```
1  /*-------------------------------------------------------------------
2  Copyright (c) 2014 Author: Jagadeesh Vasudevamurthy
3  file: dsorttest.cpp
4  you require:  ../util/util.cpp ../complex/complex.cpp dsorttest.cpp
5  On linux:
6  g++  ../util/util.cpp ../complex/complex.cpp dsorttest.cpp
7  valgrind a.out
8  -- All heap blocks were freed -- no leaks are possible
9  -------------------------------------------------------------------*/
10
11 /*-------------------------------------------------------------------
12 This file test dsort object
13 -------------------------------------------------------------------*/
14
15 /*-------------------------------------------------------------------
16 All includes here
17 -------------------------------------------------------------------*/
18 #include "dsort.h"
19 #include "../complex/complex.h"
20
21 /*-----------------------------------------------------------
22 static definition - only once at the start
23 Change to false, if you don't need verbose
24 -----------------------------------------------------------*/
25 template <typename T>
26 bool dsort<T>::_display = false;
27
28 /*-----------------------------------------------------------
29 local to this file. Change verbose = true for debugging
30 -----------------------------------------------------------*/
31 static bool verbose = true;
32
33 /*-------------------------------------------------------------------
34 test complexity
35 -------------------------------------------------------------------*/
36 static void test_complexity(const char* t, const int N, int w){
37   Random r;
38   dsort<int>::set_display(false);
39   darray<int> a;
40   for (int i = 0; i < N; ++i) {
41     int x = r.get_random_number(1000000);
42     if (i) {
43       a[i] = -x;
44     }
45     else {
46       a[i] = x;
47     }
48   }
49   cout << "-------------------------" << endl;
50   cout << t << endl;
51   dsort<int> s(a, int_descending_order, N);
52   if (w == 0) {
53     s.bubble_sort();
54   }
55   else if (w == 1) {
56     s.insertion_sort();
57   }
58   else if (w == 2) {
59     s.merge_sort();
60   }
61   else if (w == 3) {
62     s.quick_sort();
63   }
64 }
65
66 /*-------------------------------------------------------------------
```

```
67 main
68 ------------------------------------------------------------------*/
69 int main() {
70   const char* s[] = { "bubble sort", "insertion sort", "merge_sort", "quick sort" };
71   int size = sizeof(s) / sizeof(char *);
72   for (int i = 0; i < size; ++i) {
73     for (int n = 2; n < ((2 << 20) + 1); n = n * 2) {
74       test_complexity(s[i], n, i);
75     }
76   }
77   return 0;
78 }
79
80
81 //EOF
82
83
```