79882301 周德智 Advanced Graph Algorithms and Applications HW4

本次作業有兩種題目可以選擇，第一題爲標出 Dijkstra’ s algorithm 的時間、空間復雜度，以及演算法的正確性。下圖爲 Dijkstra’ s algorithm

```
      begin
1.        S ← {v0};
2.        D[v0] ← 0;
3.        for each v in V − {v0} do D[v] ← l(v0, v);
4.        while S ≠ V do
              begin
5.                choose a vertex w in V − S such that D[w] is a minimum;
6.                add w to S;
7.                for each v in V − S do
8.                    D[v] ← MIN(D[v], D[w] + l(w, v))
              end
      end
```

Fig. 5.24. Dijkstra's algorithm.

Running cost

n 個點 E 個邊

在一般情況每一步都要找一個最小邊加入到已判定的 set 裡面，所以 search 的 cost 是 n + n-1 + n-2 + n-3 ... + 1 = n(n+1)/2 => O(n^2)

不過在稀疏矩陣可以考慮使用 Fibonacci heap 使 search 的步驟縮減到 O(E + nlogn)

Space Complexity

空間複雜度在一般情況要把所以點到點的距離都記錄起來，最多的邊可以有 n(n-1)/2 個邊所以空間複雜度也是在 O(n^2)，

在稀疏矩陣可以假設邊的數量遠小於 n^2，所以估計空間的使用量用 O(E)會比較適合

Correctness

V_{set}:所有的 vertices

S_{set}:已完全 Dijkstra 的 graph(到每一點都已知最短路徑)

V_{min}:在 V_{set}-S_{set} 選擇的最小邊的點 E_{min}:V=>V_{set}-S_{set} 的最小邊

D_{min}:從起點到 E_{min} 的最小 cost 的合

考慮起點在 S_{set} 內，然後終點在 V_{set} - S_{set} 內，現在要選擇一個 E_{min} 在 V_{set} 到 V_{set} - S_{set} 內，使得 D_{min} 是目前 S_{set} 內的起點到 V_{set}-S_{set} 內的任一點的 cost 最小之路徑，如果找一個 edge E_t 從 V_{set} 到 V_{set}-S_{set} 內且不等於 E_{min}，可知 D_t>=D_{min}，如此一來包含 E_t 所形成的最短路徑必定總距離會大於等於包含 E_{min} 所形成的最短路徑，所以在每一次選擇 vertex 來加入都依照選擇最小邊之原則最後必定可以滿足由起點到終點之最短距離。

電機碩二 79782301 陳泰均

Homework 4

本次作業有兩種題目可以選擇，第一題為標出 Dijkstra's algorithm 的時間、空間復雜度，以及演算法的正確性。下圖為 Dijkstra's algorithm

```
begin
1.      S ← {v0};
2.      D[v0] ← 0;
3.      for each v in V − {v0} do D[v] ← l(v0, v);
4.      while S ≠ V do
            begin
5.              choose a vertex w in V − S such that D[w] is a minimum;
6.              add w to S;
7.              for each v in V − S do
8.                  D[v] ← MIN(D[v], D[w] + l(w, v))
            end
end
```

Fig. 5.24. Dijkstra's algorithm.

第二題，用 Peterson　Diameter=3、Degree=3 做出一個 routing algorithm，以下有兩個提示:

1. 每點的 address 要定義好
2. 可以 self routing

Theorem :從 V_0 到每個 Vertex 的最短路徑需要 O(n^2)。

Proof.步驟第 7 和 8 需要執行 N 次，而從步驟 4 到步驟 8 則執行 N 次，所以總共的執行時間會為 O(n^2)m。

Dijkstar 算法的正確性证明：

設兩個集合 A、B（A 為已知最短路徑的點的集合，B 為未知最短路徑的点的集合）此时：(1) A 至少包含起始點（S），B 不可以為空集（因為為空集合的時候選點過程已结束）。(2) 已選出 B1 为最短路徑的點。(此时路徑的特點是：B1 之前的所有点都是在 A 中（前提），B1 是 B 中所有點中到 S 點距離最短的點）

