

主題: Single-source Shortest Paths

- 基礎
- 應用
- 作業與自我挑戰

1

基礎

- The Single-source shortest paths problem
- Dijkstra's algorithm

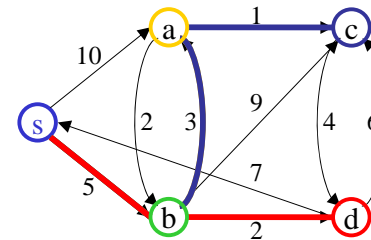
2

The single-source shortest paths problem

- 給一個 **directed weighted graph** 以及一個起始點 **s**，求 **s** 到其它每一個 vertex 的 **shortest path**
- 若 graph 為 **undirected**，可以把每條 edge 都換成兩條有方向的 directed edges
- 若 graph 為 **unweighted**，則可以視為每一條 edge 的 weight 皆為 1

3

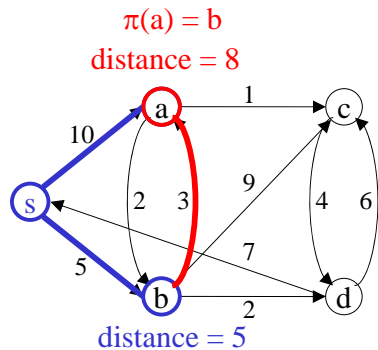
Example



- s 到 a: $s \rightarrow b \rightarrow a$
distance = 8
- s 到 b: $s \rightarrow b$
distance = 5
- s 到 c: $s \rightarrow b \rightarrow a \rightarrow c$
distance = 9
- s 到 d: $s \rightarrow b \rightarrow d$
distance = 7

4

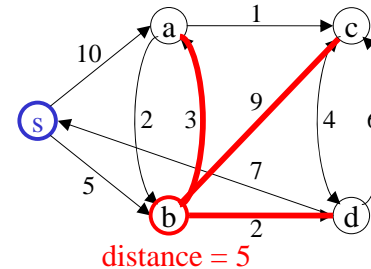
Relaxation



- 已知
 - s 到 a 有 length = 10 的 path
 - s 到 b 有 length = 5 的 path
 - b 到 a 有 weight = 3 的 edge
- 若先到 b 再到 a，是否比原來到 a 的 path 更短？
 - Yes, $5 + 3 = 8 < 10$
 - 用 8 取代 10

5

Relaxation (cont.)



- relax(u, v): 試著用 u 目前最好的答案去修改 neighbor v 目前最好的答案
 - 比較 $\text{distance}[u] + \text{weight}(u, v)$ 和 $\text{distance}[v]$ 的大小
 - 如果比 v 原本的 distance 要短，則更新 v 的 distance

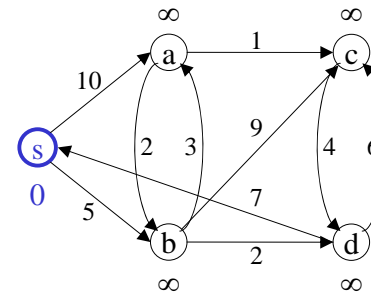
6

Dijkstra's algorithm

- 注意: 此方法只適用於 edge-weights 皆不為負數的情況
 - negative edges 需要其它作法, 較少用
- 用 set S 代表已經算出正確答案的 vertices
- 用 set Q 代表還沒有算出正確答案的 vertices
 - Initially, $S = \emptyset$ and $Q = V$
- 每回合從 S 之外 (也就是 set Q 中) 挑離 s 最近的 vertex u 加入 S, 並用此 vertex u 對其 neighbors 作 relax

7

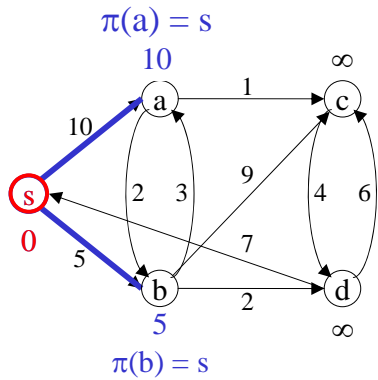
Initialization



- s 一開始的 distance 為 0
- 其他 vertices 的 distance 皆設為 ∞
- $S = \emptyset$; $Q = V$ (所有 vertices 都在 Q 中)

