VE281 Homework I

Performance Analysis for Sort Algorithms

Name: Tianyi Ge **Stu Number:** 516370910168

1 Source Codes

Listing 1: Sort algorithms

```
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

typedef struct node_t {
    int key;
    struct node_t *next;
    node_t(int v) {key = v; next = NULL;}
} node_t;

void add(node_t *prev, int key) {
    node_t *tmp = new node_t(key);
    tmp->next = prev->next;
    prev->next = tmp;
}

void swap(int &a, int &b) {
    int t = a; a = b; b = t;
}

void bubblesort(int *d, int n) {
    for (int i = 0; i < n; ++i)
        for (int j = n - 1; j > i; --j)
            if (d[j] < d[j-1])
                swap(d[j], d[j-1]);
}

void insertionsort_array(int *d, int n) {
    for (int i = 1; i < n; ++i) {
        int key = d[i], j;
        for (j = 0; j < i && d[j] <= key; ++j);
        for (int k = i; k > j; --k) d[k] = d[k-1];
```

```
35          d[j] = key;
36      }
37  }
38
39  void insertionsort_list(int *d, int n) {
40      node_t *head = new node_t(0);
41      for (int i = 0; i < n; ++i) {
42          node_t *tmp = head;
43          while (tmp->next && tmp->next->key < d[i]) tmp = tmp->next;
44          add(tmp, d[i]);
45      }
46      node_t *tmp = head->next, *ptr;
47      while (tmp) {
48          cout << tmp->key;
49          ptr = tmp;
50          tmp = tmp->next;
51          delete ptr;
52      }
53      delete head;
54  }
55
56  void selectionsort(int *d, int n) {
57      for (int i = 1; i < n - 1; ++i) {
58          int flag = i;
59          for (int j = i + 1; j < n; ++j)
60              if (d[j] < d[flag])
61                  flag = j;
62          swap(d[flag], d[i]);
63      }
64  }
65
66  void merge(int *d, int l, int m, int r) {
67      int i = l, j = m + 1, k = 0;
68      int *tmp = new int[r-l+1];
69      while (i <= m && j <= r) {
70          if (d[i] <= d[j]) tmp[k++] = d[i++];
71          else tmp[k++] = d[j++];
72      }
73      while (i <= m) tmp[k++] = d[i++];
74      while (j <= r) tmp[k++] = d[j++];
75      for (i = l; i <= r; ++i) d[i] = tmp[i-l];
76      delete[] tmp;
77  }
78
79  void mergesort(int *d, int l, int r) { //close interval
80      if (l >= r) return;
81      int m = ((l + r) >> 1);
82      mergesort(d, l, m);
83      mergesort(d, m + 1, r);
84      merge(d, l, m, r);
85  }
86
87  void quicksort_extra(int *d, int l, int r) {
```

```
88      if (l >= r) return;
89      int p = rand()%(r-l+1)+l;
90      swap(d[l], d[p]);
91      int key = d[l];
92      int i = 0, j = r - l;
93      int *b = new int[r-l+1];
94      for (int k = l + 1; k <= r ; ++k) {
95          if (d[k] < key) b[i++] = d[k];
96          else b[j--] = d[k];
97      }
98      b[i] = key;
99      for (int k = 0; k <= r - l; ++k) d[k + l] = b[k];
100     delete[] b;
101     quicksort_extra(d, l, l + i - 1);
102     quicksort_extra(d, l + i + 1, r);
103 }
104
105 void quicksort_inplace(int *d, int l, int r) {
106     if (l >= r) return;
107     int p = rand()%(r-l+1)+l;
108     swap(d[l], d[p]);
109     int key = d[l];
110     int i = l, j = r;
111     while (i < j) {
112         while(d[j] >= key && i < j) --j; // make sure finally i == j and d[j]=d[i] <
                key so that you could put it on the left
113         while(d[i] <= key && i < j) ++i;
114         if (i < j) swap(d[i], d[j]);
115     }
116     d[l] = d[i];
117     d[i] = key;
118     quicksort_inplace(d, l, i - 1);
119     quicksort_inplace(d, i + 1, r);
120 }
121
122 void rd(int *d, int n){
123     for (int i = 0; i < n; ++i)
124         cin >> d[i];
125 }
126
127 void prt(int *d, int n) {
128     for (int i = 0; i < n; ++i)
129         cout << d[i] << "\n";
130 }
131
132 int main() {
133     ios::sync_with_stdio(false);
134     srand(time(NULL));
135     int cmd, n;
136     cin >> cmd >> n;
137     int *d = new int[n];
138     rd(d, n);
139     int start = clock();
```

```
    switch(cmd) {
        case 0: bubblesort(d, n); break;
        case 1: insertionsort_array(d, n); break;
        case 2: selectionsort(d, n); break;
        case 3: mergesort(d, 0, n-1); break;
        case 4: quicksort_extra(d, 0, n-1); break;
        case 5: quicksort_inplace(d, 0, n-1); break;
        default: return 0;
    }
    //prt(d, n);
    cout << (clock() - start)*1.0/CLOCKS_PER_SEC << "\n";
    delete[] d;
    return 0;
}
```

Listing 2: Testcase generator

```
#!/usr/bin/python

import sys
reload(sys)
sys.setdefaultencoding('utf-8')

import random
import os
import commands
import time

MAX = 15
INT_MAX = 2**31;
size = [5, 5000, 8000, 12000, 20000, 30000, 40000, 50000, int(0.25e7), int(0.5e7),
    int(1e7), int(1.5e7), int(2.0e7), int(2.5e7), int(3e7)]
if __name__ == "__main__":
    for cases in range(MAX):
        with open('input{}'.format(cases), 'w') as w:
            n = size[cases]
            w.write(str(n) + '\n')
            for i in range(n):
                w.write(str(random.randint(-INT_MAX, INT_MAX-1)) + '\n')
        print("Testcase{}".format(cases))
```

Listing 3: Run the programs

```
#!/bin/zsh

for i in `seq 0 2`
do
    echo $i
    for j in `seq 0 7`
    do
        echo $i | ./a1_test < input$j >> $i.out
        echo $j $?
```

```
    done
done

for i in `seq 3 5`
do
    echo $i
    for j in `seq 8 14`
    do
        echo $i | ./a1_test < input$j >> $i.out
        echo $j $?
    done
done
#rm -rf Testcase*
```