

# COMSC-200

## Lab 4

Ryan Jacoby

20 September 2020

### 1 Quicksort vs. Select Median

Median Select is around 28 times faster than Quicksort.

```
1 // Ryan Jacoby
2
3 /*
4 Done! (Intel Core-i5 8600k@4.6GHz)
5
6 --Sort + Middle--
7 Min: 95149      Max: 96755      Avg: 95649
8
9 --Mean--
10 Min: 1675      Max: 5597      Avg: 3467
11
12
13 Done! (Intel Core-i9 9900k@5.00GHz)
14
15 --Sort + Middle--
16 Min: 85769      Max: 90052      Avg: 86201
17
18 --Mean--
19 Min: 1583      Max: 4988      Avg: 3221
20 */
21
22 #include<iostream>
23 #include<ctime>
24 #include<chrono>
25
26 using namespace std;
27
28 void printArr(int[], int);
29 void swap(int[], int, int);
30 int select(int[], int, int, int);
31 void quickSort(int[], int, int);
32 int med(int, int, int);
33 void fillArr(int[], int);
34
35 int main() {
36     srand(time(NULL));
37
38     int a[100000];
39
40     int s[100];
```

```

41     int q[100];
42
43     for(int i = 0; i < 100; i++) {
44         fillArr(a, 100000);
45
46         auto start = chrono::high_resolution_clock::now();
47         select(a, 100000 / 2, 0, 100000);
48         auto stop = chrono::high_resolution_clock::now();
49
50         s[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
51
52         fillArr(a, 100000);
53
54         start = chrono::high_resolution_clock::now();
55         quickSort(a, 0, 99999);
56         stop = chrono::high_resolution_clock::now();
57
58         q[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
59
60         cout << i << "\r";
61     }
62
63
64
65     cout << "Done!\n";
66
67     cout << "\n--Sort + Middle--\n";
68     int min = q[0], max = q[0], sum = 0;
69     for(int i = 0; i < 100; i++) {
70         if(q[i] > max) max = q[i];
71         if(q[i] < min) min = q[i];
72         sum += q[i];
73     }
74
75     cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';
76
77     cout << "\n--Mean--\n";
78     min = s[0];
79     max = s[0];
80     sum = 0;
81     for(int i = 0; i < 100; i++) {
82         if(s[i] > max) max = s[i];
83         if(s[i] < min) min = s[i];
84         sum += s[i];
85     }
86
87     cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';
88
89     return 0;
90 }
91
92 void swap(int arr[], int a, int b) {
93     int temp = arr[a];
94     arr[a] = arr[b];
95     arr[b] = temp;
96 }
97
98 void printArr(int arr[], int n) {
99     for(int i = 0; i < n; i++)

```

```

100     cout << arr[i] << '\t';
101     cout << '\n';
102 }
103
104 int select(int arr[], int k, int a, int b) {
105     int pivot = arr[b];
106     int i = a;
107     for(int j = a; j < b; j++)
108         if(arr[j] <= pivot) {
109             swap(arr, i, j);
110             i++;
111         }
112     swap(arr, i, b);
113
114     if(i - a == k) return arr[i];
115     if(i - a > k) return select(arr, k, a, i - 1);
116     return select(arr, k - i + a - 1, i + 1, b);
117 }
118
119 int med(int a, int b, int c) {
120     if (a >= b && a <= c || a >= c && a <= b) return a;
121     if (b >= a && b <= c || b >= c && b <= a) return b;
122     return c;
123 }
124
125 void fillArr(int arr[], int n) {
126     for(int i = 0; i < n; i++) {
127         arr[i] = rand() % 100;
128     }
129 }
130
131 void quickSort(int arr[], int a, int b) {
132     if(a < b) {
133         int pivot = arr[b];
134
135         int i = a - 1;
136         for(int j = a; j < b; j++)
137             if(arr[j] < pivot) {
138                 i++;
139                 swap(arr, i, j);
140             }
141         swap(arr, i + 1, b);
142
143         quickSort(arr, a, i);
144         quickSort(arr, i + 2, b);
145     }
146 }

