

Homework 2: The weighted center of a tree 閱讀心得

R10942152 游家權

二、問題定義

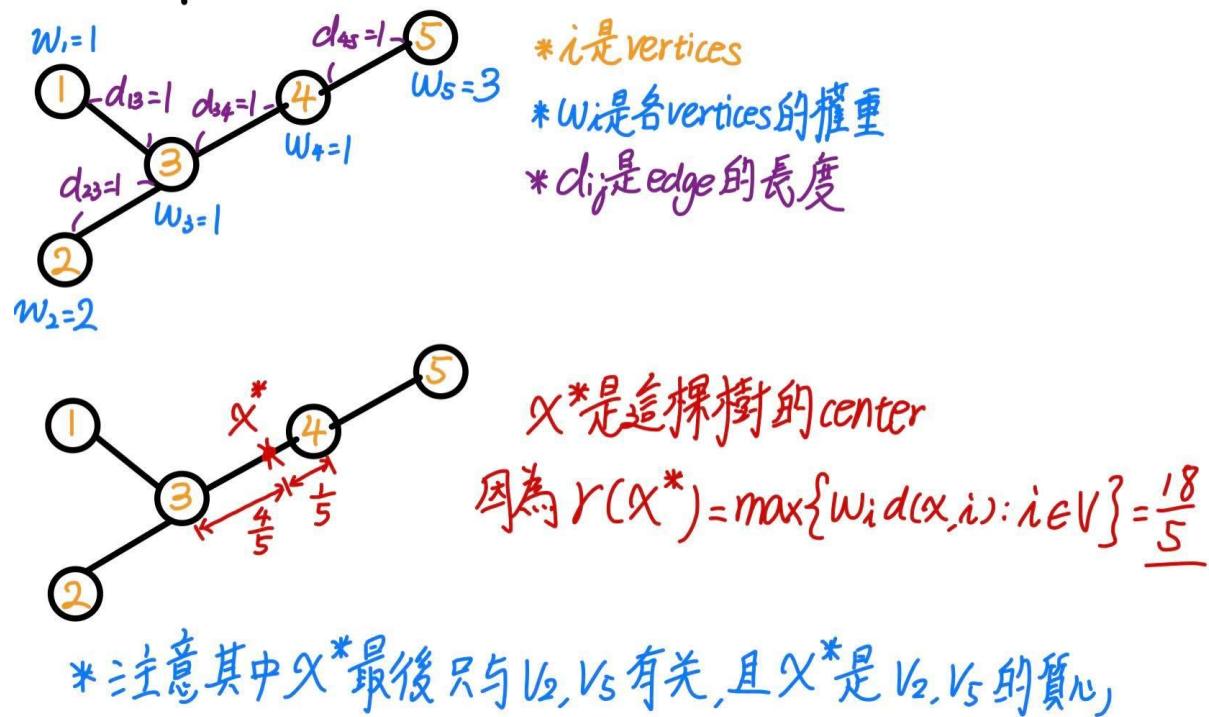
定義一棵樹，每個vertices i 都有權重 w_i ，每個edge (i, j) 都有定義的距離 d_{ij} 。而根據這個距離，我們可以在這棵樹上定義任兩點 x, y 之間的距離 $d(x, y)$ 。

這個問題的目標，就是要找出樹上一點 x^* 。整體來說， x^* 異離樹中大部分的其他vertices都蠻近的，所以稱他為center。 x^* 的特性是就算考慮離 x^* 最遠的vertex，它們之間的權重距離，跟任何其他點比是最遠的。嚴謹的數學定義如下：

$$r(x) = \max\{w_i d(x, i) : i \in V\}$$

而我們的目標就是找到某個 x^* ，使他的 $r(x)$ ，比任何其他 x 都來的小。

Example:



二、觀察問題

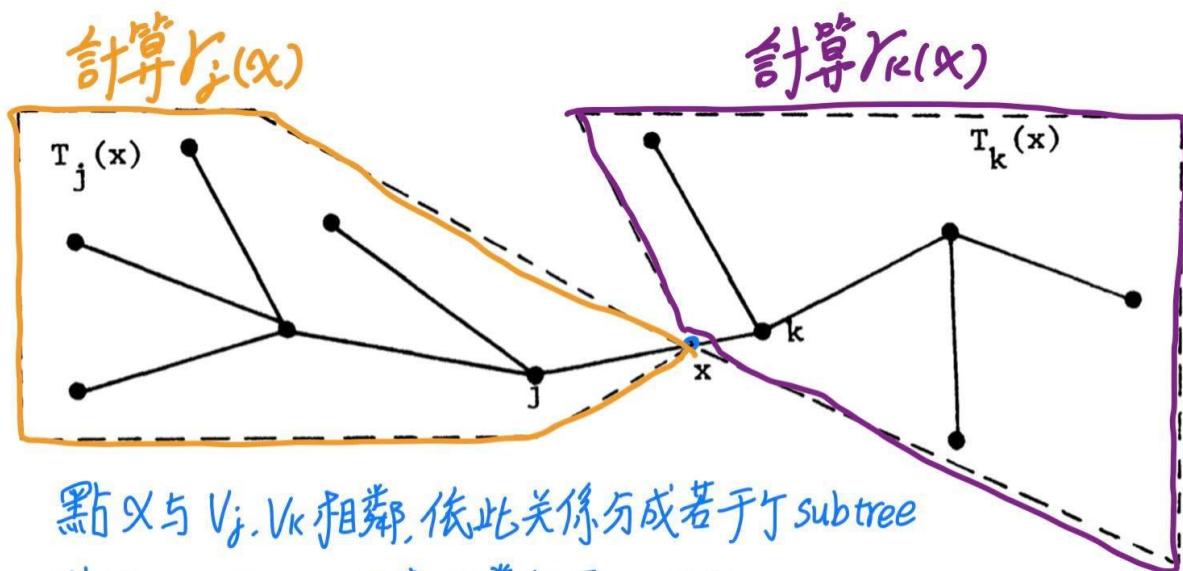
Observation 1：在兩棵subtree中，找出 x^* 在哪個subtree裡，只要花 $O(n)$

定義 $V_j(x)$ 為走到 X 會經過頂點 j 的所有頂點集合

定義 $T_j(x)$ 為 $V_j(x)$ 加上 X 點所形成的Subtree

考慮情形如下圖，可以將 X 的兩個adjacent vertices 分成兩個subtree，接下來我們分別考慮 $r_j(x), r_k(x)$ ，我們可以利用 $r_j(x)$ 與 $r_k(x)$ 的關係來找出 x^* 在哪個子樹裡面，而且這件事情只需要花費 $O(n)$ 的代價，這個性質可以讓我們知道 x^* 的大致方向，縮小搜索範圍。

這個Observation很直觀，因為這個問題想要縮小的是最遠距離，所以最遠距離的頂點在哪裡， x 就應該往那個方向去移動。同時也注意到如果 $x^* = x$ 則接下來不管 X 往哪個方向移動， $r(x)$ 都會變大，如同一個碗的底部。 $r(x)$ 是一個Convex function, local minimum就是global minimum，這也是這個問題能被Prune and Search 解決的其中一個原因。



點 x 与 V_j, V_k 相鄰，依此關係分成若干子樹

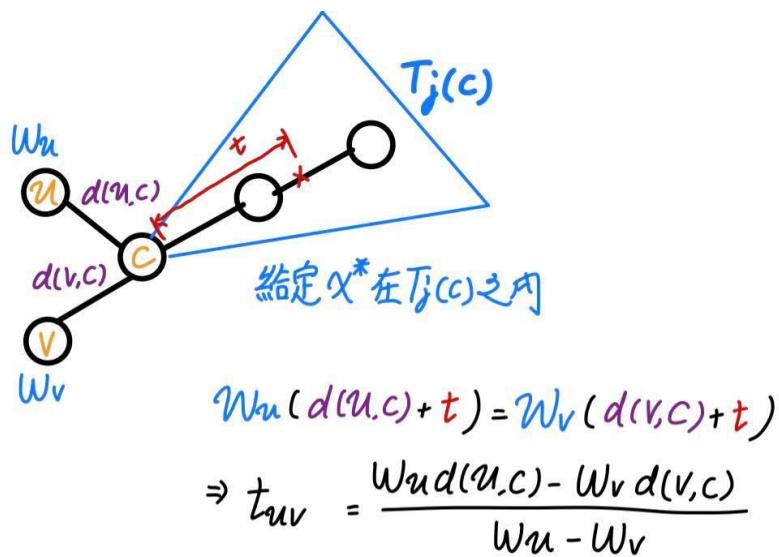
若 $r_j(x) > r_k(x)$, 代表 x^* 在 $T_j(x)$ 之中

若 $r_j(x) < r_k(x)$, 代表 x^* 在 $T_k(x)$ 之中

若 $r_j(x) = r_k(x)$, 代表 $x^* = x$

Observation 2: 如果有兩個頂點 u, v , 不屬於center的子樹 $T_j(x)$ 裡面。則我們可以花費 $O(n)$ 的時間, 把 u, v 其中一個點prune掉。

若兩個頂點 u, v , 不屬於center的子樹 $T_j(x)$ 裡面, 我們可以用圖中方程式計算這兩個頂點的權重距離跟 t_{uv} 之間的關係, 而這個 t_{uv} 就是一個距離臨界點。如果假設 x^* 到C的距離比 t_{uv} 小, 那方程式的左手邊(V_u)會比較大, 也就是說 V_u 會比 V_v 重要, V_u 更有可能在最後去拉扯center的位置, 因為它的權重距離比 V_v 大, 所以我們能安心地把 V_v prune掉。另外一種狀況是, 針對在 x^* 到C的距離比 t_{uv} 大, 方程式的右手邊(V_v)會比較大, 經由類似的邏輯推演, 一樣可以把 V_u prune掉。



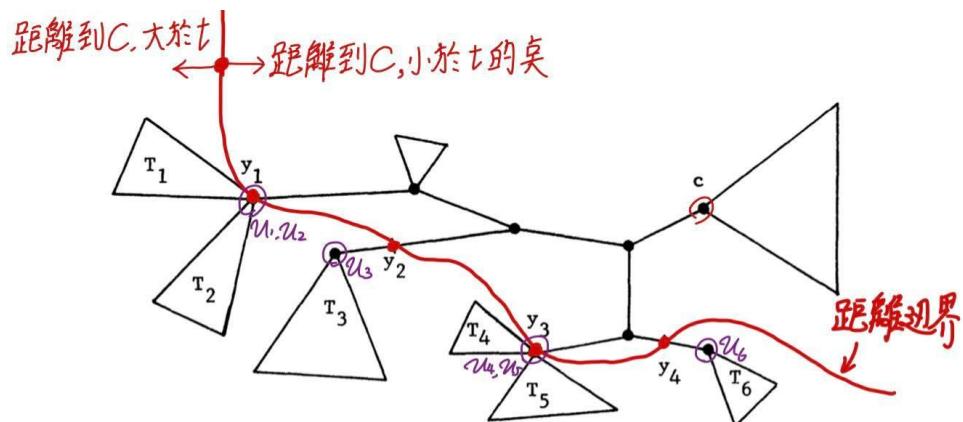
- Case 1: 當 $W_u d(u, C) \geq W_v d(v, C)$ 且 x^* 在範圍 t_{uv} 之內 prune 掉 v
- Case 2: 當 $W_u d(u, C) \geq W_v d(v, C)$ 且 x^* 在範圍 t_{uv} 之外 prune 掉 u
- Case 3: 當 $W_u d(u, C) \leq W_v d(v, C)$ 且 x^* 在範圍 t_{uv} 之內 prune 掉 u
- Case 4: 當 $W_u d(u, C) \leq W_v d(v, C)$ 且 x^* 在範圍 t_{uv} 之外 prune 掉 v

但要這麼做, 需要有能力判斷 x^* 在範圍 t_{uv} 之內還是之外, 故需要observation3。

Observation 3: 給定樹上一點 c 與距離 t , 我們可以用 $O(n)$ 的時間確定 x^* 跟 c 是否在距離 t 的範圍之內。

如下圖所示, 我們想知道 x^* 在紅色的距離邊界內, 還是在距離邊界外。所以我們去找距離邊界外面的這些Subtree, 並且分別計算這些Subtree內部的 $r(u_i)$ 。如果發現有其中一個Subtree的 $r(u_i)$ 很大, 代表 x^* 會被這棵Subtree拉走, x^* 必定掉在距離邊界外面。另外如果發現有兩棵以上的Subtree的 $r(u_i)$ 一樣是最大的, 那代表這兩棵Subtree用一樣的力道去拉扯 x^* , 使得 x^* 不會落在任何一棵Subtree之內, x^* 必定掉在距離邊界內。

以上的計算包含尋找 $y_1 \sim y_i$ 、以及計算各個Subtree內部的 $r(u_i)$, 這些工作皆能在 $O(n)$ 時間內完成。



- * $y_1 \dots y_4$ 是剛好距離到 C , 長度是 t 的矣
- *去計算每個subtree到root的 $r(u_i)$, 並且把所有 $r(u_i)$ 拿出來比較.

Case 1: 如果有任一個 $r(u_i)$ 比其他人都大, 那 x^* 在 t 距離外 (x^* 會被這棵 T_i 拉走)

Case 2: 如果有任2個 $r(u_i)$ 都比其他人都大, 那 x^* 在 t 距離內 (x^* 不可能在這些樹裡)

三、解法敘述

step1: 找樹的質心(centroid) c

step2: 用Observation 1的結果，將 c 相鄰的子樹分出來，並且找出知道 x^* 在哪個子樹裡，此時如果運氣好，也有可能發現 $x^* = c$

step3: 把所有不屬於 $T_j(c)$ 子樹裡的所有點，分成兩兩一組pair，一共會有約 $n/4$ 組

step4: 針對所有pair分別計算 t_{uv} ，如果 t_{uv} 是正值，按照Observation 2的結果把權重距離較小的頂點直接prune掉。但如果 t_{uv} 是負值，該pair要繼續做step5,6。

step5: 把所有未能成功Prune掉的Pair集合起來，找這些pair的 $|t_{uv}|$ 的中位數，是為 t_m 。

step6: 用Observation 3的結果，如果我們知道 x^* 落在 t_m 的範圍內，則針對那些 $|t_{uv}| >= t_m$ 的pair，我們可以把它權重距離較小的頂點刪掉。另外如果 x^* 落在 t_m 的範圍外，則針對那些 $|t_{uv}| < t_m$ 的pair，我們可以把它權重距離較大的頂點刪掉。

step7: 重複step1到step6，直到整個problem size被縮減到base case，最後將base case直接解出答案。

時間複雜度分析：

上述Step1~Step6，每一個步驟都只花費 $O(n)$ 的時間，而每次做完六個步驟，都會有 $n/8$ 個頂點被刪除，使得Problem size 變成 $7n/8$ 。可以寫出遞迴式。

$$T(n) = T(7n/8) + O(n)$$

解上述遞迴式可以得到，時間複雜度是 $O(n)$

四、讀後心得

我發現能用Prune and Search解的問題都有個共同特徵，就是解的Constraint其實很鬆。換言之，解只跟很少一部份的Constraint 有關，如果能把大部分跟解沒有關係的限制式刪掉之後，那答案就呼之欲出了(變成Base case)。舉例來說，上課教的線性規劃，最低的那點其實最後只跟feasible region最下面的限制式有關，其他都可以刪掉。找最小包含所有點的圓的問題也是有一樣的性質，最後的答案只跟在圓周上的那些點有關，其他在圓內部的點都不重要。本次作業的題目也是一樣，大部分的vertices都跟center無關，我們其實只在意離的最遠的那些點而已。

另外，如何縮小搜索範圍也是Prune and Search的重點，這些方法幾乎都是用一些特殊的性質，來確保某些點不會跟解有關，可以安全地被刪掉。其中Median(Centroid of tree)的手法幾乎在全部的Prune and Search 都有被用到，另外讓兩點成雙成對這招也很常用，最後總是會有辦法刪掉pair中其中一個點。這些方法跟觀察問題的結論很有關係，雖然這篇論文提出的觀察結果乍看之下很難看懂，但後來發現其實都是在講很直觀的觀念，如果佐以例子或是圖表輔助理解，應該會更容易。