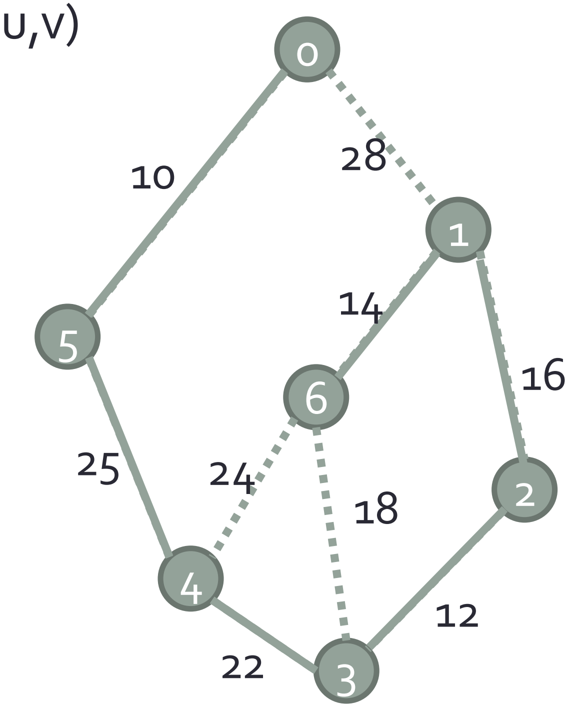
GRAPH 3

Michael Tsai

2013/04/30

Prim’s Algorithm

- T={}
- TV={0}
- while(T少於n-1條edge) {
  - 找出一條$u \in TV$但$v \notin TV$中cost最小的edge (u,v)
  - 如果找不到就break;
  - add v to TV
  - add (u,v) to T
- }
- 如果T中少於n-1條edge, 就output失敗

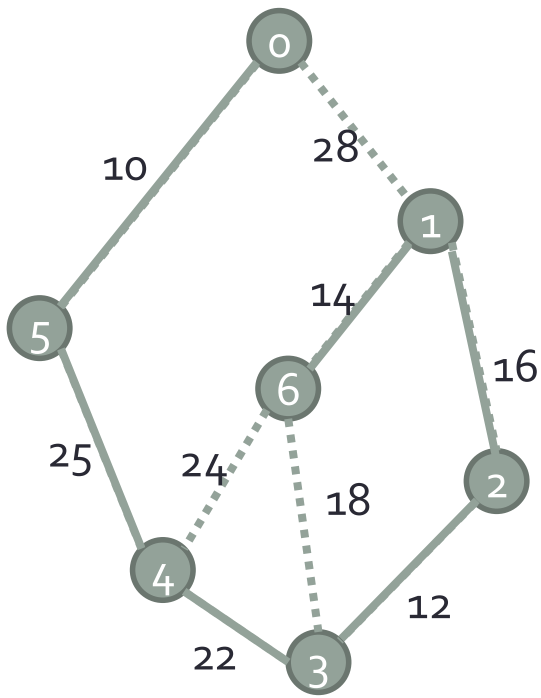


Prim's Algorithm

u.key: 記錄到G_A最短的邊的weight
u.pi: 紀錄上面講的那條邊的另外一端是哪個vertex

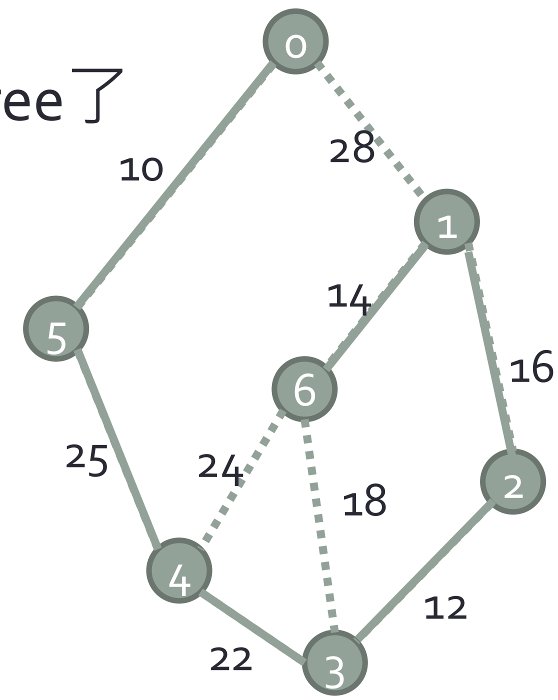
```
MST-PRIM(G,w,r)
for each u ∈ G.V
        u.key=∞
        u.pi=NIL
r.key=0
Q=G.V
while Q≠{}
        u=EXTRACT-MIN(Q)
        for each v ∈G.Adj[u]
                if v∈ Q and w(u,v)<v.key
                        v.pi=u
                        v.key=w(u,v)

A={(v,v.pi):v∈V-{r}}
```