反證法：

前提：假设此條路（L1）徑並不是最短路徑。設從 S 到 B1 的最短路径为 S。。B1（L2）。因為不同於 L1：

情況 1：B1 之前所有點在 A 中：但由於相互比較，此時 B1 上的值一定是在{L|L：

S->S 到 A'的最短路徑->B1}的集合中是最小的，同时其它的可能就會破壞“最短路徑”這個条件。所以，情况 1 是不成立的。

情况 2：L2 中出 B1 外至少存在一個點是屬於 B 的，而在這些點中排在第一個的 B 點設為 B'，可知 B'前的所有的點都是 A 中的點，那麼得出：S 到 B'的距離小於 S 到 B1 的距離，這與前提是矛盾的，所以情况 2 是不成立的。
綜合上述：Dijkstar 的算法成立。

Homework No.4

黃郁書 79882304

第一題：標出 Dijkstra's algorithm 的時間、空間複雜度，以及演算法的正確性。下圖為 Dijkstra's algorithm

```
      begin
1.        S ← {v0};
2.        D[v0] ← 0;
3.        for each v in V − {v0} do D[v] ← l(v0, v);
4.        while S ≠ V do
              begin
5.                choose a vertex w in V − S such that D[w] is a minimum;
6.                add w to S;
7.                for each v in V − S do
8.                    D[v] ← MIN(D[v], D[w] + l(w, v))
              end
      end
```

Fig. 5.24. Dijkstra's algorithm.

Ans.

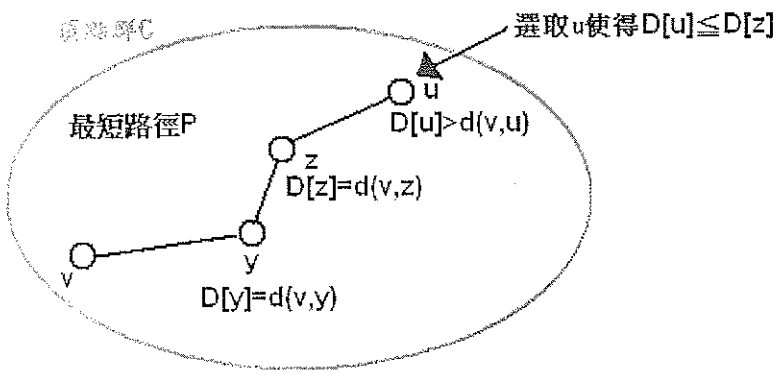
時間複雜度：

已知有 n 個頂點與 m 個邊的加權圖 G，且每個邊權重均為非負值。則 Dijkstra algorithm 的時間複雜度為 O(m log n)。

演算法的正確性：

假設在 V 中的某個頂點 t，D[t] > d(v,t)，且令 u 為第一個從 V 中移除的頂點並加入到 C 頂點群中，且 D[u] > d[u,v]的頂點。從 v 到 u 有一條最短路徑 P。當 u 將從 V 中移除的那一瞬間，令 z 為 P 上第一個不在 C 裡面的頂點。又令 y 為 z 再路徑 P 上的前一個頂點。D[y] = d(v,y)，因為 u 為第一個不正確的頂點。當 y 被加入至 C 頂點群時，D[z]□D[y] + w((y,z)) = d(v,y) + w((y,z))，但由於 z 是再 v 到 u 之最短路徑上的下一個頂點，即 D[z] = d(v,z) ，但目前卻是將 u 從 V 中移除的那一瞬間，最短路徑的子路徑，仍為最短路徑。因此，既然 z 在 v 到 u 的最短路徑上，即 d(v,z) + d(z,u) = d(v,u) ，又因無負值權重的邊，故 d(z,u) □0。

因此，D[u] □D[z] = d(v,z) + d(z,u) = d(v,u)，與 u 之定義矛盾。故不沒有這樣的頂點 u。所以選入頂點群 C 的點必為最短路徑上的點。因此，dijkstra algorithm 所演算的為結果為正確的最短路徑。



