CS32 HW4 Report

1e: Iterator p points to random spot in PC’s memory when new items are inserted, so after the insertion the value p points to will never equals to MAGIC.

3. In the function insert which called FindAtLeast, which compares p->m\_value to value. Both operands are coord and we haven’t define their < operands. Also in insert, p-> m\_value == value, both operands are coord and we haven’t define its == operator.

4b. By using only one variable, there’s no way to track the string path from all the way from the root to every single node.

5a. const int N = *some value*;

assert(N > 2); // algorithm fails if N <= 2

double dist[N][N];

...

int bestMidPoint[N][N];

for (int i = 0; i < N; i++) O(N)

{

bestMidPoint[i][i] = -1; // one-stop trip to self is silly O(1)

for (int j = 0; j < N; j++) O(N)

{

if (i == j)O(1)

continue;

int minDist = *maximum possible integer*;

for (int k = 0; k < N; k++). O(N)

{

if (k == i || k == j)

continue;

int d = dist[i][k] + dist[k][j];

if (d < minDist)

{

minDist = d;

bestMidPoint[i][j] = k;

}

}

}

}

With three major for loops each with steps of N, and some trivial operations between them which doesn’t affect the highest order term so the O is N\*N\*N = O(N^3).

5b. Although this algorithm is almost twice as fast as the first one, its time complexity is still O(N^3). Only the second for loop has a major change which might affect the highest order term of O. So, the first loop has O of N. The second and third for loop has O of (0+1+2+…+N) which is N(N+1)/2 which approximate to N^2. So, the total O is still O(N^3).

6a. The worst case scenario is two sets of size N have totally different content. get(k,v) will find the kth item in the set, which takes a length of min(k, N-k). This is a constant inside each loop. Insert check for copies, and at the worst case, s1’s numbers are all smaller than s2’s. So each findFirstAtLeast will visit N items in each loop, which results insert checking N items in a loop. The loop runs for N times, and the codes before the for loop contribute minor factors to O. So, the worst time complexity is O(N^2).

6b.Copying s1 and s2 to V is O(N), sort is O(NlogN), delete original set is O(N), copy unique items from v to original set is O(N), destroy v is O(N). So the dominant term is sort O(NlogN). Since we are not having loop inside loop in this problem. All of the time complexity is adding up instead of multiplying. So, only the most dominant O is this function’s time complexity which is O(NlogN).

6c.Only the while loop and for loop’s time complexity matters. The while loop has O(N), and the worst case of for loop has time complexity O(N). So this function’s time complexity is O(N).