# Homework 18

Brian Knotten, Brett Schreiber, Brian Falkenstein

October 7, 2018

## 28

The decision tree for this problem can be constructed with the following rules:

- Nodes in the tree have $n$ children, where the $i'th$ child for $1 <= i <= n$ corresponds to choosing $v_i$ to be a center point. Thus, the tree has height $k$ and will contain all solutions at the leaves.
- The nodes will store the current aggregate distance, that is, the sum of all the distances from each $v \in V$ to its closest center point $c \in C$.

The total size of the tree is $n^k$. This can then be pruned with the following rules:

1. If a node at depth $i$ already exists in $C$, prune it. That is, if that vertex has already been added to $C$, remove that sub tree, as a valid solution contains $k$ different vertices in $C$.
2. If two nodes $u$ and $v$ at depth $i$ have the same $c_1...c_{i-1}$ center points, and $D_u < D_v$, prune $v$ (where $D_u$ is the total aggregate distance for $u$).

The first pruning rule leaves the tree as still roughly $n^k$. However, under the second pruning rule, every depth $i$ is limited to $n$ nodes. This is because at depth $i-1$, for some node $v$, all of $v's$ children have have the same $c_1...c_{i-1}$, and only one will not be pruned, that is the one with the shortest $D$. This brings the size of the tree down to $nk$. At each node $v$, a new aggregate distance must be calculated, which takes $n$ calculations, which leaves this tree with a run time of $n^2k$.
The dynamic algorithm is given as follows:

```
A = [k, n]                          %construct A to be an array of size k by n
A[0, :] = infinity                  %having zero centers gives a distance of infinity
for i = 1 to k do                  %loop over the rows
     last_v = min(A[i-1, :])  %grab the minimum D from the previous row. use this V for subsequent
                                        %calculations on the current row
     for j = 1 to no do          %for each row, check all possible center values
         if(last_v != v[j])          %can't repeat a vertex in the solution
               A[i, j] = new_dist(last_v, V[j])   %calculate new distance with V[j] as a new center

output min(A[k, :])              %return the minimum value in the last row
```

Here, A[i, j] represents the minimum aggregate distance using $i$ center points, where $v_j$ is the last vertex to be chosen as a center.