```

Listing 1: main.cpp

## 2 Quicksort using pivot at the end vs. Quicksort using median pivot

Supprisingly, using median is actually 1.02 times slower than using the last element.

```
1 // Ryan Jacoby
2
3 /*
4 Done! (Intel Core-i5 8600k@4.6GHz)
5
6 --DEFAULT--
7 Min: 95133      Max: 102687      Avg: 95797
8
9 --Modified--
10 Min: 97079      Max: 103806      Avg: 97657
11
12
13 Done! (Intel Core-i9 9900k@5.00GHz)
14
15 --DEFAULT--
16 Min: 85851      Max: 86795      Avg: 86224
17
18 --Modified--
19 Min: 86840      Max: 87906      Avg: 87181
20 */
21
22 #include<iostream>
23 #include<ctime>
24 #include<chrono>
25
26 using namespace std;
27
28 void printArr(int[], int);
29 void swap(int[], int, int);
30 void quickSort(int[], int, int);
31 void quickSortMod(int[], int, int);
32 int med(int, int, int);
33 void fillArr(int[], int);
34
35 int main() {
36     srand(time(NULL));
37
38     int a[100000];
39
40     int q[100];
41     int qM[100];
42
43     for(int i = 0; i < 100; i++) {
44         fillArr(a, 100000);
45
46         auto start = chrono::high_resolution_clock::now();
47         quickSortMod(a, 0, 99999);
48         auto stop = chrono::high_resolution_clock::now();
49
50         qM[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
51
52         fillArr(a, 100000);
53     }
```

```

54     start = chrono::high_resolution_clock::now();
55     quickSort(a, 0, 99999);
56     stop = chrono::high_resolution_clock::now();
57
58     q[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
59
60     cout << i << "\r";
61
62 }
63
64
65 cout << "Done!\n";
66
67 cout << "\n--DEFAULT--\n";
68 int min = q[0], max = q[0], sum = 0;
69 for(int i = 0; i < 100; i++) {
70     if(q[i] > max) max = q[i];
71     if(q[i] < min) min = q[i];
72     sum += q[i];
73 }
74
75 cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';
76
77 cout << "\n--Modified--\n";
78 min = qM[0];
79 max = qM[0];
80 sum = 0;
81 for(int i = 0; i < 100; i++) {
82     if(qM[i] > max) max = qM[i];
83     if(qM[i] < min) min = qM[i];
84     sum += qM[i];
85 }
86
87 cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';
88
89 return 0;
90 }
91
92 void swap(int arr[], int a, int b) {
93     int temp = arr[a];
94     arr[a] = arr[b];
95     arr[b] = temp;
96 }
97
98 void printArr(int arr[], int n) {
99     for(int i = 0; i < n; i++)
100         cout << arr[i] << '\t';
101     cout << '\n';
102 }
103
104 int med(int a, int b, int c) {
105     if (a >= b && a <= c || a >= c && a <= b) return a;
106     if (b >= a && b <= c || b >= c && b <= a) return b;
107     return c;
108 }
109
110 void fillArr(int arr[], int n) {
111     for(int i = 0; i < n; i++) {
112         arr[i] = rand() % 100;

```

```

113     }
114 }
115
116 void quickSort(int arr[], int a, int b) {
117     if(a < b) {
118         int pivot = arr[b];
119
120         int i = a - 1;
121         for(int j = a; j < b; j++)
122             if(arr[j] < pivot) {
123                 i++;
124                 swap(arr, i, j);
125             }
126         swap(arr, i + 1, b);
127
128         quickSort(arr, a, i);
129         quickSort(arr, i + 2, b);
130     }
131 }
132
133 void quickSortMod(int arr[], int a, int b) {
134     if(a < b) {
135         int pivot;
136         if(b - a <= 7) pivot = (a + b) / 2;
137         else if(b - a <= 40) pivot = med(arr[a], arr[(a + b) / 2], arr[b]);
138         else med(med(arr[0 * (b - a - 1) / 8], arr[1 * (b - a - 1) / 8], arr[2 * (
139 b - a - 1) / 8]),
140 med(arr[3 * (b - a - 1) / 8], arr[4 * (b - a - 1) / 8], arr[5
141 * (b - a - 1) / 8]),
142 med(arr[6 * (b - a - 1) / 8], arr[7 * (b - a - 1) / 8], arr[8
143 * (b - a - 1) / 8]));
144
145         pivot = arr[b];
146
147         int i = a - 1;
148         for(int j = a; j < b; j++)
149             if(arr[j] < pivot) {
150                 i++;
151                 swap(arr, i, j);
152             }
153         swap(arr, i + 1, b);
154
155         quickSortMod(arr, a, i);
156         quickSortMod(arr, i + 2, b);
157     }
158 }

