DSCA 第二篇 paper 閱讀報告

問題敘述：

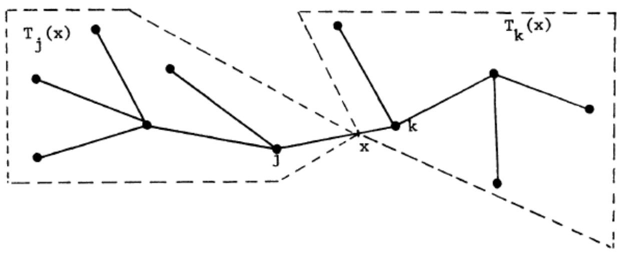
想找一個 weighted center of a tree，則此 weighted center 須滿足到各個 vertex 的加權值，也就是$w_i d(x, i)$: $i \in V$，並從各個加權值中找最大者，即是這個 tree 中的 weighted center。

解題方法：

Step 1. 一開始，可以先透過[HM]找到 centroid。

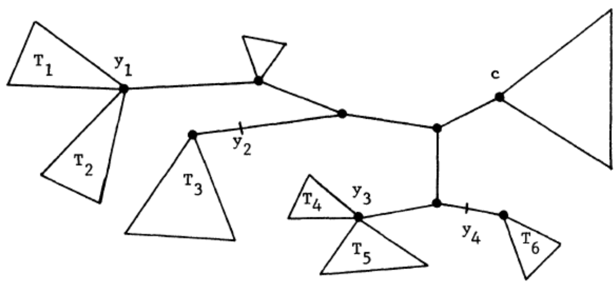
Step 2. 並從各個相鄰 centroid 的 vertex 中算出他們所形成的 subtree 中各個 vertex 到 centroid 的最大加權值。

Step 3. 假如有兩個以上的最大加權值，則 centroid 就會是 weighted center；相反的，只有一個最大的加權值，這 weighted center 就會在這個最大加權值所代表的 subtree 中。像是：



Step 4. 從這個 subtree 中，選擇任一個 leaf，到這個 leaf 的某一個距離的 edge，把超過且包含在這個 edge 上的每一個 vertex 都當成會在形成另一個 subtree，並分析這些新形成的 subtree 中最大加權值。這些最大加權值的最大值如果有兩個以上，weighted center 就不在這個 subtree 中；如果只有一個則 weighted center 就有可能在這個 subtree 中；小於這個最大值者則 weighted center 也不會在該 subtree 中。

Step 5. 從 Step 4. 中，可以發現 weighted center 在 subtree 中的某段距離內，然後從 subtree 外部找兩個點，並計算他們的加權值。假如為了使 $w_u d(u, x) \geqq w_v d(v, x)$ 是這個不等式成立的 t_{uv}，小於或大於上述的某段距離，則可以把外部其中一點刪掉。像是：



Step 6. 在 Step 5. 中的各個 t_{uv} 中，找到 t_m，把各段距離中的 weighted

center x^*找出來，並把x^*跟外部任兩點的加權值找出來。如果t_{uv}不超過或者不少於t_m，則外部任兩點可以刪掉其中一點。

Step 7. 反覆進行上面的 Step 4. 到 Step 6.。在 subtree 中最多有 n/2 個 vertex，並且至少有 $[n/4]-1$個 pairs，並且每次會把 pairs 裡其中一個 vertex 砍掉，也就是說，每次至少可以砍掉 n/8 個 vertex。

時間複雜度分析：

從上述的解題方法中，可以得出此演算法的 recurrence relation 為 $\text{time}\ (n) \leqq \text{time}\ (7n/8) + Cn$。分析他的 recursion tree，能得知$\text{time}\ (n) = O(n)$，亦即，可以在 O(n)時間複雜度內，就可以得知 weighted center of a tree。

心得：

又看到另外一個關於 prune and search 的實際運用的例子，運用 prune and search 可以成功把需多問題的時間複雜度降低，遠勝於其他能解此一問題的演算法。然而，可以看到 prune and search 解問題的分析和過程，並沒有那麼容易，而且用 code 實踐還需另外考慮很多狀況，雖然手法都很像。關於我個人對 prune and search 的感覺是，還需要多加了解，才能熟悉運用 prune and search 來分析其他問題。