8

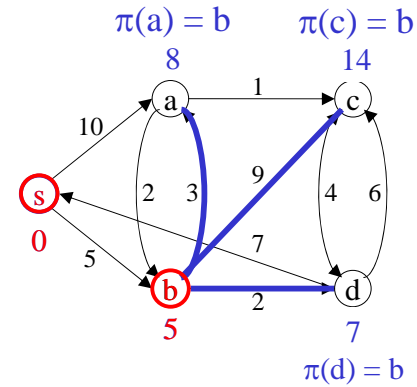
Phase 1



- S 以外 (Q 中) distance 最小的 vertex 是 s
- 將 s 放進 S
- 用 s 對其 neighbors 作 relax

9

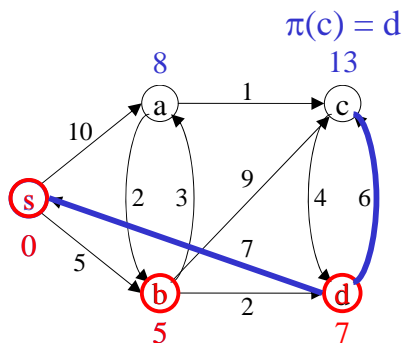
Phase 2



- S 以外 distance 最小的 vertex 是 b
- 將 b 放進 S
- 用 b 對其 neighbors 作 relax

10

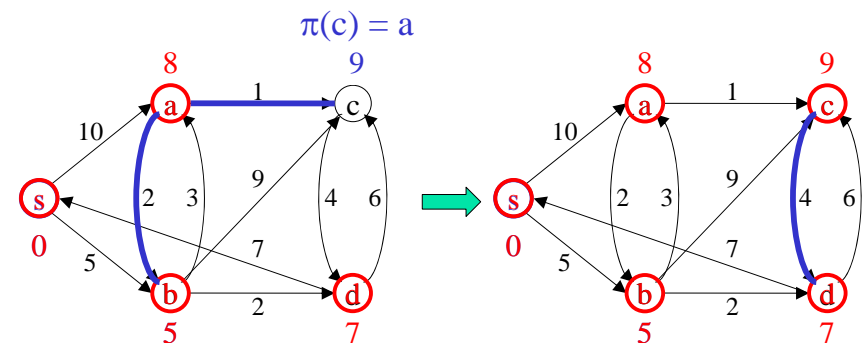
Phase 3



- S 以外 distance 最小的 vertex 是 d
- 將 d 放進 S
- 用 d 對其 neighbors 作 relax

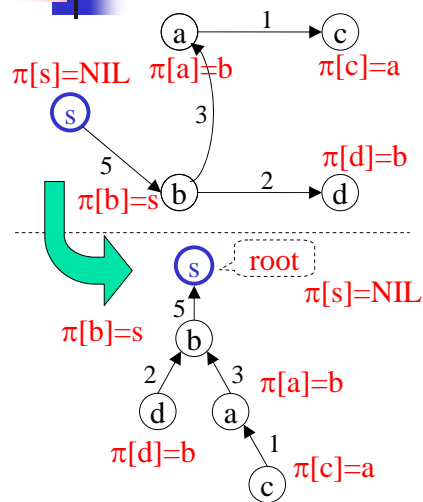
11

Phase 4, 5



12

Shortest path tree



- 若 v 最後是被 u relax 的，表示 v 的 shortest path 是由 u 走過來的
- π : $\pi[i]$ 記錄 s - i shortest path 上， i 的前一個 vertex
- 所有 π -link 合起來會稱為 shortest path tree
- 循著 π -link backtracking 可以找出每個人的 shortest path

13

Variables

- W : 用 adjacency matrix 來存 edges, $W[i, j]$ 存 edge (i, j) 的 edge length (∞ 表示 no edge)
- Q : $Q[i]$ 記錄 vertex i 是否屬於 set Q
 - $1: i \in Q$ $0: i \in S$
- d : $d[i]$ 記錄 vertex i 和 s 之間目前最好的答案
- π : $\pi[i]$ 記錄 s - i shortest path 上， i 的前一個 vertex
- 注意 ∞ 的處理

