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，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。假如為了使是這個不等式成立的，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. 中的各個中，找到，把各段距離中的weighted center 找出來，並把跟外部任兩點的加權值找出來。如果不超過或者不少於，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 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。分析他的recursion tree，能得知，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。