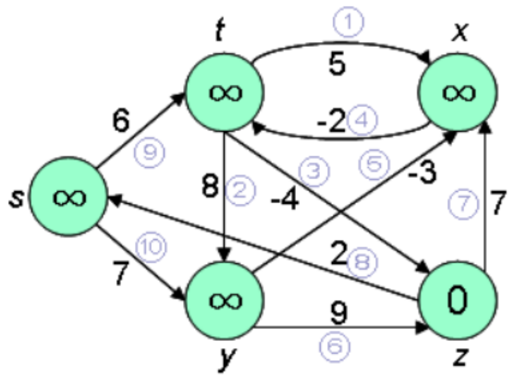
Algorithm

HW 10-第三組

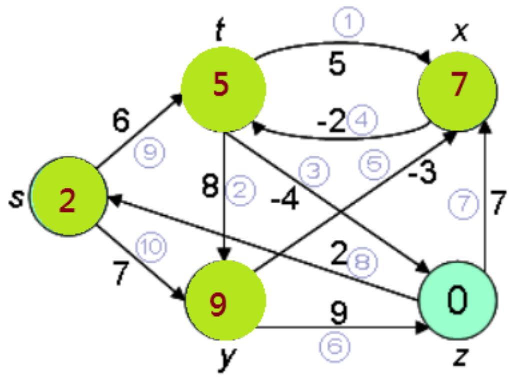
1.

a)

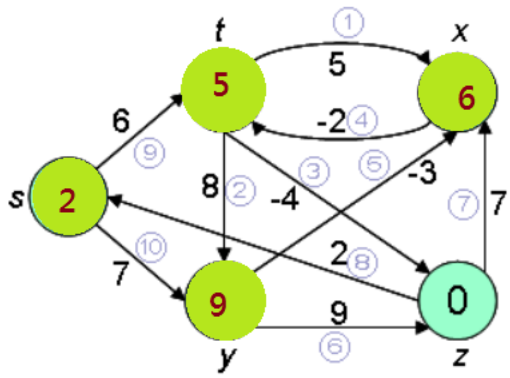
	z	s	t	x	y
d	0	∞	∞	∞	∞
π		NIL	NIL	NIL	NIL



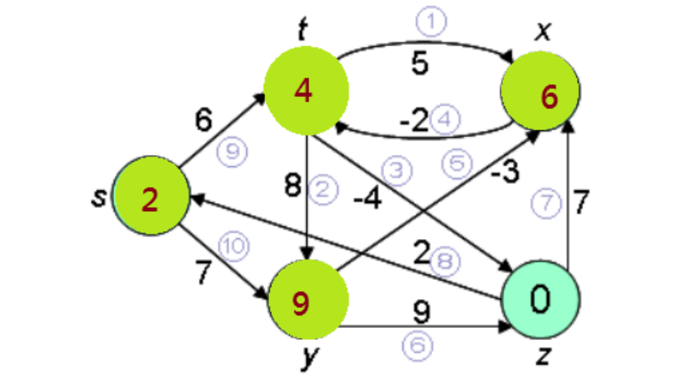
2	z	s	t	x	y
d	0	2	5	7	9
π		z	x	z	s

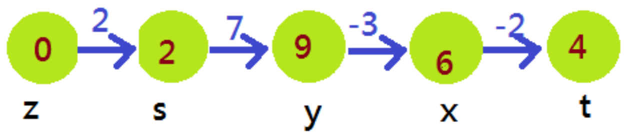


3	z	s	t	x	y
d	0	2	5	6	9
π		z	x	y	s



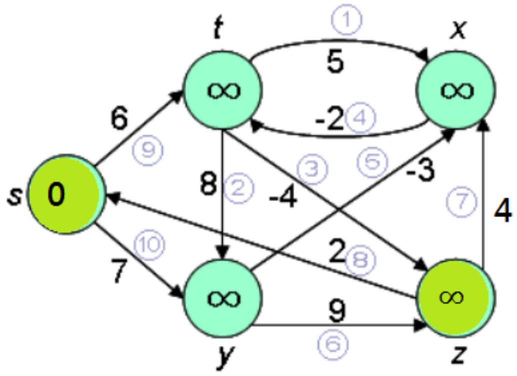
4	z	s	t	x	y
d	0	2	4	6	9
π		z	x	y	s



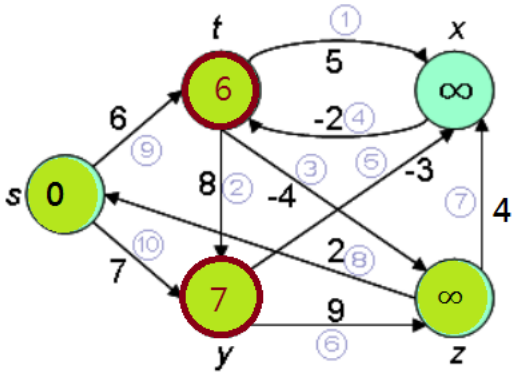


b)