14

Pseudo code

```

initialize(s) {
  for (i = 0; i < n; i++) {  $Q[i]=1$ ;  $d[i] = \infty$ ;  $\pi[i]=NIL$ ;}
   $d[s]=0$ ;
}

relax(u, v) {
  if ( $d[v] > d[u] + W(u, v)$ ) {
     $d[v] = d[u] + W(u, v)$ ;
     $\pi[v] = u$ ;
  }
}

```

$\in Q$
 接受修正
 for backtracking

15

Pseudo code (cont.)

```

Dijkstra(s)
{
  initialize(s);
  for (i=0; i<n; i++)
  {
    find the  $u \notin S$  having minimum  $d[u]$ ;
     $Q[u] = 0$ ;
    for each  $v \in \text{adj}(u)$  do relax(u, v);
  }
}

```

a simple scan: $O(n)$
 move into S
 adjacency matrix: $O(n)$

16

Analysis

- 需要做 n 回合，每回合要做
 - 找到最近的 vertex $u \in Q$: $O(n)$
 - scan Q 找出在 Q 內且 distance 最小的 vertex
 - 將 u 搬入 S ，並 relax u 的 neighbors: $O(n)$
 - scan u 在 adjacency matrix 中的那個 row
- 總共需要 $O(n^2)$ 的時間
- **Remark:** 有 $O(m + n \lg n)$ 的 implementation
 - adjacency-lists + heap (course of Algorithms)
 - 程式較不容易寫

17

應用

- 應用一: A.318 Domino Effect
- 應用二: AT2003E Cave Raider
- 應用三: H.89.3 大眾運輸系統

18

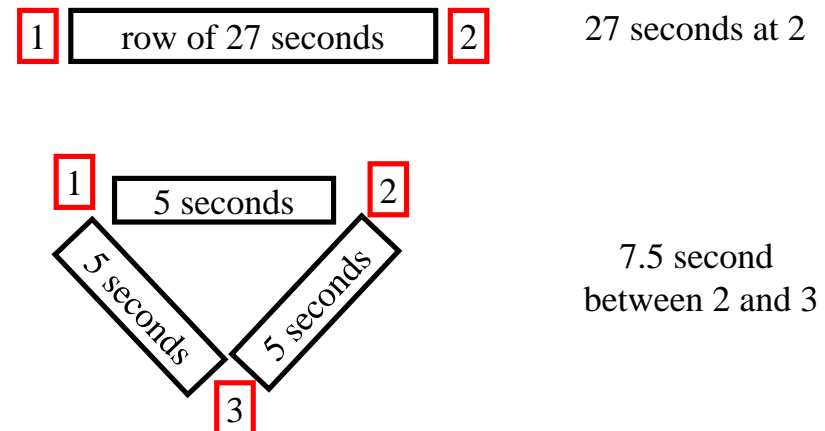
應用一: A.318 Domino Effect

<http://uva.onlinejudge.org/external/3/318.html>

- 給一堆骨牌的排列
 - 其中有 $n < 500$ 個骨牌稱為 key dominoes (編號 1, 2, ..., n)，key dominoes 之間由 simple dominoes 構成的 row 連接
 - key domino 周圍若有一個 simple domino 倒下，key domino 就會倒下並開始推倒周圍所有的 rows
- 推動 key domino 1 後，請問最後一張骨牌倒下的時間和位置

19

Examples



20

Solution

- Step 1. 計算每一個 key domino i 倒下的時間 $t(i)$
 - single-source shortest paths
- Step 2. 計算每一個 row (x, y) 完全倒下的時間 $r(x, y)$
 - 利用 $t(x)$ 和 $t(y)$ 計算
- Step 3. 找 $t(i)$ 和 $r(x, y)$ 中的 maximum
- $O(n^2)$ (用 $O(n^3)$ all-pairs algorithm 不會過)

