Homework 1: Introduction to Algorithmic Analysis and Recurrence

Author: Gabriel Hofer

CSC-372 Analysis of Algorithms

Instructor: Dr. Rebenitsch

Section 1 DUE: Thursday, Aug 27th, at 7AM
Section 2 DUE: Thursday, Sept 3 th, at 7AM

Department of Computer Science and Engineering
South Dakota School of Mines and Technology

Introductory Information

1. (3 pt) How soon do you need to notify me for a normal extension?
   36 hours

2. (3 pt) How many projects will there be?
   5 projects

3. (3 pt) How long do you have to notify me for a possible grading error, starting when? the grade
   One week

4. (3 pt) What is the ONLY option to bring up your grade at the end of the semester?
   Take the optional second chance Exam

5. (8 pt) When did you attend ZOOM office hours after Aug 19 (this will confirmed later)?
   August 21, 2020

6. (3 pt) Should your microphones/video initially be on or off when attending a Zoom recitation/office hours.
   Start with camera and microphone off.

7. (3 pt) What topic(s) are tentatively planned for Oct. 9?
   F: Closest Pair of points

8. (3 pt) At minimum view, the entry quiz (competition is not required)
   Check-in Completed

9. (6 pt) What is the run time for the following code. You MUST show your work for any credit

```
Let A be an array of size n
for a in the range of 1 to x
    for b in the range of 1 to y
        for c in the range of 1 to z
            print all of A
```

The outer loop executes x times. The first nested loop executes y times. The second nested loop executes z times. Printing all of A requires n operations. Because these loops are all nested, we multiply their run times together.

$$\text{Runtime} = O(x) * O(y) * O(z) * O(n) \Rightarrow \text{Runtime} = O(x * y * z * n)$$

Insertion Sort:

```
vector<int> insertionSort(vector<int> & v) {
    int key, i;
    for (int j = 1; j < v.size(); j++) {
        key = v[j];
        i = j - 1;
        while (i >= 0 && v[i] > key) {
            v[i + 1] = v[i];
            i--;
        }
        v[i + 1] = key;
    }
    return v;
}
```

Merge Sort:

```
void merge(vector<int> & A, int p, int q, int r) {
    vector<int> L(q - p + 1 + 1, INT_MAX), R(r - q + 1, INT_MAX);
    for (int i = p, j = 0; i <= q; i++, j++) L[j] = A[i];
    for (int i = q + 1, j = 0; i <= r; i++, j++) R[j] = A[i];
    for (int i = p, j = 0, k = 0; i <= r; i++)
        A[i] = (L[j] <= R[k]) ? L[j++] : R[k++];
}

void mergeSort(vector<int> & A, int p, int r) {
    if (p < r) {
        int q = (p + r) / 2;
        mergeSort(A, p, q);
        mergeSort(A, q + 1, r);
        merge(A, p, q, r);
    }
}
```

Records and Data

N	Insertion	Merge
2	2.8E-06	2.26E-05
12	7.4E-06	6.82E-05
22	1.32E-05	0.000144
32	3.16E-05	0.000426
42	5.17E-05	0.000278
52	5.44E-05	0.000637
62	0.000105	0.000423
72	0.000109	0.000477
82	0.000139	0.000845
92	0.00028	0.000641
102	0.000277	0.00071
112	0.000254	0.000824
122	0.000322	0.00091
132	0.000279	0.000884
142	0.000296	0.000836
152	0.000451	0.001013
162	0.000668	0.001512
172	0.000454	0.000996
182	0.00068	0.001134
192	0.000551	0.001193
202	0.000684	0.001308
212	0.001111	0.001311
222	0.000864	0.001433
232	0.000894	0.001658
242	0.001053	0.001791
252	0.001188	0.002537
262	0.001353	0.001944
272	0.001536	0.00211
282	0.001573	0.001898
292	0.002229	0.002178
302	0.001794	0.002423
312	0.001817	0.002344
322	0.001807	0.003932
332	0.001809	0.002253
342	0.00255	0.004014
352	0.002653	0.002773
362	0.002819	0.002902
372	0.003229	0.003095
382	0.002981	0.003008
392	0.003653	0.003061
402	0.00306	0.002751
412	0.003643	0.004946
422	0.004085	0.003657
432	0.003703	0.005908
442	0.003743	0.003569
452	0.004199	0.004885
462	0.005681	0.003755
472	0.004103	0.004707
482	0.004412	0.003414
492	0.005028	0.003526
502	0.005166	0.003779
512	0.007006	0.003937
522	0.007487	0.003739
532	0.00668	0.006516
542	0.005296	0.00511
552	0.005203	0.004216
562	0.006328	0.005794
572	0.006098	0.004787
582	0.007263	0.005589
592	0.007651	0.005159
602	0.008113	0.005112

N	Insertion	Merge
2	4.8E-06	1.45E-05
32	3.27E-05	0.000311
62	0.000112	0.000435
92	0.000198	0.000647
122	0.000281	0.000877
152	0.000392	0.001051
182	0.000583	0.001278
212	0.000796	0.001463
242	0.001419	0.002321
272	0.002022	0.004091
302	0.002543	0.003632
332	0.003025	0.003635
362	0.004464	0.005102
392	0.0053	0.005258
422	0.00504	0.004503
452	0.005122	0.004523
482	0.004094	0.00375
512	0.004182	0.003466
542	0.005663	0.004358
572	0.008436	0.00525
602	0.008722	0.005673
632	0.009864	0.00575
662	0.010688	0.008104
692	0.012003	0.007729
722	0.012004	0.005653
752	0.011041	0.006331
782	0.01257	0.007183
812	0.014293	0.006099
842	0.01298	0.007149
872	0.014778	0.006656
902	0.018732	0.008235
932	0.016635	0.007442
962	0.018255	0.007099
992	0.018113	0.008137
1022	0.021694	0.00899
1052	0.024052	0.014016
1082	0.02547	0.009001
1112	0.025947	0.008596
1142	0.024558	0.009166
1172	0.027252	0.0108
1202	0.031999	0.011615
1232	0.029614	0.009332
1262	0.034356	0.012114
1292	0.039802	0.018422
1322	0.041124	0.010386
1352	0.03685	0.010469
1382	0.039021	0.013496
1412	0.046007	0.012005
1442	0.038869	0.01037
1472	0.045933	0.012891

Plotting Insertion Sort vs Merge Sort