Shortest Path

Donat un graf dirigit $G = (V, E)$ amb arestes amb pesos $w : E \to \mathbb{R}$, un camí $p = \{v_0, \dots, v_k\}$ és una seqüencia consecutiva d'arestes, on $(v_i, v_{i+1}) \in E$ defineix $w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$.

El ***shortest path*** entre u i v és

$\delta(u, v) = min_p\{w(p)|u \rightsquigarrow^p v\}$

Single Source Shortest Path (SSSP)

L'objectiu és computar el camí més curt des d'un vèrtex fins la resta de vèrtexs.

És a dir, donat $G = (V, E)$, $w : E \to \mathbb{R}$ i $s \in V$, computa $\delta(s, v), \forall v \in V - \{s\}$.

L'algoritme més bàsic per computar un SSSP és la **relaxació**.

Relaxació

Aquest algoritme ens donarà el SSSP del graf G.

```
Relax(u,v,w(u,v))
  if d[v] > d[u] + w(u,v) then
    d[v] = d[u] + w(u,v)
  end if

Relaxation(G, W, s)
  for all v in V - {S} do
    d[v] = infinity
  end for
  d[s] = 0
  while exists (u,v) with d[v] > d[u] + w(u,v) do
    Relax(u,v,w(u,v))
  end while
```

Dijkstra

És un algoritme **voraç** que **no funciona amb pesos negatius**. A més és l'algoritme **més ràpid** per calcular un SSSP.

Complexitat de Dijkstra $T(n) = O(mlg(n))$, si utilitzem *Heap* per la cua de prioritat.

```
Dijkstra(G, w, s)
  for all v in V do
    d[v] = infinite
    visited[v] = false
  end for
  d[s] = 0
  Q = {(s, d[s])} //priority queue, sorts by increasing d[x]
  while Q != {} do
    u = Q.pop()
```

```
    visited[u] = true
    for all v in Adj[u] do
      Relax(u,v,w(u,v))
      Q.append(v, d[v])
    end for
  end while
```

Bellman-Ford-Moore-Shimbel (BFMS)

Permet pesos **negatius** però no cicles negatius.

Per un graf $G = (V, E)$ amb n vèrtex l'algoritme fa n iteracions i a cada iteració i relaxa els vèrtex que estan a una distancia màxima i de s.

Complexitat de BFMS $T(n) = O(nm)$, on $n = |V|$ i $m = |E|$.

```
BFMS(G, w, s)
  for all v in V  and v not equal s do
    d[v] = infinite
  end for
  d[s] = 0
  for i = 1 to n - 1 do
    for all (u,v) in E do
      Relax(u,v,w(u,v))
    end for
  end for
  for all (u,v) in E do
    if d[v] > d[u] + w(u,v) then
      return negative-cycle
    end if
  end for
```

Shortest path in a DAG

Complexitat $T(n) = O(n + m)$

```
Shortest_dist_gag(G)
  Topological_sort(G)
  for all v in V
    d[v] = infinite
  end for
  d[s] = 0
  for all v in V-{s} do
    for all (u,v) in E do
      Relax(u,v,w(u,v))
    end for
  end for
```

All Pairs Shortest Paths (APSP)

Volem trobar el camí més curt entre tots els vèrtex del graf.

Donat $G = (V, E)$ i $w : E \to \mathbb{R}$ volem trobar $\forall u, v \in V, \delta(u, v)$.

Podem tenir **camins amb pes negatiu** però **no cicles negatius**.

Si apliquem **BFMS** n vegades el cost serà de $O(n^2m)$ i si fem el mateix amb ***Dijkstra*** el cost serà $O(nmlg(n))$.

A més de la distancia mínima entre tots els vèrtex també volem computar una **matriu de predecessors**.

Bernard-Floyd-Warshall (BFW)

Aquest algoritme utilitza **programació dinàmica** per comparar tots els camins entre dos vèrtex.

Té una **complexitat** de $O(n^3)$ i obviament el nombre màxim d'arestes és $O(n^2)$.

```
BFW(w)
  d = int[|V|][|V|] // initialised to ∞
  next = int[|V|][|V|]
  for all (u,v) in E do
    d[u][v] = w(u,v)
    next[u][v] = v
  end for
  for all v in V do
    d[v][v] = 0
  end for
  for k from 1 to |V| do
    for i from 1 to |V| do
      for j from 1 to |V| do
        if d[i,j] > d[i,k] + d[k,j] then
          d[i,j] = d[i,k] + d[k,j]
          next[i][j] = next[i][k]
        end if
      end for
    end for
  end for
  return d, next
```

Complexitats Resumides

SSSP

	Dijkstra	BFMS
$w \geq 0$	$O(mlg(n))$	$O(nm)$
$w \in (R)$	-	$O(nm)$

APSP

	Dijkstra	Johnson	BFMS	BFW
$w \geq 0$	$O(nmlg(n))$	-	$O(n^2m)$	$O(n^3)$
$w \in (R)$	-	$O(n^2lg(n))$ (G amb poques arestes)	$O(n^2m)$	$O(n^3)$