21

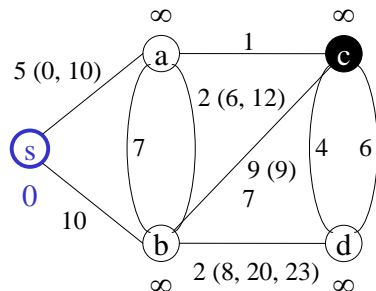
應用二: AT2003E Cave Raider

- 給一個地圖 ($n \leq 50, m \leq 500$)
 - node 代表山洞
 - edge 代表兩山洞間的 tunnel，有 weight 代表通過時間
 - 兩山洞間會有 multiple edges
 - 每根 edge 有關閉與開放時間
 - (6, 7, 8, 9): 表 [0~6) 開放, [6~7) 關閉, [7~8) 開放, [8~9) 關閉, [9~ ∞) 開放
- 給軍人和恐怖份子在山洞中的位置，找出最快抓到恐怖份子的方法

22

Solution: Dijkstra's algorithm

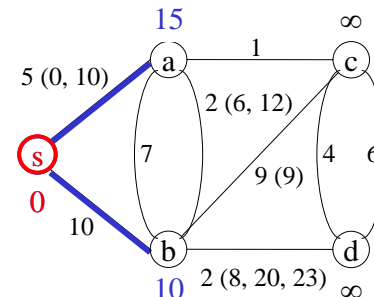
Initialization



- s 一開始的 distance 為 0
- 其他 vertices 的 distance 皆設為 ∞
- S 為空集合

23

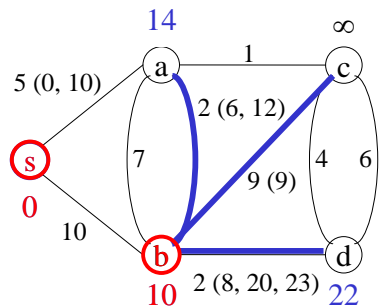
Phase 1



- S 以外 distance 最小的 vertex 就是 s
- 將 s 放進 S
- 用 s 對其 neighbors 作 relax
 - 注意考慮關閉時間

24

Phase 2



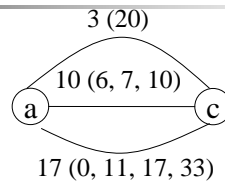
- S 以外 distance 最小的 vertex 是 b
- 將 b 放進 S
- 用 b 對其 neighbors 作 relax

25

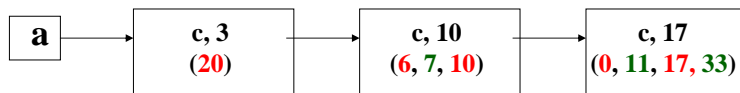
For many cases, Dijkstra's algorithm can handle **dynamic** edge lengths!

26

How to handle multiple edges?

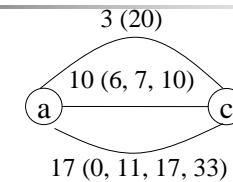


- Method 1. use adjacency lists

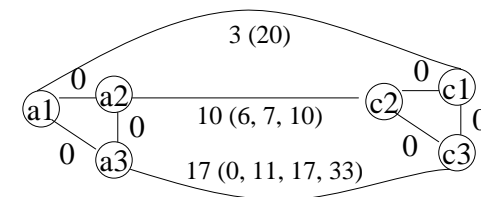


27

How to handle multiple edges?



- Method 2. **node duplication** (A very important technique!)
(making adjacency matrix applicable)



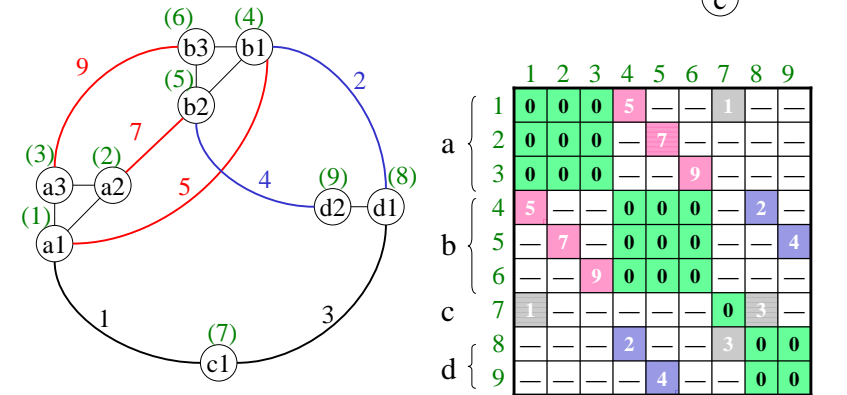
28

How to do duplication?