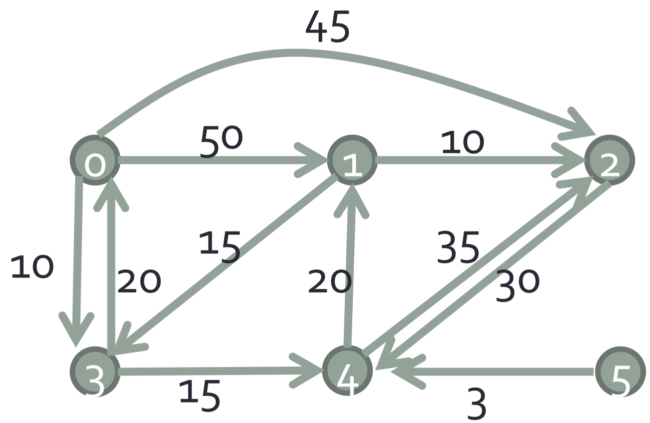
Sollin's algorithm

- 一開始每個vertex自己是一個forest
- 保持都是forest
- 每個stage, 每個forest都選擇一個該forest連到其他forest的edge中cost最小的一個
- 執行到沒有edge可以選了, 或者是變成tree了
- (Horowitz 6.3.3)



Shortest paths

- 目標: 找出vertex u到graph G中任何一點的最短距離



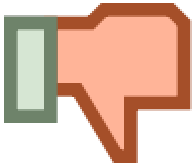
TO VERTEX	PATH	LENGTH
3	0 3	10
4	0 3 4	25
1	0 3 4 1	45
2	0 2	45
5	No path	∞

Shortest paths

- 怎麼找Shortest paths?

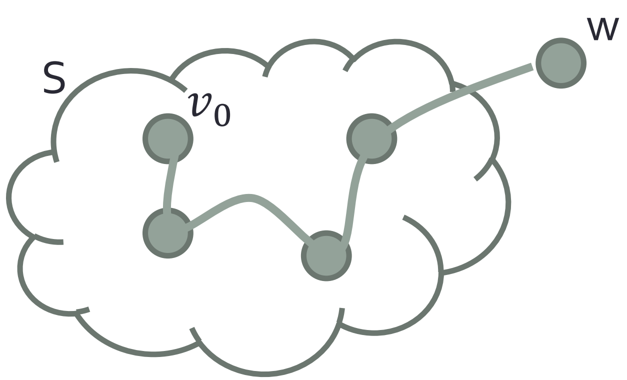
- 從起始點u開始, 把所有path的排列列出來
- 找出到點v所有paths中距離最短的那一條

- 時間複雜度?
  - 要把所有的path列出來, 所以需要花 $O(|V|!)$
  - 需要更有效率的演算法!



Dijkstra's algorithm

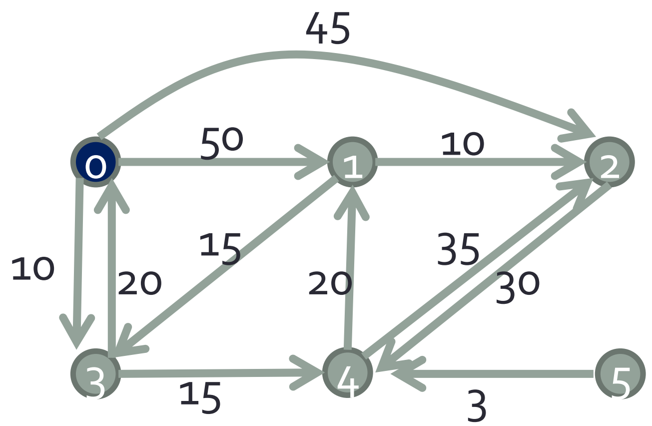
- Set S裡面有一些已經加入的vertex (包括起始點v_0)
- S裡面的vertex都已經找到由起始點v_0出發的最短路徑
- w不在S中, 則有distance[w]紀錄從v_0經過S中的任意個vertex後, 最後到達w的這樣形式最短路徑的距離



