Shortest Path in class

by 林品諺

Sprout

Floyd-Warshall

- 感覺對對的，但總有種不踏實感
- 為什麼說這個算法是某種DP呢？

Floyd-Warshall

- 感覺對對的，但總有種不踏實感
- 為什麼說這個算法是某種DP呢？

我大DP

- 原本的狀態：
- d[i][j]:從點 i 走到點 j 的最短距離
- for k = 1..n
- for i = 1..n
- for j = 1..n
- d[i][j] = min(d[i][j], d[i][k]+d[k][j]);

我大DP

- 改個狀態：
- d[k][i][j]:從點 i 走到點 j 的最短距離，且除了 i 跟 j 以外，只能經過前 k 個點
- for k = 1..n
- for i = 1..n
- for j = 1..n
- d[k+1][i][j] = min(d[k][i][j], d[k][i][k] +d[k][k][j]);

Dijkstra

ALGORITHM (MAIN PART)

- Repeat the following procedure until U is empty
- For all vertices adjacent to p, let it be q, update $d[q]$ with $d[p] + \text{weight}(p, q)$
- Remove p from U.
- For all vertices in U, find the vertice with smallest $d[]$, and assign it to p.

Dijkstra

ALGORITHM (MAIN PART)

- Repeat the following procedure until U is empty
- For all vertices adjacent to p, let it be q, update $d[q]$ with $d[p] + \text{weight}(p, q)$
- Remove p from U.
- For all vertices in U, find the vertice with smallest $d[]$, and assign it to p.

Dijkstra

- 從一個集合裡面找出最小的數字
- priority_queue(heap)!!
- 用priority_queue找到d[]最小的點，然後用它來更新其他點的d[]
- 原本的做法：(V 輪) * ((O(V)找到d[]最小的點) + (更新連到它的點)) = $O(E+V^2)$
- 新做法：(V 輪) * ((O(logV)找到d[]最小的點) + (更新連到他的點)，**把更新後的點放進heap**) = O((V+E)logV)

Bellman-Ford

- 設d[i]是從起點走到點 i 的最短距離
- 根據定義：d[j] <= d[i]+e(i, j)，否則d[j]就不是最短距離
- 小性質：只要對於所有的(i, j)都滿足d[j] <= d[i]+e(i, j)，而且對每個 j 都存在d[j] = d[i]+e(i, j)，那麼d[]就是最短距離陣列
- 想法：如果找到一組(i, j)滿足d[j] > d[i]+e(i, j)，就把d[j]更新成d[i]+e(i, j)
- 要怎麼更新？
- 亂更新！！

Bellman-Ford

算法

- 初始化所有 $\mathrm{dis}(u)$ 為無窮大（起點設為 0）
- 對圖上的每一條邊放鬆一次
- 重複上面的步驟 $V-1$ 次

Bellman-Ford

- 每次都跑完整個盤面怎麼感覺笨笨的
- 如果圖很大的話，每次就花一堆時間在更新後面的inf們
- 可不可以每次只跑那些有被更新到的點？
- SPFA(Shortest Path Faster Algorithm)!!

SPFA

- 1. 一開始把起點push到queue裡面
- 2. 從queue裡面pop出一個點 u
- 3. 用 u 去放鬆(更新)其他點
- 4. 把所有被更新到的點都push到queue裡面
- 5. 如果queue已經空了，那麼算法結束。否則回到2.

SPFA

- 這個算法感覺聰明多了！
- 我們來看看它的時間複雜度進步了多少吧！
- O(VE)
- 足夠慘的時候，還是會跑到VE的
- 不過有些人說這東西期望上大概是O(kE)，k大概是 2
- 實務上其實挺快的，競賽的話就看出題者有沒有特別卡(其實不太好卡)

怎麼卡SPFA

- SPFA感覺快快的，該怎麼卡呢？
- 生一個V=sqrt(E)的完全圖
- for i = 1..(n-1)
- 　e[i][i+1] = 1
- for i = 1..(n-1)
- 　for j = 1..n \ {i+1}
- 　　e[i][j] = 10*n-2*i
- 這樣可以讓SPFA跑成大概E*sqrt(E)
- 不知道有沒有別的卡法，而且好像也不一定卡得掉

小總結

- Bellman-Ford：亂更新一通
- SPFA：好好寫的Bellman-Ford
- Dijkstra：如果沒有負邊的話，利用這個好性質來更好的更新其他點

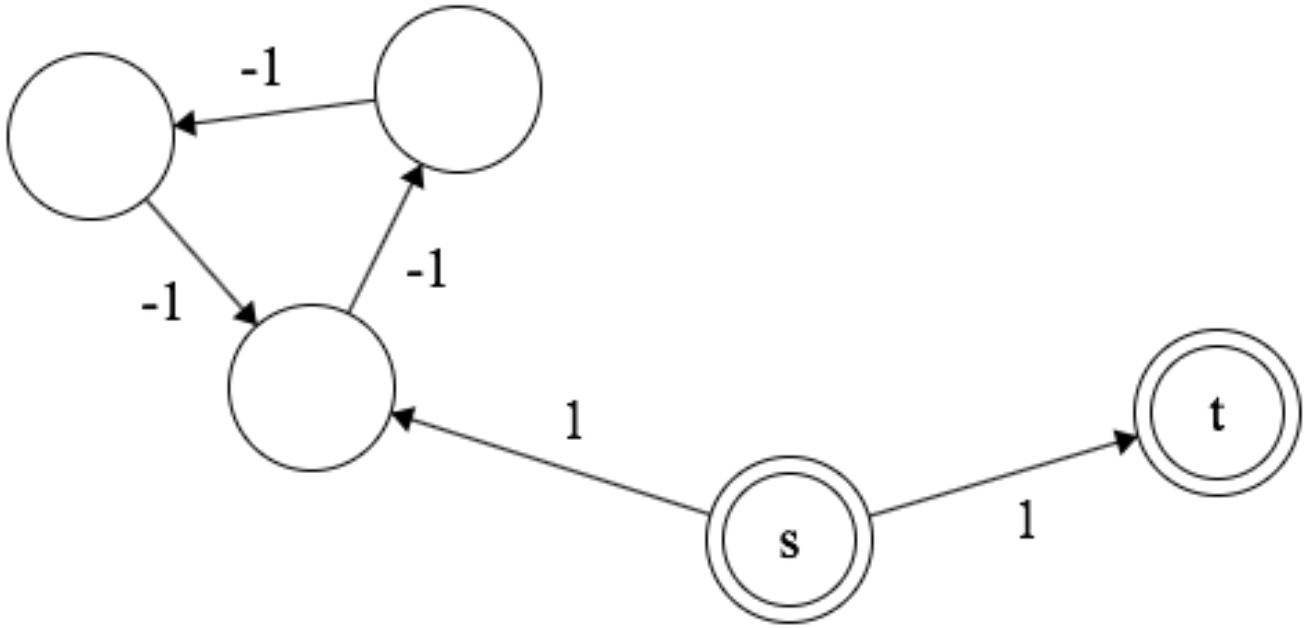
各種比較

算法	Floyd-Warshall	Bellman-Ford	SPFA	Dijkstra
時間複雜度	$O(V^3)$	O(VE)	O(VE) (期望O(kE))	$O(V^2)$ or O(ElogV)
處理負邊	O	O	O	X
處理負環	X	O	O	X
特點	好寫	沒用	可以處理負環	最快

負環

- 什麼叫做處理負環呢？
- 不是說好有負環的話就沒有最短路嗎？
- 可以拿來看有沒有負環
- 例如更新的次數太多
- 但沒負環一定沒有解嗎？

負環



負環

- 前面提到的可以處理負環的演算法都可以處理上一頁的那種情況
- 那要怎麼分辨一個有負環的圖存不存在一個s-t最短路呢？
- 更新太多次 ➔ 有負環
- 如果有負環的話，從終點回朔回去，看有沒有經過重複的點
- 如果走到了重複的點，那就表示有負環
- 順帶一提，如果只是要判斷有沒有負環的話，Floyd-Warshall也可以判斷有沒有負環，而且只要看一下有沒有d[i][i]<0就好

最短路徑樹

- 每一個最短路徑一定都是簡單路徑
- 如果起點到每個點的最短路徑都唯一的話，那把這些路徑疊起來會變成一棵樹
- 我們稱這種樹叫做最短路徑樹
- 如果最短路徑們不唯一的話，一樣會存在一顆最短路徑樹，只是那棵樹不唯一