- Let k_i be the number of copies of node i
- Method 1. **Re-label all nodes**
 - hard to implement
- Method 2. **Two-dimensional identifications**
 - Let $k = \max\{k_i\}$. Duplicate every node into k copies.
 - The nodes are $(1, 1), (1, 2), \dots, (1, k), (2, 1), (2, 2), \dots, (2, k), \dots, (n, 1), (n, 2), \dots, (n, k)$
 - need more space and time

29

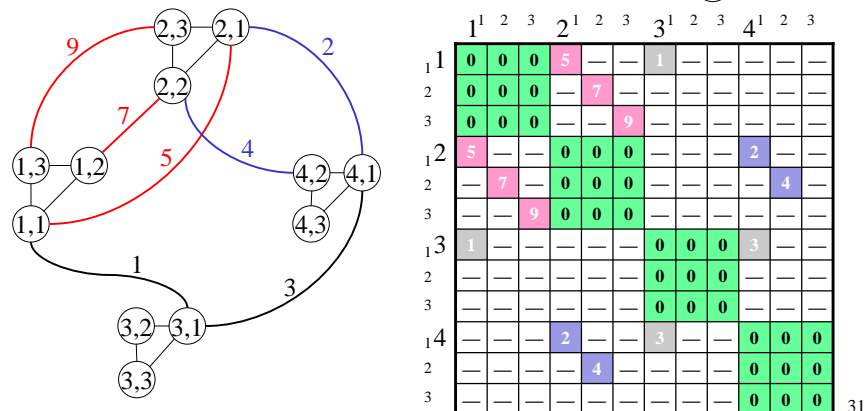
Example: Method 1



30

Example: Method 2

- $k = 3$: duplicate every node into 3 copies



31



應用三: H.89.3大眾運輸系統

(http://www.cc.nccu.edu.tw/info_race89/doc/final_program.doc)

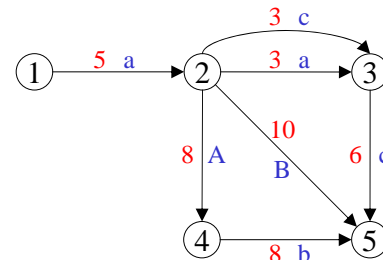
- 某城市中有數條公車及捷運路線，不同路線之間可能有交會點以供轉乘，轉乘公車需要 5 分鐘，轉乘捷運需要 10 分鐘 (就算公車轉公車或捷運轉捷運也一樣)
- 給予每條路線上車站與車站間的行車時間，求出從起點 1 號站到其他所有車站費時最少的乘車方式

32

Sample input

a  路線代號 (小寫為捷運)
 1 5 2 3 3 0
 b
 4 8 5 0
 c
 2 3 3 6 5 0
 A
 2 8 4 0
 B  路線代號 (大寫為公車)

需 0 分鐘
為終點站



33

Sample output

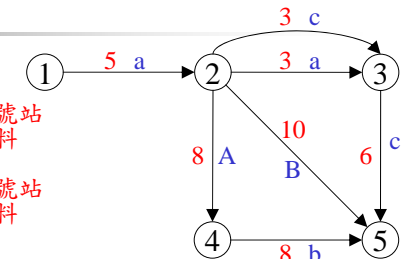
1=>2 (0 transfer, 5 min.)
 1->2 (a, 5 min.)
 1=>3 (0 transfer, 8 min.)
 1->2 (a, 5 min.)
 2->3 (a, 3 min.)
 1=>4 (1 transfer, 18 min.)
 1->2 (a, 5 min.)
 a->A (5 min.)
 2->4 (A, 8 min.)

到 2 號站
的資料

到 3 號站
的資料

到 4 號站
的資料

包括轉乘次數，花費時間，以及經過的路線和轉乘路線



...