Homework 4

本次作業有兩種題目可以選擇，第一題為標出 Dijkstra's algorithm 的時間、空間復雜度，以及演算法的正確性。下圖為 Dijkstra's algorithm

```
    begin
1.      S ← {v0};
2.      D[v0] ← 0;
3.      for each v in V − {v0} do D[v] ← l(v0, v);
4.      while S ≠ V do
            begin
5.              choose a vertex w in V − S such that D[w] is a minimum;
6.              add w to S;
7.              for each v in V − S do
8.                  D[v] ← MIN(D[v], D[w] + l(w, v))
            end
    end
```

Fig. 5.24. Dijkstra's algorithm.

B+

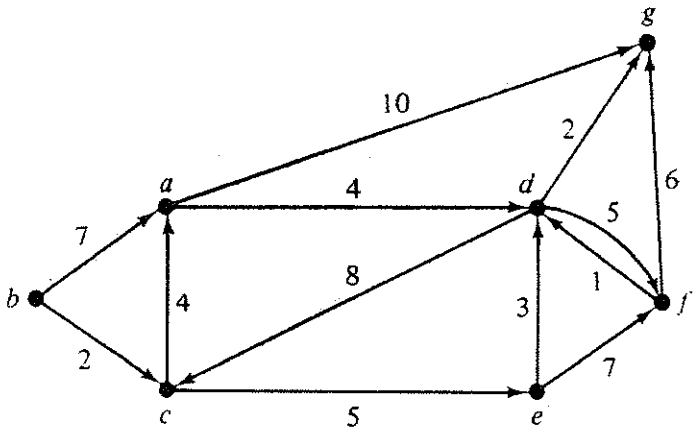


Figure 8.12　A simple weighted digraph.

Step 0. 起點 b

b-a=∞, b-b=0, b-c=∞, b-d=∞, b-e=∞, b-f=∞, b-g=∞

Step 1. 自 b 延伸（b-b 最小路徑值爲 0）

Label b, b is finalized

b-a=7 取代 b-a=∞

b-c=2 取代 b-a=∞

取最小值 2（自 b 至 c 爲最小，路徑值爲 2）

Step 2. 自 c 延伸（b-c 最小路徑值爲 2）

Label c, c is finalized

c-a=4 → b-a=6 取代 b-a=7

c-e=5 → b-e=7 取代 b-e=∞

取最小值 6（自 b 至 a 爲最小，路徑值爲 6）

Step 3. 自 a 延伸（b-a 最小路徑值爲 6）

Label a, a is finalized

a-d=4 → b-d=10 取代 b-d=∞

a-g=10 → b-g=16 取代 b-g=∞

b-e=7

取最小值 7（自 b 至 e 爲最小，路徑值爲 7）

Step 4. 自 e 延伸（b-e 最小路徑值爲 7）

Label e, e is finalized

e-d=3 → b-d=10 原 b-d=10 不可取代，亦可不取代

e-f=7 → b-f=14 取代 b-f=∞

b-g=16

取最小值 10（自 b 至 d 爲最小，路徑值爲 10）

Step 5. 自 d 延伸（b-d 最小路徑值爲 10）

Label d, d is finalized

d-f=5 → b-f=15 比原 b-f=14 不不不大，不取代

d-g=2 → b-g=12 取代 b-g=16

取最小值 12（自 b 至 g 爲最小，路徑值爲 12）

Step 6.

Label g, g is finalized

因目的地爲 g，g 已經 finalized，所以結束。

a	*b*	*c*	*d*	*e*	*f*	*g*	
∞	0	∞	∞	∞	∞	∞	Starting vertex *b* is labeled 0
7	0	2	∞	∞	∞	∞	Successors of *b* get relabeled
7	0	2	∞	∞	∞	∞	Smallest label (*c*) becomes final
6	0	2	∞	7	∞	∞	Successors of *c* get relabeled
6	0	2	∞	7	∞	∞	Smallest label (*a*) becomes final
6	0	2	10	7	∞	16	Successors of *a* get relabeled
6	0	2	10	7	∞	16	Smallest label (*e*) becomes final
6	0	2	10	7	14	16	Successors of *e* get relabeled
6	0	2	10	7	14	16	Smallest label (*d*) becomes final
6	0	2	10	7	14	12	Successors of *d* get relabeled
6	0	2	10	7	14	12	Vertex *g* gets its final label and we are done

Figure 8.13 The array *dist* as the shortest path from vertex *b* to vertex *g* is found.

