Cody Do

105140467

Professor Smallberg 3C

Homework 4 Report

2. Consider the following:

*#include "Map.h" // class template from problem 1*

*class Coord*

*{*

*public:*

*Coord(int r, int c) : m\_r(r), m\_c(c) {}*

*Coord() : m\_r(0), m\_c(0) {}*

*double r() const { return m\_r; }*

*double c() const { return m\_c; }*

*private:*

*double m\_r;*

*double m\_c;*

*};*

*int main()*

*{*

*Map<int, double> mid;*

*mid.insert(42, -1.25); // OK*

*Map<Coord, int> mpi;*

*mpi.insert(Coord(40,10), 32); // error!*

*}*

The reason why the call to Map<Coord, int>::insert causes an error lies in how both the template Map class and the Coord class are implemented. The template Map class uses a comparison (!=) between one key value and another key value. When both key values are primitive data types, a mode of comparison already exists. However, since Coords are a class that has no overloaded operators that tell the compiler whether two Coords are equal or not, there does not exist a mode of comparison. Thus, the call to Map<Coord, int>::insert causes an error since the command depends on such a comparison.

3a. O(N3); When looking at the Big O of an algorithm, we always look at the “worst-case”. Since the algorithm given in 3a has three for-loops nested within each other—with all of them limited to N iterations—the Big O is simply N\*N\*N, or N3.

3b. O(N3); Once again, we look at the “worst-case” situation of an algorithm and compute a Big O off of that. In the new algorithm, there are still three for-loops. The outermost is limited to N steps, the second-nested for-loop is limited to i (the incrementing variable in the outermost loop), and the innermost nested for-loop is limited to N. In the worse case sitaution, i will essentialy be equal to N. Thus, the Big O is once again N\*N\*N, or N3.

4a. O(N2); This algorithm has one for loop and many constant-step function calls. The first get, insert, and swap functions can be ignored as they are all constant-step functions. Thus, this algorithm relies on the for-loop, which is restricted to N times. However, the for-loop calls on the get function, with an input of “i” as the location. Since i in the first case scenario is N—and the get function makes use of a for-loop that depends on i—, this is essentially a nested for-loop. Due to this, the Big O of the algorithm is N\*N, or N2.

4b. O(N); This algorithm makes use of one for-loop. As the for-loop is limited to size N of the linked list, the big O of this algorithm is merely N. Thus, this algorithm is better than the one in question 4a as it takes fewer steps to get the same end result.