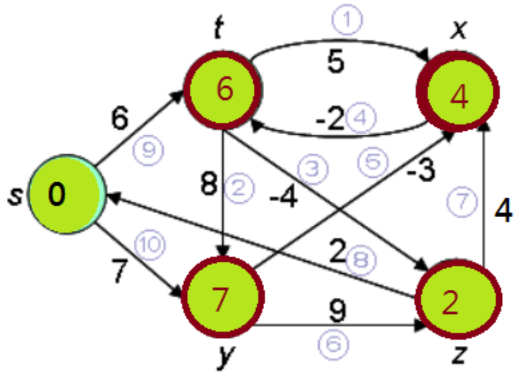
	s	t	x	y	z
d	0	∞	∞	∞	∞
π		NIL	NIL	NIL	NIL



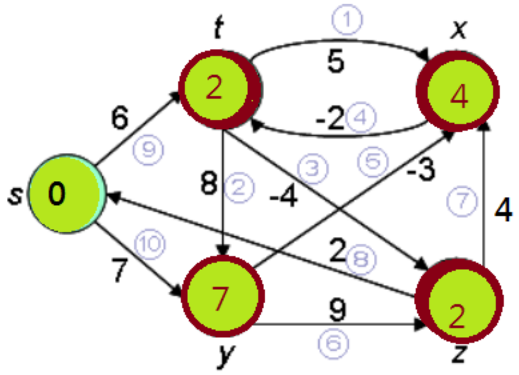
1	s	t	x	y	z
d	0	6	∞	7	∞
π		s	NIL	s	NIL



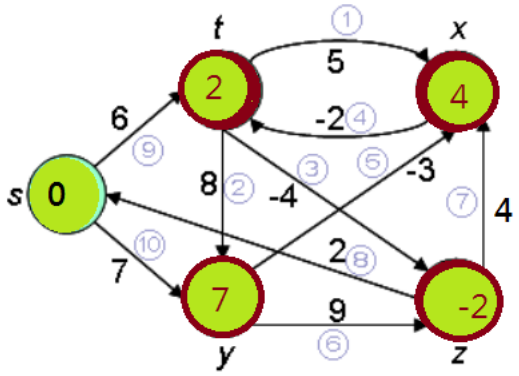
2	s	t	x	y	z
d	0	6	4	7	2
π		s	y	s	t

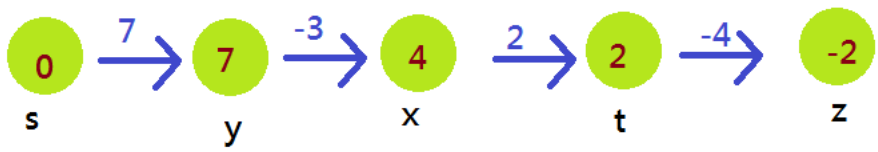


3	s	t	x	y	z
d	0	2	4	7	-2
π		x	y	s	t



4	s	t	x	y	z
d	0	2	4	7	-2
π		x	y	s	t

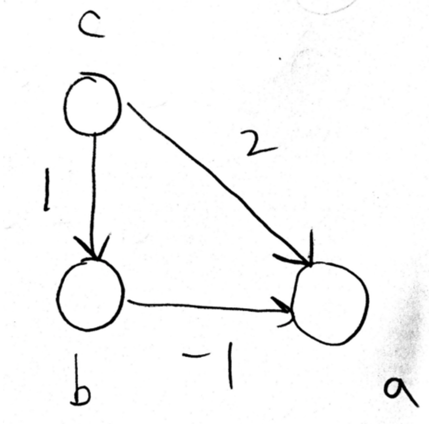




2.

題目為給定一個有權重，權重值為實數的有向 G=(V,E)，利用一個 O(|V|*|E|)的演算法找出對於 V 中的每個點 v，所有 V 中的其他點 u 到 v 的最短路徑中的最小值。例如有 b, c, d 三點可以走到 a，且最短路徑分別為 detla(b, a)=-1, detla(c, a)=2, detla(d, a)=1，所以 delta*(a)=-1。

想法為例如對於 a 點來說有兩種選擇，b->a 或是 c->b->a 分別是自已的鄰居就是起點，或是自己的鄰居為某最短路徑的一點，兩者取最小值，因此只需要修改 Bellman-Ford 演算法中的 relaxation 的部分。



Pseudocode:

Initialize(G,s) *//p[v]<-NIL, d[v]<-infinity*
for i = 1 to |V|
  for each edge uv in E
    if d[v] > min(w(u, v), w(u, v)+d[u])
      d[v] = min(w(u, v), w(u, v)+d[u])
      p[v] = p[u]

時間複雜度分析：最慘的情況為對於|V|個點每次都要做|E|個邊，因此為O(|V|*|E|)。

3.