時間複雜度: $n=|V|$, $m=|E|, O(m\log n)$

若無方向性，則時間複雜度爲 $O(|n|^2)$，空間複雜度為 $O(2|n|)$

進階圖形演算法

Homework No.4

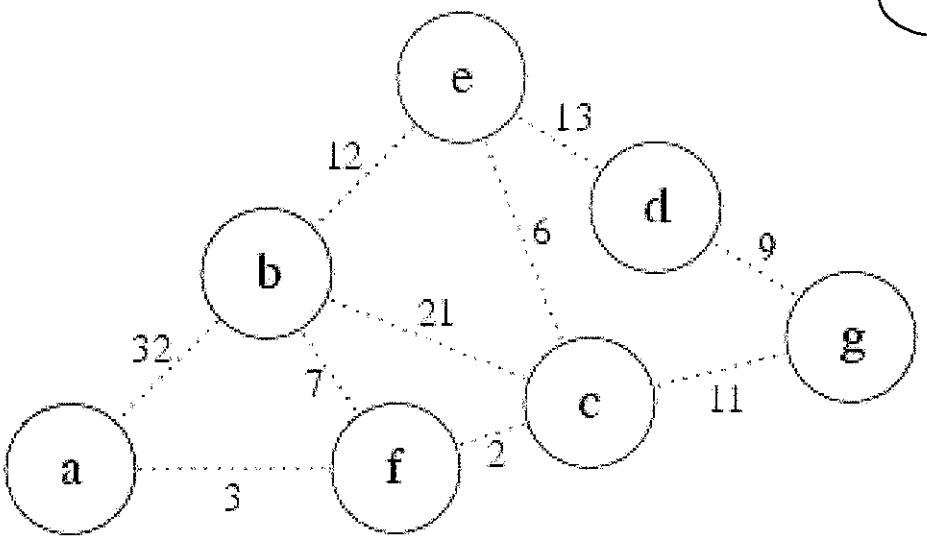


Figure 1: Dijkstra 1

電機產碩班

學生：方瑞鋐

學號：79780105

Homework 4

本次作業有兩種題目可以選擇，第一題為標出 Dijkstra's algorithm 的時間、空間復雜度，以及演算法的正確性。下圖為 Dijkstra's algorithm

```
      begin
1.        S ← {v0};
2.        D[v0] ← 0;
3.        for each v in V − {v0} do D[v] ← l(v0, v);
4.        while S ≠ V do
              begin
5.                choose a vertex w in V − S such that D[w] is a minimum;
6.                add w to S;
7.                for each v in V − S do
8.                    D[v] ← MIN(D[v], D[w] + l(w, v))
              end
      end
```

Fig. 5.24. Dijkstra's algorithm.

代克思托演算法 (Dijkstra's algorithm)

Dijkstra's algorithm 是以某一節點爲出發點，計算從該節點出發到所有其他節點的最短路徑。

首先以某一節點當作出發點，在與其相連且尚未被選取的節點裡，選擇加入離出發點距離最短的節點，並且透過新增的節點更新到達其他節點的距離。如此重覆加入新節點，直到所有的節點都被加入爲止。