```

Listing 2: main.cpp

### 3 Overall comparison

Time( $\mu$ s)	Minimum	Maximum	Average
Quicksort	85304	91291	85691
Quicksort (Modified)	87103	92961	87492
Merge Sort	7586	8408	7606
Insertion Sort	4191381	4211034	4201999

```
1 // Ryan Jacoby
2
3 /*
4 I'm not running this on my PC, only my server..... it would take wayyyy too long.
5
6
7 Done! (Intel Core-i9 9900k@5.00GHz)
8
9 --Quicksort--
10 Min: 85304      Max: 91291      Avg: 85691
11
12 --Quicksort(Modified)--
13 Min: 87103      Max: 92961      Avg: 87492
14
15 --Merge Sort--
16 Min: 7586       Max: 8408       Avg: 7606
17
18 --Insertion Sort--
19 Min: 4191381    Max: 4211034    Avg: 4201999
20 */
21
22 #include<iostream>
23 #include<ctime>
24 #include<chrono>
25
26 using namespace std;
27
28 void printArr(int[], int);
29 void swap(int[], int, int);
30 void quickSort(int[], int, int);
31 void quickSortMod(int[], int, int);
32 int med(int, int, int);
33 void fillArr(int[], int);
34 void insertionSort(int[], int);
35 void mergeSort(int[], int, int);
36
37 int main() {
38     srand(time(NULL));
39
40     const int RUNS = 1000;
41
42     int a[100000];
43
44     int qS[RUNS];
45     int qM[RUNS];
46     int mS[RUNS];
47     int iS[RUNS];
48 }
```

```

49
50     for(int i = 0; i < RUNS; i++) {
51         // Quick Sort(Modified)
52         fillArr(a, 100000);
53
54         auto start = chrono::high_resolution_clock::now();
55         quickSortMod(a, 0, 99999);
56         auto stop = chrono::high_resolution_clock::now();
57
58         qM[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
59
60
61         // Quick Sort
62         fillArr(a, 100000);
63
64         start = chrono::high_resolution_clock::now();
65         quickSort(a, 0, 99999);
66         stop = chrono::high_resolution_clock::now();
67
68         qS[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
69
70
71         //Merge Sort
72         fillArr(a, 100000);
73
74         start = chrono::high_resolution_clock::now();
75         mergeSort(a, 0, 99999);
76         stop = chrono::high_resolution_clock::now();
77
78         mS[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
79
80
81         //Insertion Sort
82         fillArr(a, 100000);
83
84         start = chrono::high_resolution_clock::now();
85         insertionSort(a, 100000);
86         stop = chrono::high_resolution_clock::now();
87
88         iS[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();
89     }
90
91
92     cout << "Done!\n";
93
94     cout << "\n--Quicksort--\n";
95     int min = qS[0], max = qS[0], sum = 0;
96     for(int i = 0; i < RUNS; i++) {
97         if(qS[i] > max) max = qS[i];
98         if(qS[i] < min) min = qS[i];
99         sum += qS[i];
100     }
101
102     cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';
103
104     cout << "\n--Quicksort(Modified)--\n";
105     min = qM[0];
106     max = qM[0];
107     sum = 0;