Dijkstra’s algorithm

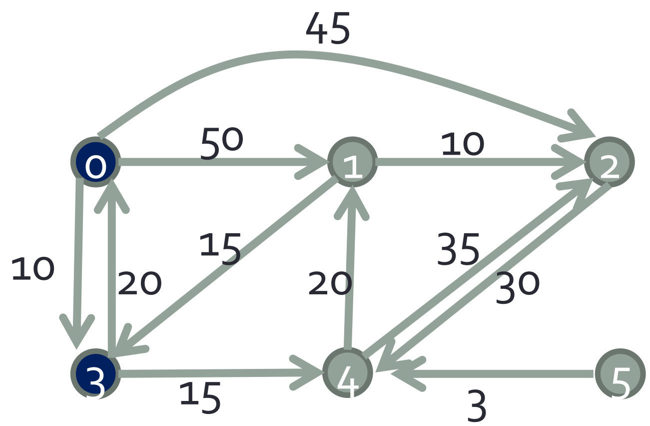
- 一開始S裡面只有v_0
- 每次選擇u, 為distance[u]中最小的
- 選進S以後, v_0到u的shortest path就找到了
- 所以找到shortest path的順序會是由最短的找到最長的
- 把u選進去S以後, 就要把所有u的edge <u,w>都看一遍:
- if distance[w]>distance[u]+cost(<u,w>)
-     distance[w]=distance[u]+cost(<u,w>)

例子



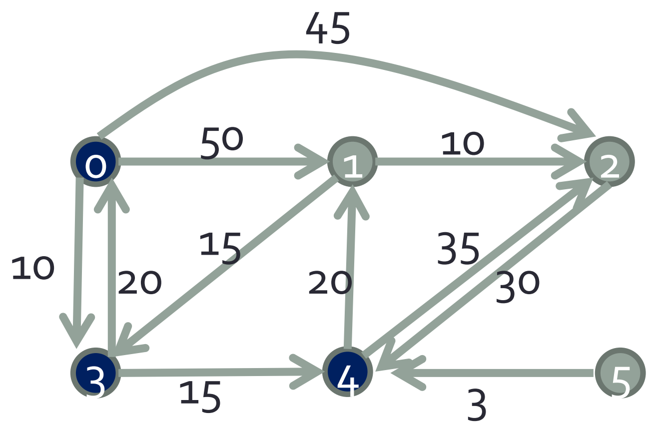
selection	0	1	2	3	4	5
Initial	0	50	45	$\boxed{10}$	∞	∞

例子



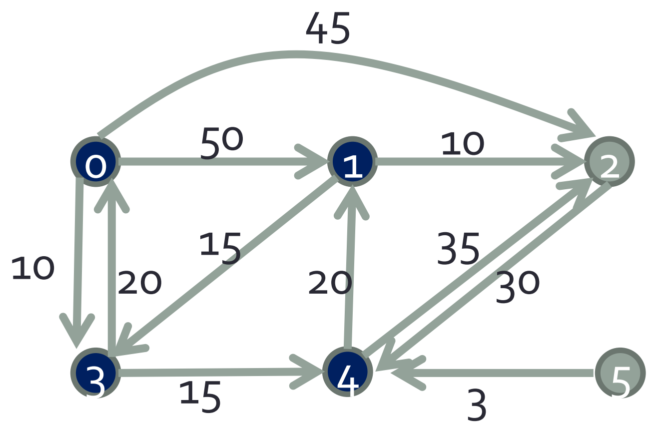
selection	0	1	2	3	4	5
Initial	0	50	45	10	∞	∞
3	0	50	45	10	$\boxed{25}$	∞

例子



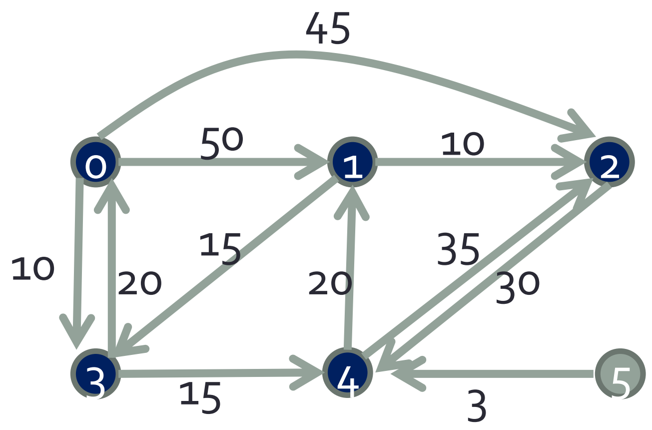
selection	0	1	2	3	4	5
Initial	0	50	45	10	∞	∞
3	0	50	45	10	25	∞
4	0	45	45	10	25	∞