1. 起始。

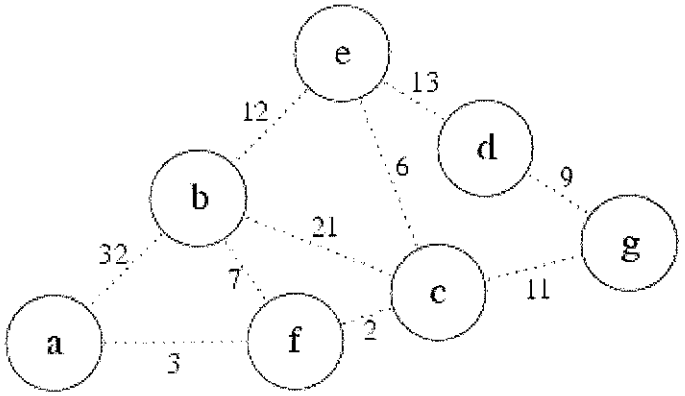


Figure 1: Dijkstra 1

步驟								即將新增
	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a

2. 從 a 開始，與 a 相連的節點有兩個 (f, b)。
   a 與 a 的距離爲 0。 (新增 0)
   f 與 a 的距離爲 3。 (新增 3)
   b 與 a 的距離爲 32。 (新增 32)

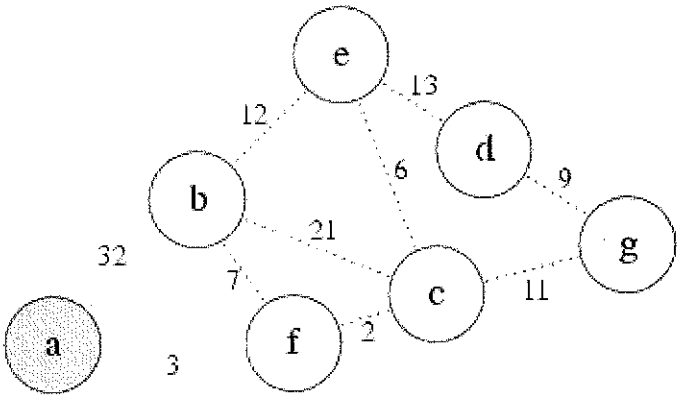


Figure 2: Dijkstra 2

與 a 距離最短的節點爲 f。

步驟								即將新增

	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a
2	0	32	-	-	-	3	-	f

3. 加入 f 後，與 f 相連的節點有兩個 (c, b)。
   透過 f ， c 與 a 的距離爲 3 + 2 = 5。 (新增 5)
   透過 f ， b 與 a 的距離爲 3 + 7 = 10。 (取代 32)

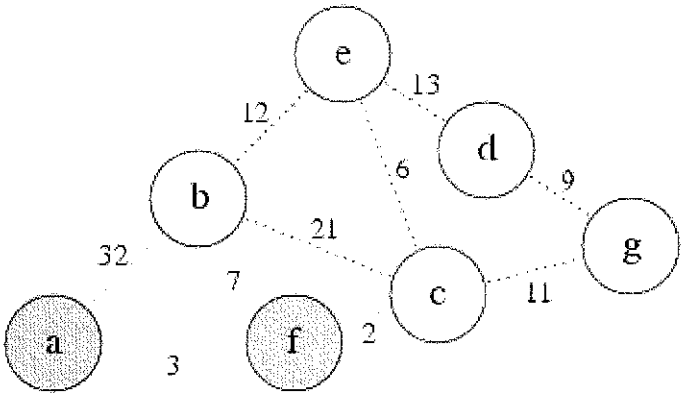


Figure 3: Dijkstra 3

與 a 距離最短的節點爲 c。

步驟								即將新增
	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a
2	0	32	-	-	-	3	-	f
3	0	10	5	-	-	3	-	c

4. 加入 c 後，與 c 相連的節點有三個 (b, e, g)。
   透過 c ， b 與 a 的距離爲 5 + 21 = 26。 (保留 10)
   透過 c ， e 與 a 的距離爲 5 + 6 = 11。 (新增 11)
   透過 c ， g 與 a 的距離爲 5 + 11 = 16。 (新增 16)