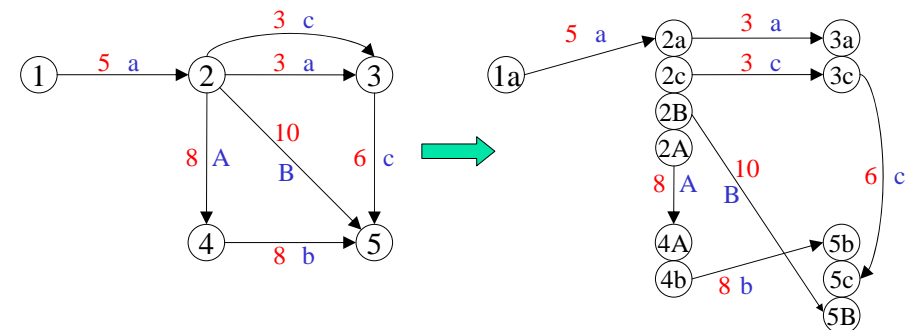
34

Solution

- 用 shortest path 去做，困難如下：
 - 任兩點間可能有一條以上的 edges，無法用 adjacency matrix 存放
 - 在同一點上需要轉乘，但是一個 vertex 無法表現出轉乘所需的時間
- Solution: 把現在的路線圖轉換成可以直接進行 shortest path 的 graph

35

- 當一個 vertex v 被好幾條路線經過時，把每條路線上的 v 都當成一個獨立的 vertex
node duplication !

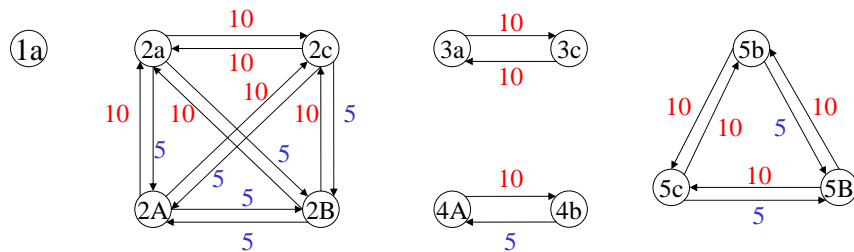


36

- 因為一個 vertex 被拆成好幾部分，所以可以把在同一站轉乘所花的時間放在各部分之間

到小寫字母
⇒ 轉捷運花 10 分鐘

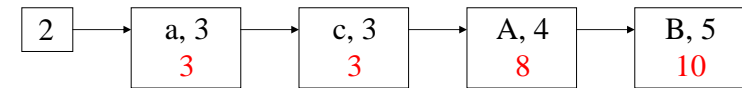
到大寫字母
⇒ 轉公車花 5 分鐘



37

Two Implementations

- Method 1. 直接造 G' (node duplication) 的 adjacency lists or matrix
- Method 2. Modified Dijkstra's Algorithm (Two-level relaxation)

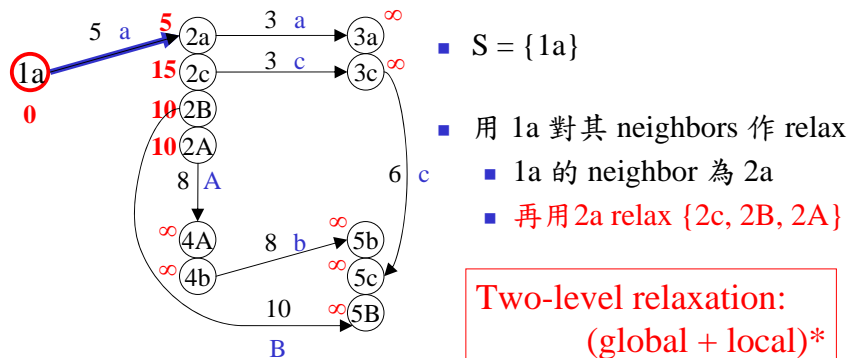


38

Method 2: Step 1

(Before) Queue : (1a, 0)

(After) Queue : (2a, 5), (2B, 10), (2A, 10), (2c, 15)