例子



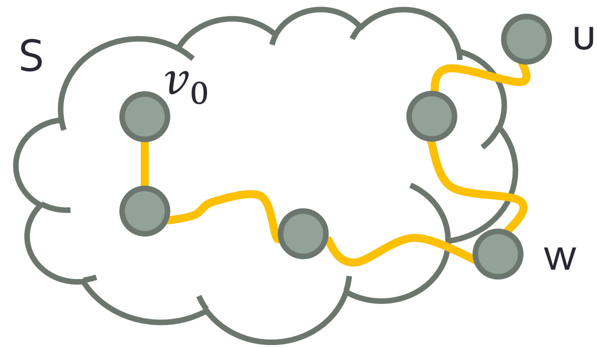
selection	0	1	2	3	4	5
Initial	0	50	45	10	∞	∞
3	0	50	45	10	25	∞
4	0	45	45	10	25	∞
1	0	45	$\boxed{45}$	10	25	∞

例子



selection	0	1	2	3	4	5
Initial	0	50	45	10	∞	∞
3	0	50	45	10	25	∞
4	0	45	45	10	25	∞
1	0	45	45	10	25	∞
2	0	45	45	10	25	∞

小證明



- 要怎麼證明演算法是正確的?
- 如果下一個選擇的vertex為u, 為什麼v_0到u的最短path會正好是前面的vertex都在S中, 只有u不在S中呢?

- 反證法: **假設v_0到u的最短path 經過vertex w, 且w不在S中**
- 則這條path中會有一段是v_0到w, 而且長度比較短 (因為是sub path)
- 可是這樣的話, w之前應該就要被選過了 (應該要在S中) (contradiction)
- 所以v_0到u的最短path前面的vertex都在S中, 只有u不在S中

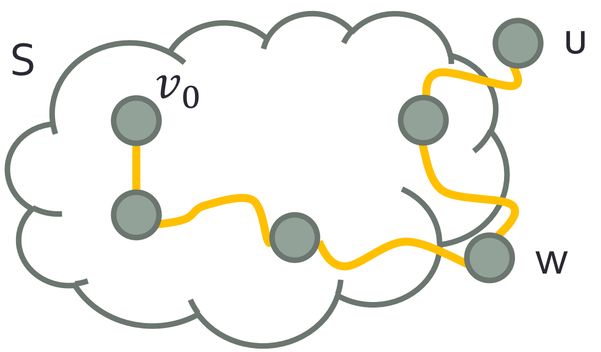
更多動畫例子

- http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/DijkstraApp.shtml?demo1

- Time complexity?
- 答案: $O(|V|^2)$
- 每次都要看一次distance[ ], 共$|V|$個

Bellman-Ford algorithm

- 另外一種找到shortest path的algorithm
- 可以handle path cost是負的情形



- 為什麼Dijkastra 有問題?
- 如果下一個選擇的vertex為u, 為什麼v_0到u的最短path會正好是前面的vertex都在S中, 只有u不在S中呢?
- 因為假如有path中有vertex w也不在S中, 那麼表示path中會有一段是v_0到w, 而且長度比較短 (因為是sub path)
- 可是這樣的話, 應該w之前應該就要被選過了 (應該要在S中) (contradiction)
- 所以v_0到u的最短path前面的vertex都在S中, 只有u不在S中

假設cost可以是負的, 這就不一定正確了.

Bellman-Ford algorithm

- 定義 $dist^l[u]$為從v_0到u的最短路徑, 最多經過l條edge

- 一開始$dist^1[u]$為cost(<v_0,u>) (如果沒有edge則為無限大)
- 由$dist^{k-1}[\ ]$算$dist^k[\ ]$
- 最後算到$dist^{n-1}[u]$為最後shortest path解
  - (因為每個node最多走一遍, 不然就有cycle了)