```



```

108     for(int i = 0; i < RUNS; i++) {
109         if(qM[i] > max) max = qM[i];
110         if(qM[i] < min) min = qM[i];
111         sum += qM[i];
112     }
113
114     cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';
115
116     cout << "\n--Merge Sort--\n";
117     min = mS[0];
118     max = mS[0];
119     sum = 0;
120     for(int i = 0; i < RUNS; i++) {
121         if(mS[i] > max) max = mS[i];
122         if(mS[i] < min) min = mS[i];
123         sum += mS[i];
124     }
125
126     cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';
127
128     cout << "\n--Insertion Sort--\n";
129     min = iS[0];
130     max = iS[0];
131     sum = 0;
132     for(int i = 0; i < RUNS; i++) {
133         if(iS[i] > max) max = iS[i];
134         if(iS[i] < min) min = iS[i];
135         sum += iS[i];
136     }
137
138     cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';
139
140     return 0;
141 }
142
143 void swap(int arr[], int a, int b) {
144     int temp = arr[a];
145     arr[a] = arr[b];
146     arr[b] = temp;
147 }
148
149 void printArr(int arr[], int n) {
150     for(int i = 0; i < n; i++)
151         cout << arr[i] << '\t';
152     cout << '\n';
153 }
154
155 int med(int a, int b, int c) {
156     if (a >= b && a <= c || a >= c && a <= b) return a;
157     if (b >= a && b <= c || b >= c && b <= a) return b;
158     return c;
159 }
160
161 void fillArr(int arr[], int n) {
162     for(int i = 0; i < n; i++) {
163         arr[i] = rand() % 100;
164     }
165 }
166

```

```

167 void quickSort(int arr[], int a, int b) {
168     if(a < b) {
169         int pivot = arr[b];
170
171         int i = a - 1;
172         for(int j = a; j < b; j++)
173             if(arr[j] < pivot) {
174                 i++;
175                 swap(arr, i, j);
176             }
177         swap(arr, i + 1, b);
178
179         quickSort(arr, a, i);
180         quickSort(arr, i + 2, b);
181     }
182 }
183
184 void quickSortMod(int arr[], int a, int b) {
185     if(a < b) {
186         int pivot;
187         if(b - a <= 7) pivot = (a + b) / 2;
188         else if(b - a <= 40) pivot = med(arr[a], arr[(a + b) / 2], arr[b]);
189         else med(med(arr[0 * (b - a - 1) / 8], arr[1 * (b - a - 1) / 8], arr[2 * (
190             b - a - 1) / 8]),
191             med(arr[3 * (b - a - 1) / 8], arr[4 * (b - a - 1) / 8], arr[5
192             * (b - a - 1) / 8]),
193             med(arr[6 * (b - a - 1) / 8], arr[7 * (b - a - 1) / 8], arr[8
194             * (b - a - 1) / 8]));
195
196         pivot = arr[b];
197
198         int i = a - 1;
199         for(int j = a; j < b; j++)
200             if(arr[j] < pivot) {
201                 i++;
202                 swap(arr, i, j);
203             }
204         swap(arr, i + 1, b);
205
206         quickSortMod(arr, a, i);
207         quickSortMod(arr, i + 2, b);
208     }
209 }
210
211 void insertionSort(int arr[], int n) {
212     int i, j, k;
213
214     for(int i = 1; i < n; i++) {
215         k = arr[i];
216         j = i - 1;
217
218         while(j >= 0 && arr[j] > k) {
219             arr[j + 1] = arr[j];
220             j = j - 1;
221         }
222         arr[j + 1] = k;
223     }
224 }

```

```

223
224 void mergeSort(int arr[], int a, int b) {
225     if(a < b) {
226         int m = a + (b - a) / 2;
227         int i, j, k;
228
229         mergeSort(arr, a, m);
230         mergeSort(arr, m + 1, b);
231
232         int n1 = m - a + 1, n2 = b - m;
233         int l[n1], r[n2];
234
235         for(i = 0; i < n1; i++) l[i] = arr[a + i];
236         for(j = 0; j < n2; j++) r[j] = arr[m + 1 + j];
237
238         i = 0;
239         j = 0;
240         k = a;
241         while(i < n1 && j < n2) {
242             if(l[i] <= r[j]) {
243                 arr[k] = l[i];
244                 i++;
245             } else {
246                 arr[k] = r[j];
247                 j++;
248             }
249             k++;
250         }
251
252         while(i < n1) {
253             arr[k] = l[i];
254             i++;
255             k++;
256         }
257
258         while(j < n2) {
259             arr[k] = r[j];
260             j++;
261             k++;
262         }
263     }
264 }

```

Listing 3: main.cpp