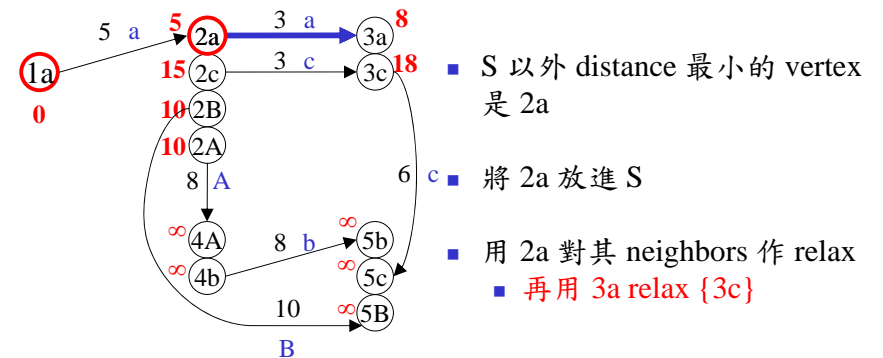


39

Step 2

(Before) Queue : (2a, 5), (2B, 10), (2A, 10), (2c, 15)

(After) Queue : (3a, 8), (2B, 10), (2A, 10), (2c, 15), (3c, 18)

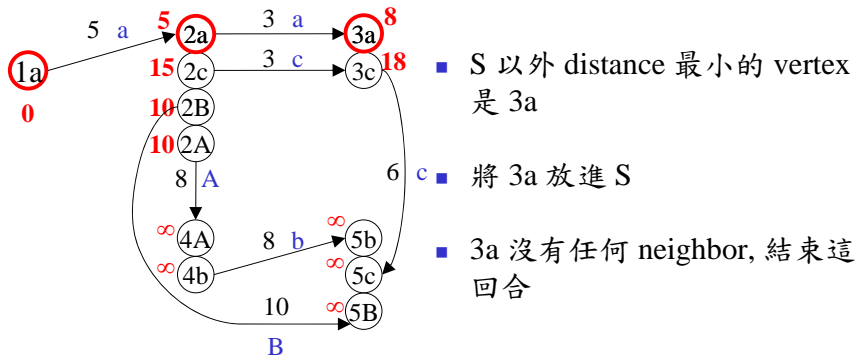


40

Step 3

(Before) Queue : (3a, 8), (2B, 10), (2A, 10), (2c, 15), (3c, 18)

(After) Queue : (2B, 10), (2A, 10), (2c, 15), (3c, 18)

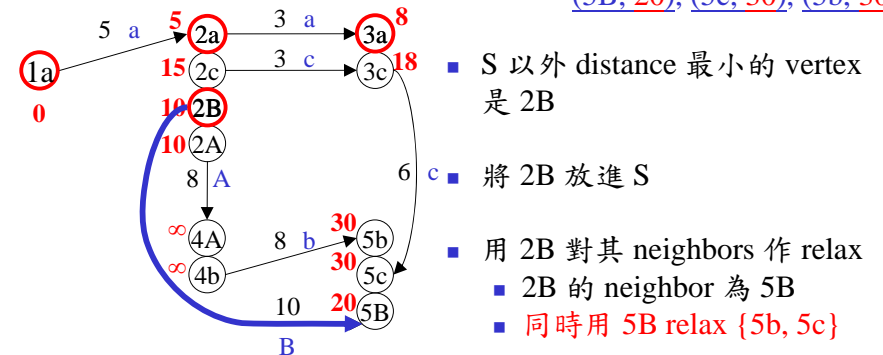


41

Step 4

(Before) Queue : (2B, 10), (2A, 10), (2c, 15), (3c, 18)

(After) Queue : (2A, 10), (2c, 15), (3c, 18),
(5B, 20), (5c, 30), (5b, 30)



42

Summary

- dynamic edge-lengths: Dijkstra's algorithm
- multiple edges: (i) lists (ii) node duplication \Rightarrow matrix
- node duplication - applications
 - multiple edges
 - different meanings of a vertex
- node duplication - implementations
 - constructing a new graph G'
 - embedding duplication in an algorithm

43

作業與自我挑戰

- 作業
 - 練習題
 - A.318 Domino Effect
<http://uva.onlinejudge.org/external/3/318.html>
 - 挑戰題
 - A.10537 Toll! Revisited (2003 ACM final Problem J)
<http://uva.onlinejudge.org/external/105/10537.html>
- 自我挑戰
 - A.10246 Asterix and Obelix
<http://uva.onlinejudge.org/external/102/10246.html>
 - AF2002B Undecodable codes
- 其它有趣題目:
 - A.248 Cutting corners
 - AT2002G Packet Radio Routing Problem
 - AF2233 Cheese
 - AT2002B The Longest Detour Problem

44