Algorithms HW6 Handwritten

2020 Spring | 90899201Y tony20715 黃悟淳

Problem One

Fill the Blanks

Shortest Path Algorithm	Design Method	Apply on graph containing negative edge	Apply on graph containing negative cycle	Time Complexity	Auxiliary Space Complexity
Bellman-Ford	(a)dynamic programming	yes	yes	$O(\mid V \mid\mid E \mid)$	$O(\mid V \mid)$
Dijkstra (binary-heap)	(b)greedy	no	no	$O(\mid V \mid^2)$	$O(\mid V \mid)$
Floyd-Warshall	(a)dynamic programming	yes	yes	$O(\mid V \mid^3)$	$O(\mid V \mid^2)$

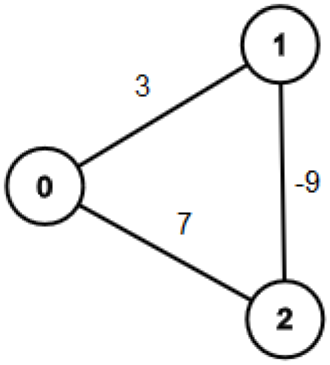
Problem Two

1. 若沒有負環，圖中任一最短路徑最多經過$G.V-1$個邊。Pseudo-code中第2-4行relax了$G.V-1$次，可視為從起點起最多經過$G.V-1$個邊的最短路徑。故若如第5-7行執行第$G.V$次的relax仍使最短路徑權值下降，代表這條路徑圖中必定經過負環。
2. 在第7-8行之間增加:
   {
   negcycle = FALSE
   for i = 1 to n
       if $d_{ii}^{(n)}$ < 0
           negcycle = TRUE
   }
   若negcycle為FALSE，代表無負環，反之若為TRUE，代表有負環。

3. 根據課本式(25.1)，若$i \rightarrow j$沒有邊，$w_{ij} = \infty$，此處使用相同符號。在第7-8行之間增加:
   {
   let $reachable$ be a new $n \times n$ matrix
   for $i = 1$ to n
       for $j = 1$ to n
           $reachable_{ij} = TRUE$
           if $d_{ij}^{(n)} = \infty$
               $reachable_{ij} = FALSE$
   }
   若$reachable_{ij}$為FALSE，代表$i \rightarrow j$沒有路徑，反之若為TRUE，代表有路徑。

Problem Three

1. Since there is no multiple edge,
   - if G is an undirected graph, the extreme condition that each vertex can reach all other vertices by a single path leads to a maximum amount of edges in graph G, which is $C_2^{|V|} = \frac{|V| \times (|V|-1)}{2} \in O(|V|^2)$. We can therefore infer that $max(|E_{undirected}|) \in O(|V|^2)$.
   - Substituting the previous result into the equation gives $O(|V| + |E|) = O(|V| + |V|^2) = O(|V|^2)$ and proves the assertion is correct.
   - if G is a directed graph, the previous result is still valid since the maximum possible edges by replacing each edge $u \leftrightarrow v$ in the undirected graph with two directed edges $u \rightarrow v$ and $v \rightarrow u$. This leads $O(|E_{directed}|)$ = $O(2 \times |E_{undirected}|) = O(|E_{undirected}|)$, same complexity as the undirected one. The remaining proof is identical to the undirected one.
2. Consider the following graph, taking node 0 as the source node:
   -



- Use Bellman-Ford algorithm and relax the edges in the order of $(0,1),(1,2),(0,2),(2,0),(2,1),(1,0)$. We can obtain the optimal result after two iterations of all-edges' relaxation:
  - after first iteration:

Node	**0**	**1**	**2**
Distance	0	-2	-6

  - after second iteration - done:

Node	**0**	**1**	**2**
Distance	0	-2	-6

- Using Dijkstra Algorithm, we obtain:

Node	**0**	**1**	**2**
Distance	0	3	-6

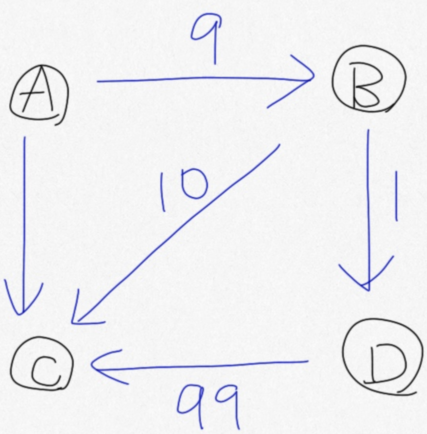
- The reason of the failure using Dijkstra to obtain the correct result is that just after denoting node 0 as a "finished node", we update the distance of node 1, 2 as 3 and 7, respectively. We picked a node 1 and mark it as a "finished node", which is a greedy choice, and assume that the distance of node 1 can never be changed. This choice forbids us to observe alternative paths for node 1, such as the shorter path $0 \to 2 \to 1$.

3. - 兩個結果會不一樣，因為Prim's是選能連上「已生成的樹」的邊之中，權重最小的邊；Dijkstra's則是要選：在所有與「最短路徑樹」只隔一條邊的節點中，距離原(origin)節點「累

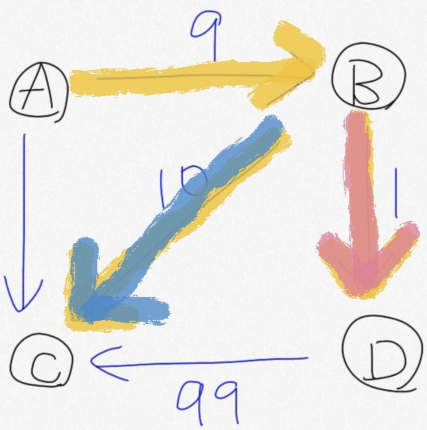
加權重最小的節點與現有最短路徑樹連接的邊」。下圖可以舉例，Prim's和Dijkstra's所產生的樹會不同，因為取的「最小邊」原則不同。該圖的起點為A。

-

1、原圖

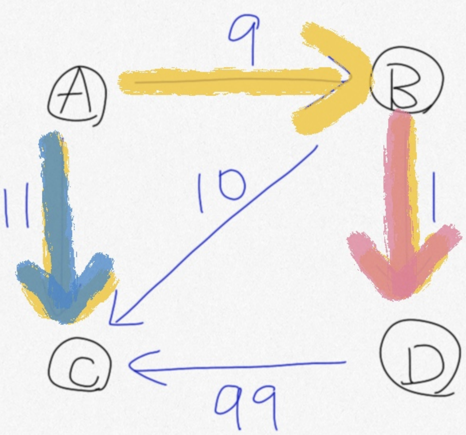


2、Prim's Algorithm



A	B	C	D
0	9	~~11~~	~~∞~~
		10	1
		~~99~~	

3. Dijkstra Algorithm



A	B	C	D
0	9	11	10
			~~19~~
		~~109~~	