Initialize-Single-Source (G, s)

1. for each vertex $v \in G.V$
2. $v.d = \infty$
3. $v.\pi = NIL$
4. $s.d = 0$ // root's distance $= 0$.

Relax (u, v, w)

① if $v.d > u.d + w(u, v)$.
② $v.d = u.d + w(u, v)$.
③ $v.\pi = u$.

Before relaxation, $d(v) \le d(u) + w(u, v)$.

After relaxation, $d(v)$ doesn't change.

Dijkstra Algo:

Dijkstra(G, w, s)

Initialize-Single-Source (G, s)

$S \leftarrow \phi$

$Q \leftarrow V[G]$. — Build-heap

while $Q \neq \phi$ do

{ $u \leftarrow$ Extract-min(Q)

$S \leftarrow S \cup \{u\}$.

for each vertex $v \in Adj[u]$ do

Relax(u, v, w)

}.

Dijkstra ≅ Prim.

Lightest edge in growing tree.

Dijkstra-analysis T.T.

① Linked list Representation

Build Queue : $O(V)$; with sorting $O(V \lg V)$

while loop → V times.

each extract-min : $O(V)$.

each edge relaxed E/V times.

∴ Total relaxation : $E/V \cdot V = E$ times.

Total extract-min : $V.V = V^2$.

∴ Total time $O(V^2 + E) = O(V^2)$.

② Binary heap representation:

Build heap $- O(V)$.

loop → V times.

extract-min $(\lg V)$.

Relaxation → $\lg V$.

Done " → $E \lg V$.

∴ Complexity $= V\lg V + E \lg V$.

∴ $O(E \lg V)$.

Fibonacchi heap: $O(V \lg V + E)$.

Relaxation-done in $O(1)$ times.

Correctness of Dijkstra : Using this loop invariant: At the start of each iteration of loop, $d[v] = \delta(s,u)$ for each vertex $v \in S$.

→ Shortest tree path at S of $G(V,E)$ is a directed subgraph $G'(V',E')$ where $V' \subseteq V$, $E' \subseteq E$ such that.

① V' is the reachable vertices set of S in G, G' is rooted tree with root S.

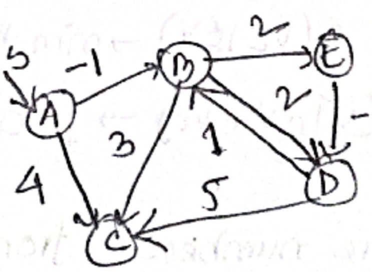
② for all $v \in V'$, unique simple path from s to v in G' is a shortest path from s to v in G.

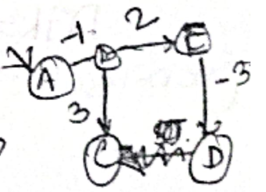
∴ Relaxation method as implemented in Dijkstra algo, is guaranteed to result in shortest-paths tree.

Bellman Ford: For negative-edges.

```
Bellman Ford (G,w,s)
Initialize-Single-Sources (G,s) → O(V)
for i=1 to |G.V|-1                  → O(V).
    for each edge (u,v) ∈ G.E   ] → O(E)
        Relax (u,v,w)
for each edge (u,v) ∈ G.E
    if v.d > u.d + w.(u.v)
        return FALSE
return True.
```

∴ Running time complexity → $O(V \cdot E)$.





A→B : -1.

A→C : 2.

A→E : 1.

A→D : -2.

MST-Prim (G, w, r).

```
for each v ∈ G.V do
        v.π ← NIL
        v.key ← ∞.
s.key ← 0
Q ← G.V                                  → V times
while Q ≠ 0 do                           → O(V) times
        u ← extractmin(Q)                → lg V
        for each v ∈ adj[u] do             E/V times
            if v ∈ Q   and   w(u,v) < v.key
              then   v.π ← u.
                     v.key ← w(u,v).       O(lg V)
```

complexity : $O(E \lg V)$.

$= O(V+E)\lg V$

$E > V$. ∵ $E \lg V$.

complexity by: adjacency list & binary heap.

Array + List:

- Extract min : $O(V)$

  Decrease key : $O(1)$.

∴ Total complexity. $O(V.V) = O(V^2)$.

List + Fibo :

- $O(V \lg V + E)$

  কারণ :

  extract min → $\lg n$.

  Decrease key → 1.

Kruskal MST (G,W)

Disjoint-SET DS.

complexity: $O(E \lg E)$ or $O(E \lg V)$

1. $A \leftarrow \emptyset$
2. for each $v \in G.V$ do
3. MAKE-SET(v) → $O(V)$
4. sort edges into non-decreasing order by weight. → $O(E \lg E)$
5. for each $u,v \in E$ non-decreasing edge do
   if FIND-SET (u) ≠ FIND-SET(v).
6. then $A \leftarrow A \cup \{(u,v)\}$.
7. UNION (u,v).
8.
10. Return A;

E times

Disjoint set operation $O(\lg V)$ or $O(\lg E)$

Ford – Fulkerson – Algo (G, c, s, t)

```
f(u,v) <- 0 for all edge (u,v)
G_f = [n x n]
While true. do
    m, P = Augmenting path (L, E, s, t, G_f)
    if m = 0
        break
    f += m
    for each edge (u,v) ∈ P
        F[u,v] = F[u,v] - m
        F[v,u] = F[v,u] + m
return G_f
```

↳ Replace DFS with it for Edmond Karp.

complexity : $O(E * \text{max flow})$

There can be $V*E$ path.

∴ While loop maxflow time চলবে।

→ Augment path বের করতে হবে $O(E)$ time-এ

Edmond Karp :

BFS → $V.E$ তে path বের করবে। total চলবে E times চলবে। $O(E^2 * V)$

Ford-Fulkerson Algo

Floyd Warshall (W) : complexity $O(n^3)$.

$n = W.rows$

$D^{(0)} = W$

for $k=1$ to n

Let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix

for $i=1$ to n

for $j=1$ to n

$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$.

return $D^{(n)}$.

Dinic $\rightarrow O(|E||V|^2)$.

$O(|V||E|^2)$.

Edmond Karp :

C, E, s, t, F = capa graph, edges, Sou tank, R graph

$f = 0$

$F = [n \times n]$ // residual capacity array.

m = bott capacity.

while true :

$m, P = BFS(C, E, s, t, F)$.

P = ~~Parent~~ graph Augmenting

if $m=0$

break

$f += m$

$v = t$

while $v != s$.

$u = P[v]$