|  |  |  |
| --- | --- | --- |
| Function | Time Analysis | Space Analysis |
| operator<(const Node & rhs, const Node & lhs) | Theta(1) | Theta(1) |
| Node() | Theta(1) | Theta(1) |
| pokeInput(int stops,std::priority\_queue <Node> &myQueue, std::vector<Pokemon> &myPokemon) | For n times //n poke-stops  //push to queue is lg(n)  Theta(n(lg(n))) | For n times //n poke-stops  Theta(n) |
| checkBestNode(Node checkNode, Node & myBestNode) | Theta(1) | Theta(1) |
| calcBound(std::priority\_queue <Node> & myQueue, std::vector<Pokemon> myPokemon, Node & bestNode) | while(!myQueue.empty() //n!  myQueue.pop() //lg(n)  nested for loops n2 time because tempNode.pokemonCaught approaches myPokemon.size  Theta(n!n2) | Push\_back takes 1 space \* n! times  Theta(n!) |
| Main() | pokeInput //n(lg(n))  +  pokeStops //n  +  calcBound //n!n2  Theta(n!n2) | Theta(n!) |

Failed tests 1 and 3, tried to debug but couldn’t get these cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | Reason for the test | Actual Input Data | Expected Output Data | Actual Output |
| 1 | Test for multiple Pokémon all at 0,0 | 6 0 0 pokemon1 0 0 pokemon2 0 0 pokemon3 1 1 pokemon1 2 2 pokemon2 3 3 pokemon3 | 0 | 4 |
| 2 | Testing straight line path | 4 0 1 poke1 0 2 poke2 0 3 poke3 0 4 poke4 | 8 | 8 |
| 3 | Test boundaries | 4  10 10 poke1  10 -10 poke2  -10 -10 poke3  -10 10 poken4 | 100 | 80 |
| 5 | Test for a duplicate Pokémon at different stops | 4  -5 -5 poke1  -3 -4 poke2  -10 -10 poke3  -2 -2 poke3  -1 -5 poke4 | 20 | 20 |