Homework 16

Brian Knotten, Brett Schreiber, Brian Falkenstein

October 3, 2018

20

A decision tree is as follows: The decision tree can be thought of as follows for each node v at depth i:

- v designates each of $p_1, p_2, ...p_i$ into lists A and B, which represent the routes that either taxi A or taxi B travels.
- v's left child copies the assignments of v to A and B, but appends point p_{i+1} to A.
- v's right child copies the assignments of v to A and B, but appends point p_{i+1} to B.
- At depth i, i points have been visited.

Because each point needs to be visited by at least one taxi, and since there are no benefits for having both taxis visit a point, each point is either visited by taxi A or by taxi B. So there are 2^n possible point designations at the leaves of the decision tree.
The decision tree can be pruned using the following rules:

1. If two nodes u and v have the same total distance, $dist(u_A) + dist(u_B) = dist(v_A) + dist(v_B)$, prune v.

```
taxi(i, a, b):
    if i > n:
        return 0

    return min(
        dist(p[a], p[i]) + taxi(i + 1, i, b)
        dist(p[b], p[i]) + taxi(i + 1, a, i)
    )
```

Let $taxi[i, a, b]$ = the minimum total distance of designating A and B over points $p_1...p_i$, and A's last stop is p_a, and B's last stop is p_b. The dynamic solution algorithm is:

```
for a = 1 to n do:
    for b = 1 to n do:
        for i = n to 1 do:

            taxi[a, b, i] = min(
                                taxi[i, b, i + 1] + dist(p[a], p[i]),
            taxi[a, i, i + 1] + dist(p[b], p[i])
            )
```

21

The decision tree of depth n can be defined as follows. Given a node v at depth i:

- v contains a list of stops of length i and the cumulative response time to make i stops.
- v has n children, where the jth child cointains v's path appended with x_j.

Therefore, the solution is the leaf node (a path of n stops) with the minimum average response time. Without pruning, the size of the tree will be n^n (height of n since every path must cover all points, and a branching factor of n). The tree can then be pruned using the following rules:

1. If a node v visits a point in its path that it has already visited, prune v.
2. If nodes u and v are at the same depth and have visited the same points, and u has an equal or shorter average response time, then prune v.

The first pruning rule ensures that every node v at depth i only has $n - i$ children, since i children are already somewhere in v's path.

The second pruning rule ensures that at every depth i, there will only be one node v whose path goes through the set of i points. That is, every node at a given depth are only distinguished by the points they go through.

$A[i, j] =$ The average response time making i stops in total, and ending at point x_j.

A supplementary array Visited is needed, where $Visited[i] = 1$ if point x_i has been visited during the algorithm, otherwise $Visited[i] = 0$.

```
for i = 1 to n do:
    Visited[i] = 0

for i = 1 to n do:          # For every next stop $i$

    for j = 1 to n do:      # For every point being considered:
        if Visited[j] = 1:  # If that point has already been added,
            continue        # Skip it


        if A[i - 1, j] < A[i - 1, minimum]:
            minimum = j


    A[i, minimum] = 0
    Visited[minimum] = 1


Output A[n, k] where k produces the minimum average waiting time.
```