

→

	1	2	3	4	5
1	0	-2	inf	inf	inf
2	inf	0	inf	-3	inf
3	1	-1	0	inf	inf
4	inf	inf	inf	0	20
5	inf	3	2	inf	0

→

	1	2	3	4	5
1	0	-2	inf	-5	inf
2	inf	0	inf	-3	inf
3	1	-1	0	-4	inf
4	inf	inf	inf	0	20
5	inf	3	2	0	0

Only calculate the between of vertex 1 and 2 since it says 2 iterations
Total sum = 0-2-5+0-3+1-1-4+0+20+3+2+0+0 = 11

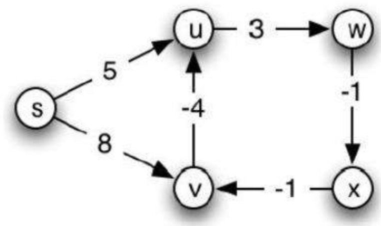
Printout

Wednesday, 1 June, 2022 21:02

Bellman-Ford

Fundamentals
- Consider all edges and find shortest distance
- In between Brute Force and Dijkstra (Can be better than Brute Force thanks to DP)
- 2 Components: Distance calculation & checking for negative cycle.
- No shortest distance if negative cycle because it'll loop forever. *(Think about this, being the algo itself, you find yourself looping the same cycle thinking that every loop would result you in having a short distance)*
- Loop |V| - 1 Times → Max number of jumps without cycle in graph / Between v and u, can have |V| - 1 edges without cycle.

Example



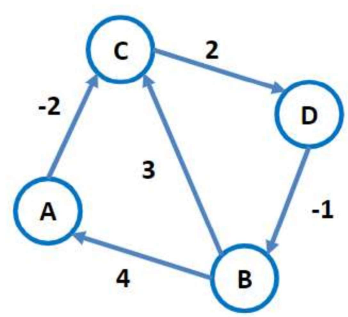
1. Start with all inf except for target source, which will be set to 0
2. Every increment of i represents the number of jumps the algo can take
3. Within that i, update all distances (like Dijkstra), refer to previous iteration for vertex distance
4. Compute until reach |V| - 1, while V iteration is used to check for negative cycle. If any of the weights are smaller, then there exists a negative cycle, with the decrease factor be the distance reduction whenever the cycle is looped.

Complexity
- Initialize → O(V)
- Calculate Distance → O(V) outer loop, O(E) inner loop (for every vertex, check all possible edges)
- Check negative cycle → O(E) one last time (check through all edges and update them accordingly to check)
- **Total → O(VE)**

Floyd-Warshall

Fundamentals
- Find **All-Pairs** Shortest Distance
- Transitive Closure: If A → B and B → C, then A → C
- If represented in a matrix, means that: If matrix[i,j] = True and matrix[j,k] = True, then matrix[i,k] = True
- Warhsall's Algorithm → Time of O(V^3) and Space of O(V^2) for matrix
- This means that, Distance (A → B) + Distance (B → C) = Distance (A → C), meaning that we can find shortest distance between A and C using B
- Why not any other algo? Dijkstra and Bellman-Ford needed to be ran multiple times.

	A	B	C	D
A				
B				
C				
D				



1. Initialize matrix with pre-written weights and infinity. Each slot that refers to same vertex as pair (e.g. A → A) all have weight 0 unless directing to itself.
2. In each vertex, update distance of all vertices going through the said vertex.
3. How to find negative cycle? Find negative values in the matrix.

Why are these important?

- **Bellman-Ford**
- **Floyd-Warshall**

What's Next?

- **Network Flow**