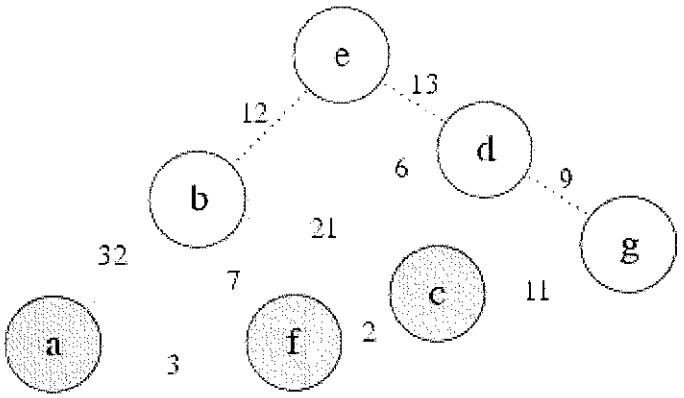


Figure 4: Dijkstra 4

與 a 距離最短的節點爲 b。

步驟								即將新增
	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a
2	0	32	-	-	-	3	-	f
3	0	10	5	-	-	3	-	c
4	0	10	5	-	11	3	16	b

5. 加入 b 後，與 b 相連的節點有一個 (e)。
   透過 b ， e 與 a 的距離爲 10 + 12 = 22。 (保留 11)

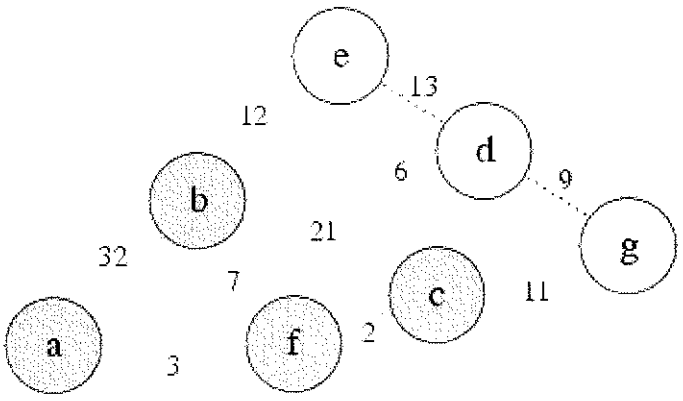


Figure 5: Dijkstra 5

與 a 距離最短的節點爲 e。

步驟								即將新增
	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a
2	0	32	-	-	-	3	-	f
3	0	10	5	-	-	3	-	c
4	0	10	5	-	11	3	16	b
5	0	10	5	-	11	3	16	e

6. 加入 e 後，與 e 相連的節點有一個 (d)。
透過 e ， d 與 a 的距離爲 11 + 13 = 24。 (新增 24)

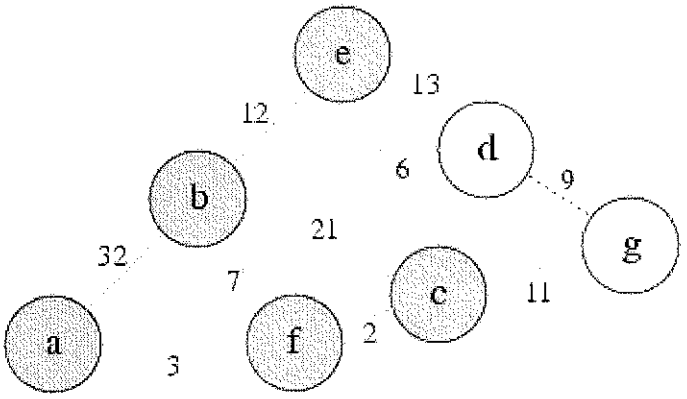


Figure 6: Dijkstra 6

與 a 距離最短的節點爲 g。

步驟								即將新增
	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a
2	0	32	-	-	-	3	-	f
3	0	10	5	-	-	3	-	c

4	0	10	5	-	11	3	16	b
5	0	10	5	-	11	3	16	e
6	0	10	5	24	11	3	16	g

7. 加入 g 後，與 g 相連的節點有一個 (d)。
   透過 g ， d 與 a 的距離爲 16 + 9 = 25。 (保留 24)

