ALGORITHMS - ASSIGNMENT 8

KAVYA SHAMSUNDARA

(1) A looped tree is a weighted, directed graph built from an n-node binary tree by adding an edge from every leaf back to the root of the tree. All edges have non-negative weight

(a) How long would it take Dijkstra's algorithm to compute the shortest path b/w 2 vertices in a looped tree?

(b) Describe & analyze a faster algorithm.

Solution:

(a) * If we implement Dijkstra's algorithm with a binary heap
- Extract min takes $O(\log V)$ time
- Each decrease-key also takes $O(\log V)$
∴ We get $O(E \log V)$.

* If we use Fibonacci heap, the amortized cost of the extract-min operations is $\log V$, but the decrease key is $O(1)$.
∴ We get $O(E + V \log V)$

(b) * "Bellman-Ford" (some early sources refer as Shimbel) algorithm. This algorithm is faster & and efficient if there are negative edges.

* Analysis

In each phase, we scan ~~the~~ each vertex at most once, so we relax each edge at most once, so the running time of a single phase is $O(E)$.

The overall running time is $O(VE)$.

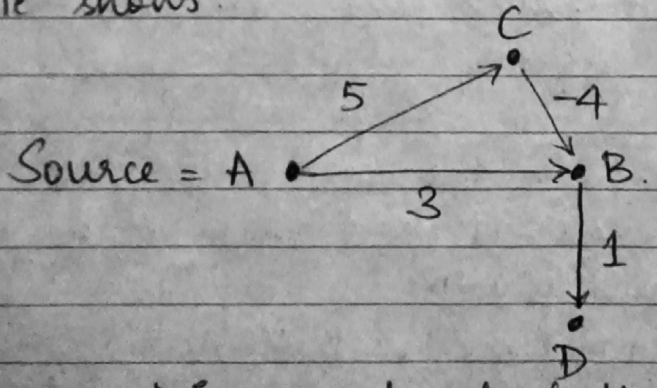
* If there are no negative edges, however, Dijkstra's algorithm is faster. (In practice, in fact, Dijkstra's algorithm is often faster even for graphs with negative edge).

(2). Give a modified version of Dijkstra's alg that works for (a) negative edge weights, as long as there are no negative cycles.

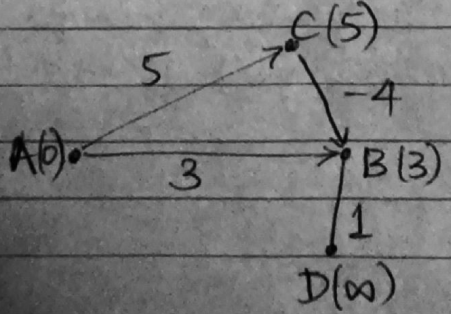
Solution:

```
Dijkstra (G, s)
for each v ∈ V
    d[v] ← ∞
d[s] ← 0
S ← ϕ
Q ← V
while Q ≠ ϕ
        u ← ExtractMin(Q)
        S ← S ∪ {u}
        for each v ∈ Adj [u]
            if d[v] > d[u] + w(u,v)
                DecreaseKey(v, d[u] + w(u,v))
```

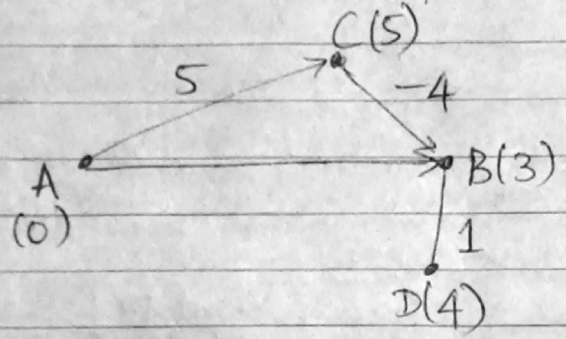
* Dijkstra's algorithm does not work if the graph has negative weight edges ie, it might return incorrect result, as following example shows.



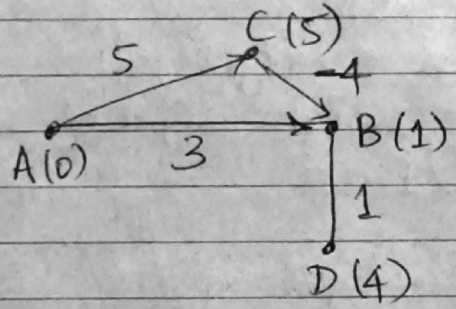
- First source A is popped out of the queue & edges (A, C) & (A, B) are relaxed.



- Now B is on top of min-heap queue Q.
We now pop B & relax the only edge out of B ie., (B,D) giving D distance 3+1=4

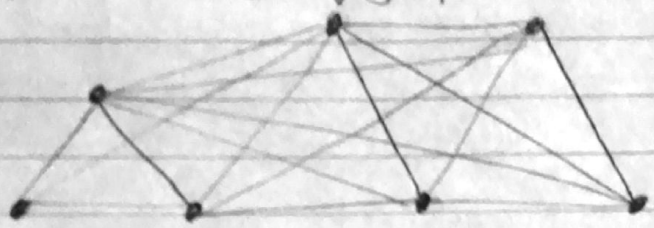


- Now D is on top of min-heap & we remove D.
There are no edges going out of D to relax.
The only vertex now left in the queue is C.
We remove C and relax the edge (C,B) Since 5-4=1<3, Distance to B must be updated to 1.



- The algorithm now terminates because the queue is empty. However, the path A → C → B → D has total length 5+(-4)+1 = 2 < 4; thus the length of the shortest path from source A to D was not correctly evaluated.

(2)(b) Infinite family of graph



// Step 1 : Initialize graph
for each vertex v in vertices:
    if v is source then weight[v] := 0
    else weight[v] := infinity
    predecessor[v] := null

// Step 2: relax edges repeatedly
for i from 1 to size(vertices) - 1:
    for each edge (u,v) with weight w in edges:
        if weight[u] + w < weight[v]:
            weight[v] := weight[u] + w
            predecessor[v] := u

// Step 3: check for negative weight cycles
for each edge (u,v) with weight w in edges:
    if weight[u] + w < weight[v]:
        error "Graph contains a negative-weight cycle"
return weight[], predecessor[]

③ Floyd-Warshall-Algorithm for negative cycle

Input: A digraph G with $V(G) = \{1, \ldots, n\}$ & weights $c : E(G) \to R$

Output: An $n \times n$ matrix M such that $M[i,j]$ contains the length of a shortest path from vertex i to vertex j.

$M[i,j] := \infty \; \forall \; i \neq j$

$M[i,i] := 0 \; \forall i$

$M[i,j] := c((i,j)) \; \forall \; (i,j) \in E(G)$

```
for i := 1 to n do
    for j := 1 to n do
        for k := 1 to n do
            if M[j,k] > M[j,i] + M[i,k]
            then M[j,k] := M[j,i] + M[i,k]
for i := 1 to n do
    if M[i,i] < 0 then return ('graph contains a
                               negative cycle').
```