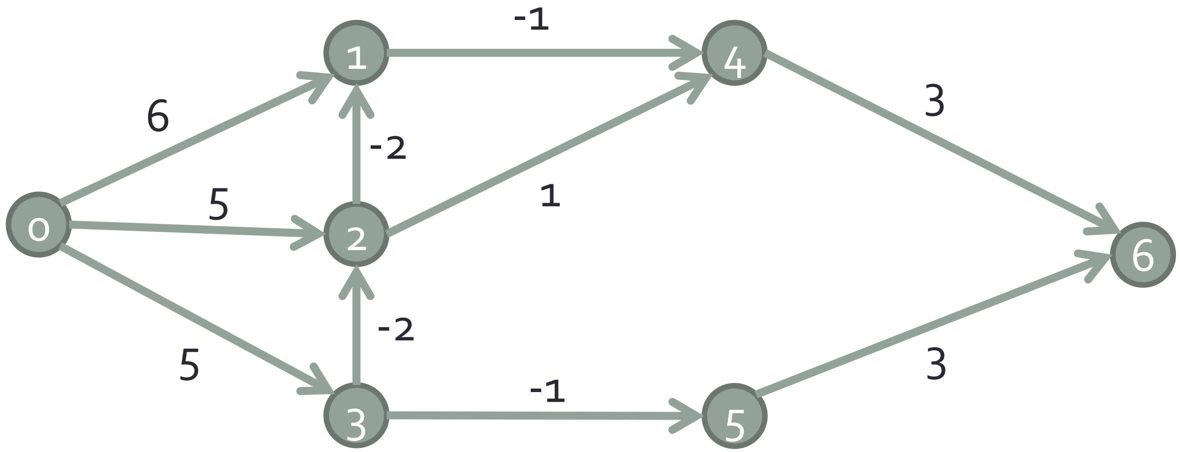
Bellman-Ford algorithm

- 由$dist^{k-1}[\,]$算$dist^{k}[\,]$
- 怎麼算呢?

- 有以下兩種可能性:
- 如果從v_0到u使用最多k個edge的最短路徑其實只經過了k-1或更少edge的話, 則$dist^{k}[u] = dist^{k-1}[u]$
- 如果從v_0到u使用最多k個edge的最短路徑使用了k條edge的話, 則最短路徑可能為經過別的vertex i的shortest path (用了最多k-1條edges)後再走edge <i,u>.

- 綜合以上兩條規則:
- $dist^{k}[u] = \min\{dist^{k-1}[u], \min_{i}\{dist^{k-1}[i] + cost(\langle i,u \rangle)\}\}$

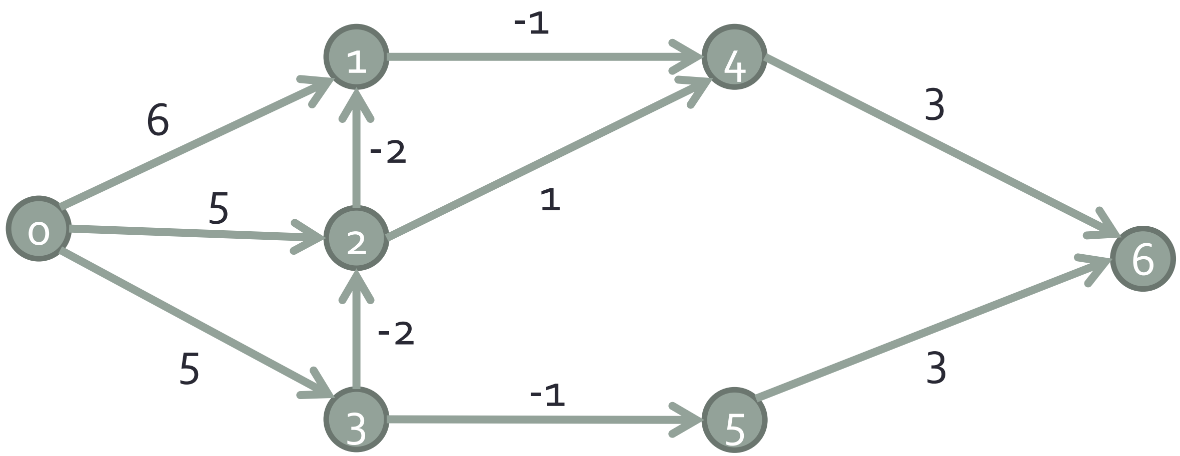
例子



vertex 0到其他vertices的最短路徑:

k	1	2	3	4	5	6
1	6	5	5	∞	∞	∞
2						
3						
4						
5						
6						

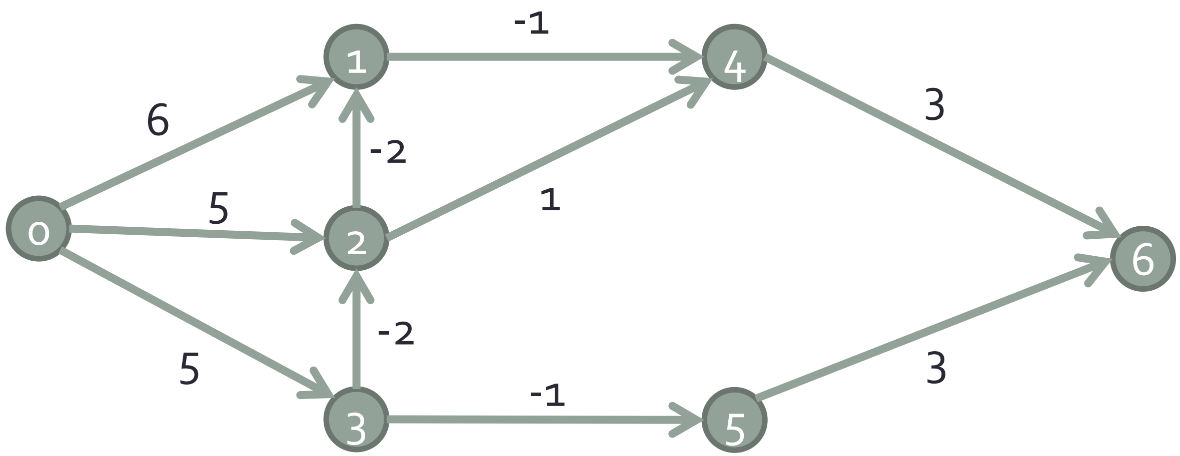
例子



vertex 0到其他vertices的最短路徑:

k	1	2	3	4	5	6
1	6	5	5	∞	∞	∞
2	3	3	5	5	4	∞
3						
4						
5						
6						

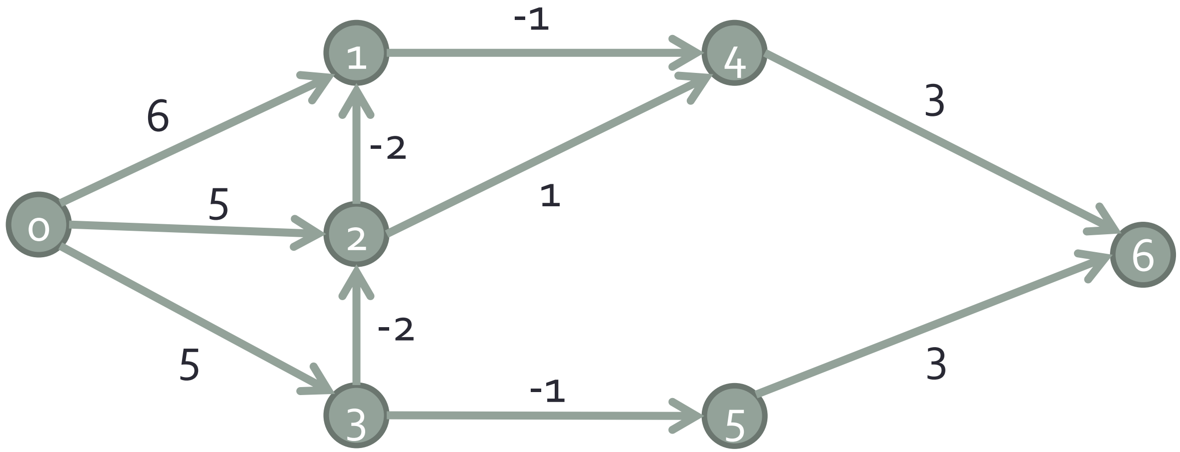
例子



vertex 0到其他vertices的最短路徑:

k	1	2	3	4	5	6
1	6	5	5	∞	∞	∞
2	3	3	5	5	4	∞
3	1	3	5	2	4	7
4						
5						
6						

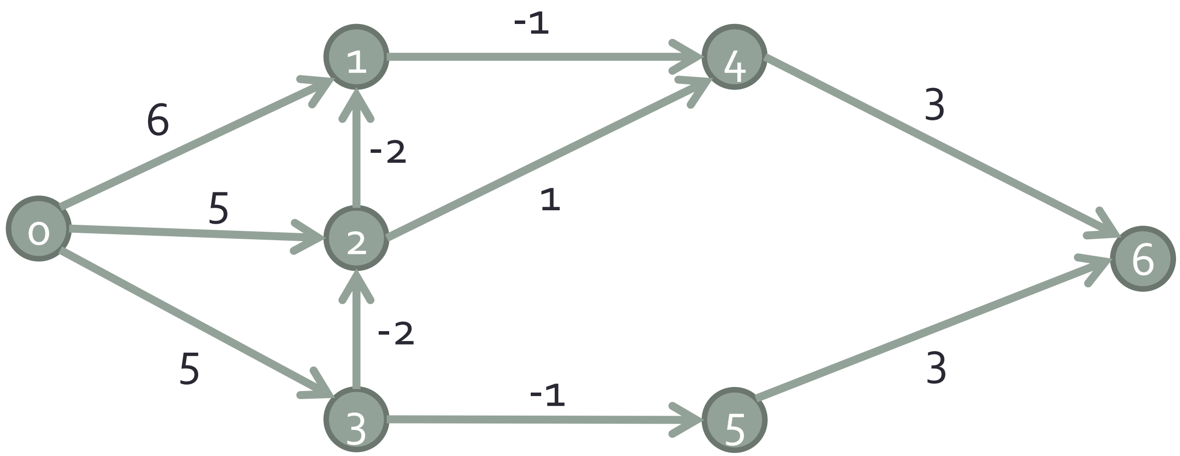
例子



vertex 0到其他vertices的最短路徑:

k	1	2	3	4	5	6
1	6	5	5	∞	∞	∞
2	3	3	5	5	4	∞
3	1	3	5	2	4	7
4	1	3	5	0	4	5
5						
6						

例子



k	1	2	3	4	5	6
1	6	5	5	∞	∞	∞
2	3	3	5	5	4	∞
3	1	3	5	2	4	7
4	1	3	5	0	4	5
5	1	3	5	0	4	3
6	1	3	5	0	4	3