BellmanFord(G,s)
π(v) = NIL ; d[v]= ∞, ∀ v ; d[s] = 0; *//initialize (G,s)*
for i=1 to |v|-1
  for each edge uv in E
    if d[v] > d[u] + w(u,v) : d[v] = d[u] + w(u,v) , π(v) = u
for each edge uv in E

```
if d[v] > d[u] + w(u,v)
      c[v] = 1          // c=1 assure we will stop when all the
 cycle is covered
      curr = v
      p = π(v)
      push v into stack
      while c[p] == 0
            curr = π(curr)
            c[curr] =1
            push curr into stack
      return stack      //vetrix in negative cycle are all in the
stack
```

4.

想法：把一個點拆成 2 個點，2 個點中間的邊的 weight 為原本的點的 weight，然後就可以用 DAG-Shortest-Paths 演算法找最長路徑，跟原本的演算法差在設定初值時 d[v] ← $-\infty$，以及 if (d[v]>d[u]+w(u,v)) 改成 (d[v]<d[u]+w(u,v))

```
DAG-Shortest-Paths(G,s){
   Topologically sort the vertices of G ;
   Initialize(G,s)     // π[v] ← nil , d[v] ← -∞ for all v , d[s] ←0
   for each vertex u taken in topological order do{
        for each vertex v in Adj[u] do{        //Relax(u,v)
              if (d[v] <d[u]+w(u,v)){
                    then d[v] ← d[u]+w(u,v) ;
                    π[v] ← u ;
              }
        }
   }
}
```

紅字為跟原本演算法不同之部分

Time complexity：DAG-Shortest-Paths 需 O(V+E)，由於一個點拆成 2 個點，new_E=E+V ，new_V=2*V，所以需 O(2*V+V+E) = O(3*V+E)=O(V+E)

5.

a) s as the source

1	s	t	y	x	z
d	0	∞	∞	∞	∞
π	NIL	NIL	NIL	NIL	NIL

2	s	t	y	x	z
d	0	3	5	∞	∞
π	NIL	s	s	NIL	NIL

3	s	t	y	x	z
d	0	3	5	9	∞
π	NIL	s	s	t	NIL

4	s	t	y	x	z
d	0	3	5	9	11
π	NIL	s	s	t	y

5	s	t	y	x	z
d	0	3	5	9	11
π	NIL	s	s	t	y

b) Z as the source

1	s	t	y	x	z
d	∞	∞	∞	∞	0
π	NIL	NIL	NIL	NIL	NIL

2	s	t	y	x	z
d	3	∞	∞	7	0
π	z	NIL	NIL	z	NIL

3	s	t	y	x	z
d	3	6	8	7	0
π	z	s	s	z	NIL

4	s	t	y	x	z
d	3	6	8	7	0
π	z	s	s	z	NIL

5	s	t	y	x	z
d	3	6	8	7	0
π	z	s	s	z	NIL

6.

Suppose that s is the source vertex.
1.Check $s.\pi=$ NIL and $s.d=0$.
2.For each $v\in V\setminus\{s\}$:
If $v.\pi\neq$ NIL, check $v.d=v.\pi.d+w(v, v.\pi)$.
If $v.\pi=$ NIL, check $v.d=+\infty$.

如以上條件有其中之一為 false，表示此程式的 output 是錯誤的；

如以上條件皆為 true，跑一次 Bellman-Ford 演算法，relax each edge (u, v) ∈ E，如有 v.d 改變，表示此程式的 output 是錯誤的。

先看所有 vertex，再看所有 edge，time complexity=O(V+E)

7.

想法：將原本的 Heap 改成固定大小的陣列，其大小為：

(|V|-1)W+1

Pseudocode:

```
Dijkstra_m (G, s)
π[v] ← NIL, d[v] ← ∞ ∀ v;     d[s] ← 0
A[0] ← v1, A[1~VW] ← empty,   A[infinity] ← v2~vn
    while A[ i ] is not empty
        u =A[ i ].remove( )
    for each v ∈ adj[u]
        if v∈ A && d[v] > d[u]+w(u, v)
            π[v] ← u
            d[v] ← d[u]+w(u, v)
            v move to the new place in A[ ]
    i++
```

複雜度分析：

1. While 的部分為 $O(VW)$
2. for 的部分為 O(E)

所以 $T(n) = O(VW + e)$