Programming Assignment 4

Implement the Bellman-Ford algorithm

104303206 黃筱晴

一、加速版 Bellman – Ford algorithm

原本 Bellman – Ford 演算法直接對所有 edge 進行 Relaxation，總共做 V-1 次。加速版用 Breadth-First-Search 的方式，從起點開始，對鄰邊進行 Relaxation，一層一層向外擴展，直到修正所有鄰點的最短路徑長度。

參考維基百科 Bellman – Ford algorithm 條目中提到的佇列最佳化來實作加速版。

```
modified_BellmanFord(G,s,d)

Initialize();

queue    Q;//  準備一個佇列來操作 Relaxation 向外擴展的順序

bool    color[V]//  準備一個陣列紀錄哪些 vertex 已經被放進去過了

Q.push(s);//把起點放進 Q

color[s]=1//紀錄起點已經放進去過了

while( Q!=Φ)

   u=Q.front();//從 Q 中拿出一點

   for (所有 u 的鄰邊 uv)

      Relaxation();

      if( color[v]!=1 )

         Q.push(v);//把 v 放進 Q

         color[v]=1//紀錄 v 已經放進去過了

再做一次以上步驟，若又找到可修正的邊，則return False//存在負環

//確定無負環後，利用π 回推路徑，並使用一個 stack 來裝

stack AnsPath; int AnsWeight=0;
```

```
now=d;//從終點開始往回找
last=π [now];
while(now!=s)
    AnsWeight += w[last][now];
    AnsPath.push(now);
    now=last;
    last=π [now];
AnsPath.push(s);//此時 stack 中就是從 s 走到 d 的完整最短路徑
return True;
```

二、加速版時間複雜度分析

從 Pseudo code 中觀察，第七行的 while(Q!=Φ)總共會拿出 V 個頂點；每拿出一個頂點，第九行的 for 迴圈最多有 E 個鄰邊可以 Relaxation()。最差情形的時間複雜度為 O(VE)和原版相同。但是通常每拿出一個頂點，它應該不會有那麼多鄰邊。最差情形發生在超密集的圖上(每個頂點都相鄰)。如果是稀疏一點的圖，加速版可以快許多。

三、原版與加速版的時間比較

測試環境：ASUS UX303L筆電

處理器：Intel(R) Core(TM) i5-5200U CPU　@ 2.20GHz　2.19GHz

RAM：8.00GB

系統類型：64位元作業系統 x64型處理器

(1) 1000點 2000條邊。

```
original time: 314.76 ms
modified time: 101.184 ms
0 325 784 42 43 255 860 840 904 289 367 434 378 979 998
309
```

original time:314.76 ms

modified time:101.184 ms

(2) 5000點 100萬條邊

```
original time: 926834 ms
modified time: 2957 ms
0 3063 4995
73
```

original time:926834 ms

modified time:2957 ms

(3) 5000點 150萬條邊

```
original time: 1.39308e+006 ms
modified time: 3357 ms
0 454 4995
61
```

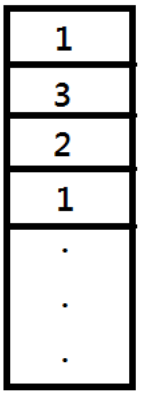
original time:$1.39308*10^6$ ms

modified time:3357 ms

四、輸出Negative weight cycle的方式

再度發現可以修正的邊uv時，從u開始，利用π 往回推，直到遇到v即是一個負環。

```
if d[v]>d[u]+w(u,v)
   stack    Ans;
   int HaveFinded[n];//紀錄 1:已經被放進 stack    0:未放進 stack
   Ans.push(v);
   HaveFinded[v]=1;
   while(HaveFinded[u]!=1)
      v=u; u=π [v];
      Ans.push(v); HaveFinded[v]=1;
   AnsPath.push(u);//此時負環已經在 stack 的頂端
```



輸出 1 3 2