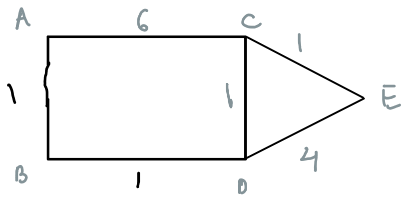
Single Source shortest path : Dijktra's Algorithm and Bellman-ford

03 April 2025 23:46



given $G(V,E)$ directed graph $W: E \to R$

$$W(p) = \sum_{i=1}^{k} W(V_{i-1}, V_i)$$

$A \to E$ $A \to B \to D \to C \to E$.

$\delta(u,v)$ shortest path :→ $\delta(u,v) = \begin{cases} \min(W(p) ; u \overset{p}{\leadsto} v) & ; \text{there is a path} \\ \infty & ; \text{otherwise} \end{cases}$

Optimal substructure → shortest path contains another shortest paths within it.

If has negative weight cycles → Not well defined path.
every time gets relaxed → $-\infty$.

Negative edge still works well.

Dijkstra's → All +ve edge

Bellman Ford → can be -ve edge but no negative cycles.

Representing shortest path.

predecessor → $v.\pi$.
$E(v.\pi, v)$

INITIALIZE single source (G, S)

for each vertex $V \in G.V$
$V.d = \infty$
$V.\pi = NIL$

$s.d = 0$.

Print_Path(G, s, v)

1.) If $s == v$
print S
2)
3.) else if $v.\pi = NULL$
4.) print "No path"
5.) else
6.) PRINT_PATH$(G, s, v.\pi)$
7.) print v.

RELAX(u, v, w)

1.) if $v.d > u.d + w(u,v)$
$v.d = u.d + w(u,v)$

→ if weight till now > some $u \to v$
then add $u \to v$

1.) if v.d ...

$v.d = u.d + w(u,v)$

2.) $v.\pi = u$

3.)

then add u→v in the path.

BELLMAN – FORD ALGORITHM.

* can have –ve weight
* can detect if negative cycles present.

Dijkstra > BF (running time)

Assume :→ $a \in R$ then $a + \infty = \infty + a = \infty$

$a + (-\infty) = -\infty + a = -\infty$

Returns boolean → whether or not a negative cycle from source

↳ if NO → path.

BELLMAN – FORD (G, W, S)

1.) Initialise single source (G, S)

2.) for i = 1 to |G.V| − 1

3.) for each edge e ∈ G.E and e(u,v)

4.) Relax (u, v, w)

5.) for all edges (u, v) ∈ G.E

6.) if v.d > u.d + w(u,v)

return FALSE

7.)

8.) return TRUE.

DIJKSTRA'S ALGO → choose lightest edge or closest edge.

DIJKSTRA'S (G, W, S)

1.) Initialize single source (G, S)

2.) S = φ → visited

1.) [illegible]

2.) $S = \varphi$ → visited

3.) $Q = G.V$ → priority queue of all vertices.

4.) while $Q \neq \varphi$

5.) $u =$ EXTRACT-MIN(Q); → extract min of the queue

6.) $S = S \cup (u)$ → add to visited.

7.) for each vertex $v \in G.V$

8.) relax (u, v, w);

T.C.

i.) depends on priority queue

ii) Relax(u,v,w) involves decrease Key $(Q, u, u[d] + w(u,v))$

If array implementation → Insert + update → $O(1)$

But Minimum extract → $O(V)$

(for each V → extract will cost $O(V^2)$) + every edge Relax $= |E|$

∴ Total complexity of array $= O(V^2 + E) = O(V^2)$

If priority queue

EXTRACT-MIN(Q) → $O(\log V)$ → (V such operations.

→ $O(|V| \log V)$

Building queue $= O(|V|)$

Each Decrease key → $O(\log V)$ → $|E|$ such operation → $O(E \log V)$

∴ overall Running time $= O((V+E) \log V + V) = O(E \lg V)$

Using Fibonacci Heap

extract min → $O(V \lg V)$

decrease Key → $O(1) \to O(E)$

Total complexity = $O(V \log V + E)$