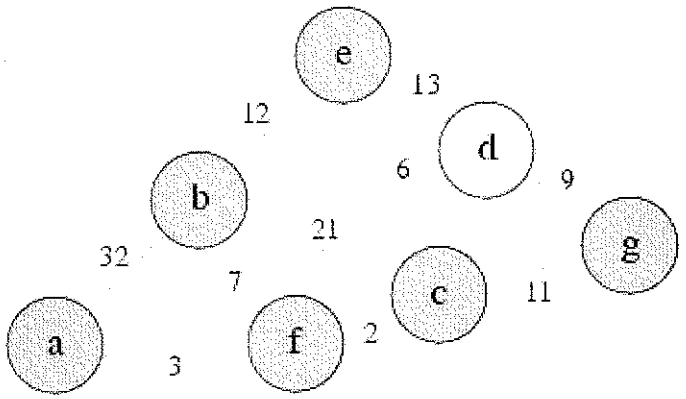


Figure 7: Dijkstra 7

與 a 距離最短的節點爲 d。

步驟								即將新增
	a	b	c	d	e	f	g	
1	-	-	-	-	-	-	-	a
2	0	32	-	-	-	3	-	f
3	0	10	5	-	-	3	-	c
4	0	10	5	-	11	3	16	b
5	0	10	5	-	11	3	16	e
6	0	10	5	24	11	3	16	g
7	0	10	5	24	11	3	16	d

8. 加入 d 後，所有結點皆已加入，所以 Dijkstra's algorithm 完成。

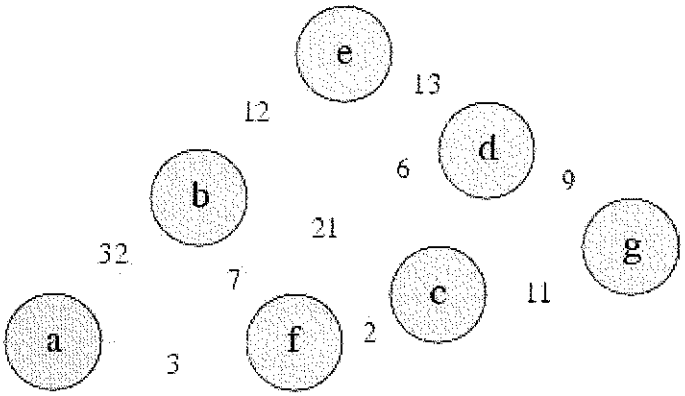


Figure 8: Dijkstra 8

所有節點與 a 的最短距離如下：

	a	b	c	d	e	f	g
a	0	10	5	24	11	3	16

時間複雜度

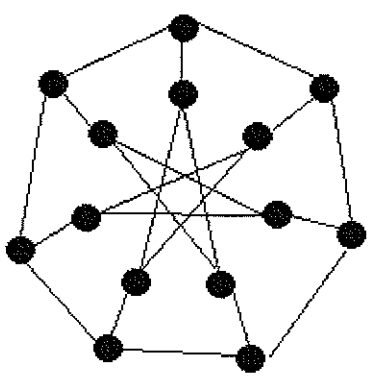
時間複雜度為 $O(|V|^2)$。

空間複雜度

空間複雜度為 $O(2|V|)$。

第二題，用 Peterson　Diameter=3、Degree=3 做出一個 routing algorithm，以下有兩個提示:

1. 每點的 address 要定義好
2. 可以 self routing



學號:79780110 姓名:黃耀正

Homework 4

本次作業有兩種題目可以選擇，第一題為標出 Dijkstra's algorithm 的時間、空間復雜度，以及演算法的正確性。下圖為 Dijkstra's algorithm

```
     begin
1.       S ← {v0};
2.       D[v0] ← 0;
3.       for each v in V − {v0} do D[v] ← l(v0, v);
4.       while S ≠ V do
             begin
5.               choose a vertex w in V − S such that D[w] is a minimum;
6.               add w to S;
7.               for each v in V − S do
8.                   D[v] ← MIN(D[v], D[w] + l(w, v))
             end
     end
```

Fig. 5.24. Dijkstra's algorithm.

Ans :

time complexity :$\underline{O(V^2)}$

space complexity :O(V)

証明: