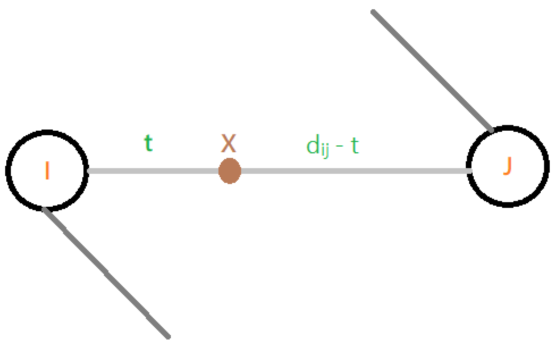
The weighted center of a tree

一、問題定義

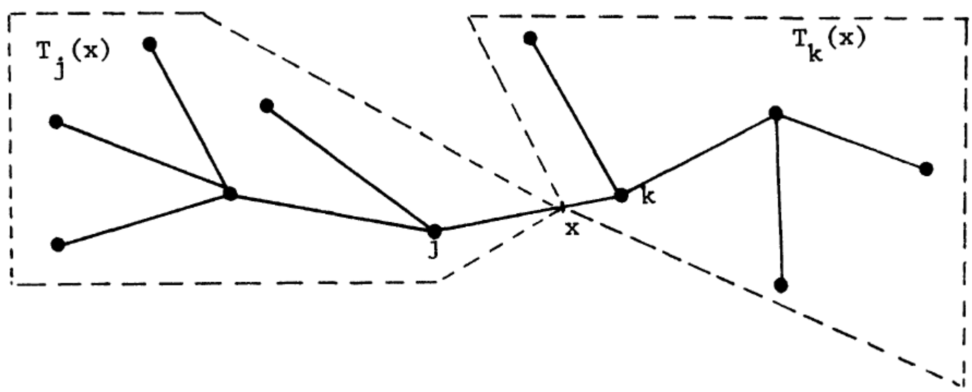
給定一棵具有 n 個節點的樹 T = (V , E)，其中每條邊(i , j)皆為一非負的長度 d_{ij} 且每個節點 i 具有一個非負的權重值 w_i 。

定義一個點 x = (i , j : t) 是位在邊(i , j)上的一點，並且與節點 i 距離 t 單位長度，與節點 j 距離為 d_{ij} - t 單位長度，如下圖表示。



對於此棵樹 T 上，我們可以找到一個加權中心使下列結果最小：

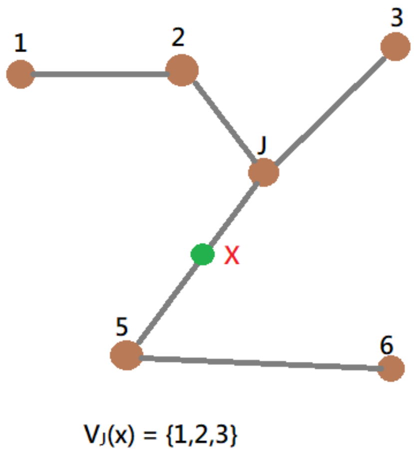
$$r(x) = \max \{ w_i d(x,i)，i \in V \}$$

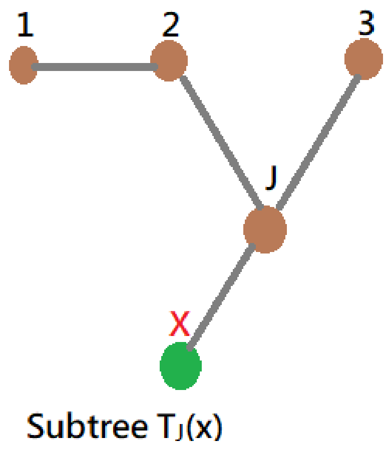


令 x 為樹上的任一個點，若節點 j 與 x 相鄰，則 x 位在與 j 相鄰的一條邊上。我們將點 x 經 simple path 到 i 節點上所會經過的節點 j 所形成的

集合記做 $v_j(x)$，而將 $v_j(x) \cup \{x\}$ 所生成的子樹，記做 $T_j(x)$，

如下圖所示。





最後定義一個函數， $r_j(x) = \max\{ w_i d(x, i)$， $i \in v_j(x) \}$，令 $j_1, j_2, \ldots,$

j_l 為所有與 x 相鄰的節點，則我們可以發現上述的

$$r(x) = \max\{ r_{j1}(x), \ldots, r_{jl}(x) \}。$$

根據上述定義我們可以得到結論，假設存在兩個節點 $j_a; j_b$，使得

$r_{ja}(x) = r_{jb}(x) = r(x)$，則 x 即為此問題的解，因為 r(x) 為一個 convex

function。

二、解法敘述

此問題中所要求解的 r (x)在考慮簡單路徑的狀況下為一 convex function。

也就是說針對一點 x 若為 $r(x) = \max\{ r_{j1}(x), r_{j2}(x), \ldots, r_{jl}(x) \}$ 的一個局

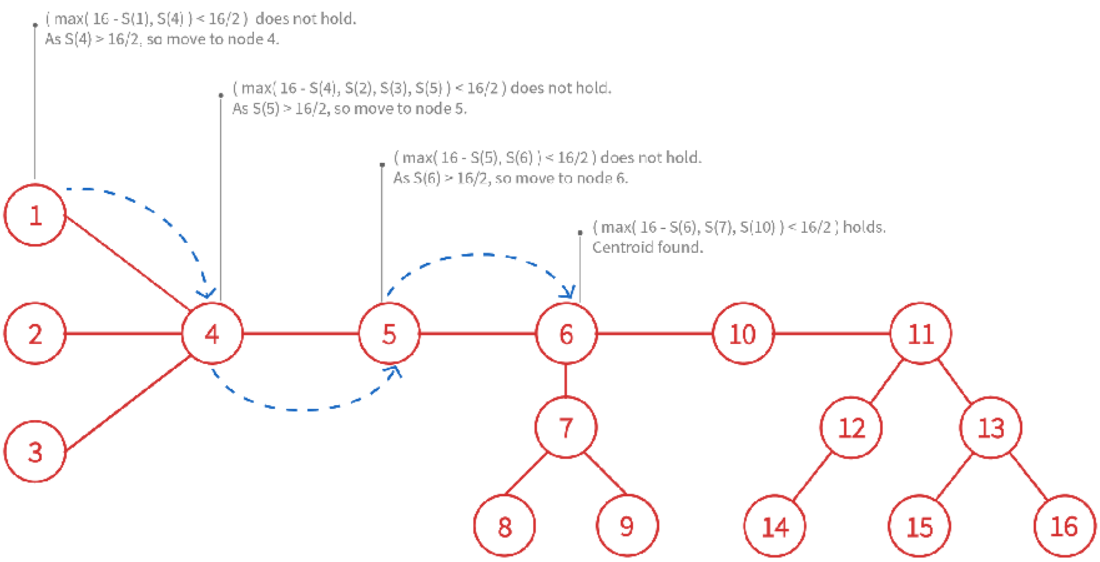
部極值， 根據 convex function 的性質可以知道此解即為所求的加權中心

點。論文作者以 Prune and Search 的方式提出了以下的演算法：

I. Step 1:

首先，令 c 為 centroid of T，根據此篇論文的定義，c 是一個節點使得跟它相鄰的所有節點 j ， $V_j(c) \le n / 2$，n 為此樹的節點數。

這一步只需要 O(n) 的時間就可以找到 c。找到 c 的方法如下，先跑一次 DFS 來找到每個節點的 subtree 節點個數，接下來再跑一次 DFS 來找到 c。在第二次 DFS 當中，我們會不斷往 subtree 節點個數 ≥ n/2 的節點移動，並且記錄該節點，直到移動到某個節點使得該節點 subtree 的節點個數均 < n/2，該節點即為 centroid。而我們知道 DFS 可在線性時間內完成，所以可以在 O(n) 的時間就可以找到 c。

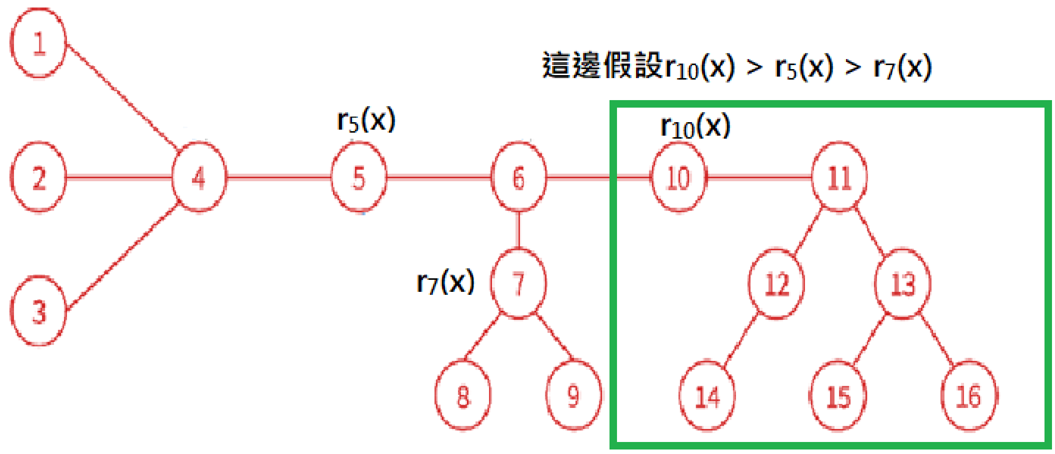


II. Step2:

已知重心 c 並計算所有每一相鄰節點 j 的 $r_j(c)$，此部分可以於 O(n) 時間內完成，並作以下討論：

(1) 若存在兩相鄰節點 j_1 和 j_2 (其中 $j_1 != j_2$) 滿足： $r_{j1}(c) = r_{j2}(c) = r(c)$ 則此重心 c 即為所求加權中心。

(2) 判斷加權中心位置，若存在相鄰節點 j 對於其他所有相鄰節點 k 而言，滿足： $r_j(c) > r_k(c)$ 則可知所求加權重心會位於子樹 $T_j(c)$上。

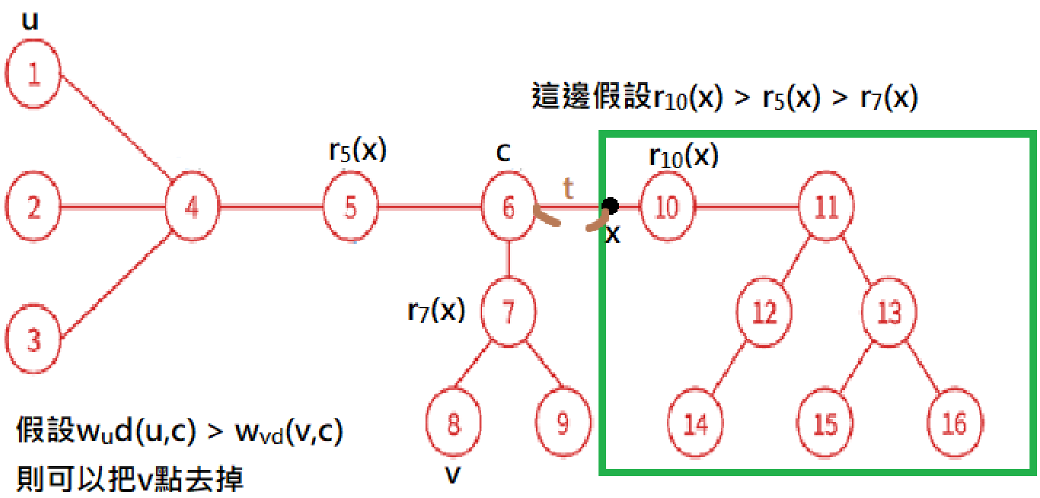


III. Step3:

對於任意兩節點 u, v ∉ $V_j(c)$和與重心 c 距離 t 的點 x ∈ Tj(c) 而言，則此兩點與重心的距離即為 $d(u, x) = d(u, c) + t$ 和 $d(v, x) = d(v, c) + t$，則在求解方程式 $w_u(d(u, c) + t) = w_v(d(v, c) + t)$ 的過程中，不失一般性的狀況下可假設 $w_u d(u, c) \geq w_v d(v, c)$ 若且唯若存在一非負的 t_{uv} 滿足 $0 \leq t \leq t_{uv}$，$t_{uv} = (w_u d(u, c) - w_v d(v, c)) / (w_v - w_u)$，其中：

若加權中心與重心位置小於 t_{uv}，則在尋找加權中心的過程中可以不用考慮節點 v 。

若加權中心與重心位置大於 t_{uv}，則在尋找加權中心的過程中可以不用考慮節點 u。

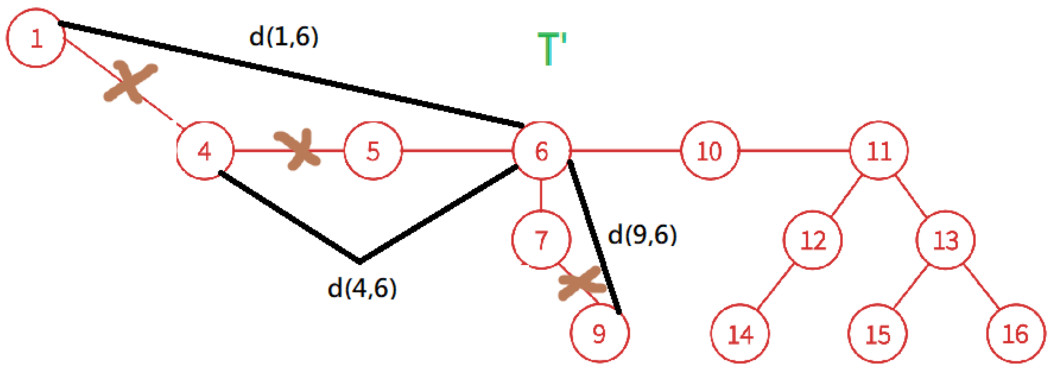


除此之外，計算完所有的 t_{uv} 後，我們計算這些值的中位數 t_m，然後對所有的節點 i 計算 d(c, i)，這步驟能在 O(n)的時間內可以完成，如果能找到一個邊(i, j)使得 $d(x, i) \le tm \le d(x, j)$，那我們就知道在(i, j)之間有一個 y 使得 d(x, y) = tm。所有與 c 距離超過 tm 的點可以組成以 $y_1, y_2, \ldots, y_l$ 為 root 的 subtrees，令他們映射到 subtrees 分別為 $T_1, T_2, \ldots, T_m$，而這些 subtrees 的 root 分別 $u_1, u_2, \ldots, u_m$。定義 $R_i(x) = \max\{ w_k d(x, k), k \in V_i \}$ and $R = \max\{ R_i(u_i), i = 1, \ldots, m \}$，以上計算 R_i 的動作也只需線性時間即可完成，之後分成以下幾種情形，如果 $R_i(u_i) < R$，那此問題的解自然不會在 T_i 當中。然後，可以知道解應該存在與 c 距離小於 tm 的點當

中。最後，如果存在一個 i 使得 $R_i(u_i) = R$，在這個情況下，我們只需要從 $r_j(y)$ 來判斷，而 j 為任何一個與 u_i 相鄰的點，那 $y = u_i$。

IV. Step4:

對所有沒被刪掉的節點 u，新增一條邊(u, c)，並從 T 當中算出 d(u, c) 以及沿用 w_u，將這些節點加回 Tj(c) 當中，組成一顆新的樹 T'。重複流程直到找到解為止。



V. Time Complexity

根據 c 的定義，不在 $T_{jmax}(x)$ 的節點至少有 n/2 個，所以至少會有 n/4 個配對，在每個配對中，我們至少會刪除其中一個節點，以在每一輪中，至少會有 n/8 個節點被刪除，又每輪的時間複雜度為 O(n)，則 $T(n) \le T(7n/8) + O(n)$，最後我們可以得到 $T(n) = O(n)$。

三、讀後心得

Prune and Search 為 Divide and Conquer 的一種特例，常被用於求解最

佳化問題。problem2 就是採用此策略去解 2-Dimensional Linear Programming，將原本的線性規劃問題進行簡化，直到 problem set 夠小後，直接去解他，進而免去了對矩陣進行高斯消去法的繁複計算，我也在網路上稍微查詢了一下，使用 Prune and Search 概念求解的問題其實不算多，大致上有：Linear Programming in the Plane、The Weighted Center of Tree、Smallest Circle enclosing n Points 與 Linear Programming in R^3。在寫作業的過程中發現其實整體演算法的內容其實十分相近，皆是藉由配對後找出中位數，並依條件刪去掉不需要納入考慮的限制式或者是節點。在本篇論文出來之前，The Weighted Center of Tree 問題的最佳解為 O(n log n)，而本篇論文利用 prune and search 方法，找到一個 O(n) 的解法，關鍵便是在每一輪的計算中，都會去除掉一部分的節點，使得此演算法能在線性時間內完成。

另外，我覺得這篇論文有些地方並沒有解釋得很清楚，讓我有些地方一看再看才能理解，閱讀這篇論文的時候讓我體會到教授提到要舉例的重要性，之後的作業我會更注意在如何淺顯易懂的讓人理解論文。