其他例子: http://www.ibiblio.org/links/applets/appindex/graphtheory.html

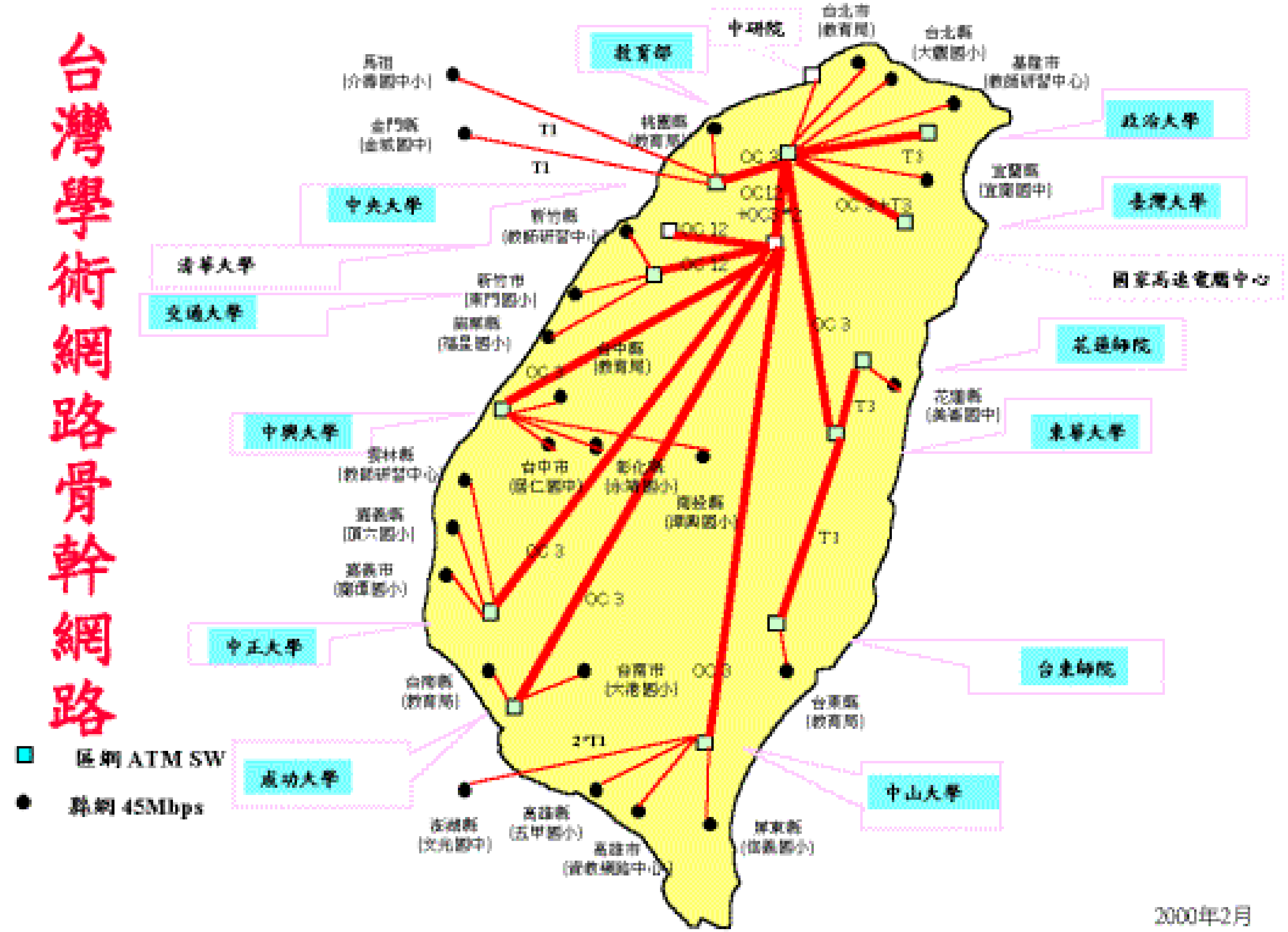
一些可以思考的東西

- time complexity = ?
- 答案:
- adjacency matrix: $O(|V|^3)$
- adjacency lists: $O(|V||E|)$

- 前面都沒有提到怎麼真的輸出path本身經過哪些vertex
- 想想看, 要怎麼做?

答案: 每一次update distance的時候, 順便update附在旁邊的linked list. (用linked list表示path)

TANET


台灣學術網路骨幹網路
教育部
中研院
台北市
台北縣
基隆市
政治大學
宜蘭縣
臺灣大學
國家高速電腦中心
中央大學
清華大學
交通大學
新竹縣
新竹市
苗栗縣
台中縣
花蓮師院
花蓮縣
東華大學
中興大學
雲林縣
台中市
彰化縣
南投縣
嘉義縣
嘉義市
中正大學
台南縣
台南市
台東縣
台東師院
成功大學
中山大學
澎湖縣
高雄縣
高雄市
屏東縣
馬祖
金門縣
桃園縣
區網 ATM SW
縣網 45Mbps
T1
T3
OC 3
OC 12
2*T1
2000年2月

Routing protocol

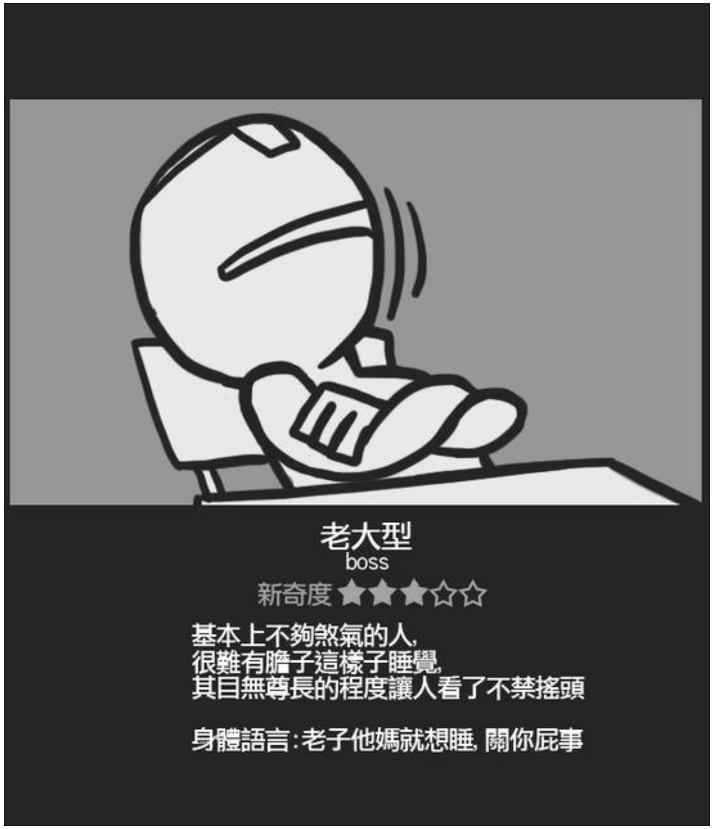
- 要計算從某個點到另外一個點的路徑(route)
- 目標: 連線速度?
- path cost =?
- Dijkstra比較適合還是Bellman-ford比較適合?

答案: 似乎Bellman-Ford比較適合, 因為不需要知道所有link cost即可建立routing table

Reading Assignments

- Cormen Ch 23.2, 24

下課時間



老大型
boss
新奇度★★★☆☆
基本上不夠煞氣的人,
很難有膽子這樣子睡覺,
其目無尊長的程度讓人看了不禁搖頭
身體語言:老子他媽就想睡, 關你屁事