COMSC-200
Lab 4

Ryan Jacoby

20 September 2020

1 Quicksort vs. Select Median

Median Select is around 28 times faster than Quicksort.

```
// Ryan Jacoby

/*
Done! (Intel Core-i5 8600k@4.6GHz)

--Sort + Middle--
Min: 95149      Max: 96755      Avg: 95649

--Mean--
Min: 1675       Max: 5597       Avg: 3467


Done! (Intel Core-i9 9900k@5.00GHz)

--Sort + Middle--
Min: 85769      Max: 90052      Avg: 86201

--Mean--
Min: 1583       Max: 4988       Avg: 3221
*/

#include<iostream>
#include<ctime>
#include<chrono>

using namespace std;

void printArr(int[], int);
void swap(int[], int, int);
int select(int[], int, int, int);
void quickSort(int[], int, int);
int med(int, int, int);
void fillArr(int[], int);

int main() {
    srand(time(NULL));

    int a[100000];

    int s[100];
```

```
    int q[100];

    for(int i = 0; i < 100; i++) {
        fillArr(a, 100000);

        auto start = chrono::high_resolution_clock::now();
        select(a, 100000 / 2, 0, 100000);
        auto stop = chrono::high_resolution_clock::now();

        s[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();

        fillArr(a, 100000);

        start = chrono::high_resolution_clock::now();
        quickSort(a, 0, 99999);
        stop = chrono::high_resolution_clock::now();

        q[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();

        cout << i << "\r";

    }


    cout << "Done!\n";

    cout << "\n--Sort + Middle--\n";
    int min = q[0], max = q[0], sum = 0;
    for(int i = 0; i < 100; i++) {
        if(q[i] > max) max = q[i];
        if(q[i] < min) min = q[i];
        sum += q[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';

    cout << "\n--Mean--\n";
    min = s[0];
    max = s[0];
    sum = 0;
    for(int i = 0; i < 100; i++) {
        if(s[i] > max) max = s[i];
        if(s[i] < min) min = s[i];
        sum += s[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';

    return 0;
}

void swap(int arr[], int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}

void printArr(int arr[], int n) {
    for(int i = 0; i < n; i++)
```

```
        cout << arr[i] << '\t';
    cout << '\n';
}

int select(int arr[], int k, int a, int b) {
    int pivot = arr[b];
    int i = a;
    for(int j = a; j < b; j++)
        if(arr[j] <= pivot) {
            swap(arr, i, j);
            i++;
        }
    swap(arr, i, b);

    if(i - a == k) return arr[i];
    if(i - a > k) return select(arr, k, a, i - 1);
    return select(arr, k - i + a - 1, i + 1, b);
}

int med(int a, int b, int c) {
    if (a >= b && a <= c || a >= c && a <= b) return a;
    if (b >= a && b <= c || b >= c && b <= a) return b;
    return c;
}

void fillArr(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        arr[i] = rand() % 100;
    }
}

void quickSort(int arr[], int a, int b) {
    if(a < b) {
        int pivot = arr[b];

        int i = a - 1;
        for(int j = a; j < b; j++)
            if(arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        swap(arr, i + 1, b);

        quickSort(arr, a, i);
        quickSort(arr, i + 2, b);
    }
}
```

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.

```
// Ryan Jacoby

/*
Done! (Intel Core-i5 8600k@4.6GHz)

--DEFAULT--
Min: 95133      Max: 102687     Avg: 95797

--Modified--
Min: 97079      Max: 103806     Avg: 97657


Done! (Intel Core-i9 9900k@5.00GHz)

--DEFAULT--
Min: 85851      Max: 86795      Avg: 86224

--Modified--
Min: 86840      Max: 87906      Avg: 87181
*/

#include<iostream>
#include<ctime>
#include<chrono>

using namespace std;

void printArr(int[], int);
void swap(int[], int, int);
void quickSort(int[], int, int);
void quickSortMod(int[], int, int);
int med(int, int, int);
void fillArr(int[], int);

int main() {
    srand(time(NULL));

    int a[100000];

    int q[100];
    int qM[100];

    for(int i = 0; i < 100; i++) {
        fillArr(a, 100000);

        auto start = chrono::high_resolution_clock::now();
        quickSortMod(a, 0, 99999);
        auto stop = chrono::high_resolution_clock::now();

        qM[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();

        fillArr(a, 100000);

```

```
        start = chrono::high_resolution_clock::now();
        quickSort(a, 0, 99999);
        stop = chrono::high_resolution_clock::now();

        q[i] = chrono::duration_cast<chrono::microseconds>(stop - start).count();

        cout << i << "\r";

    }


    cout << "Done!\n";

    cout << "\n--DEFAULT--\n";
    int min = q[0], max = q[0], sum = 0;
    for(int i = 0; i < 100; i++) {
        if(q[i] > max) max = q[i];
        if(q[i] < min) min = q[i];
        sum += q[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';

    cout << "\n--Modified--\n";
    min = qM[0];
    max = qM[0];
    sum = 0;
    for(int i = 0; i < 100; i++) {
        if(qM[i] > max) max = qM[i];
        if(qM[i] < min) min = qM[i];
        sum += qM[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / 100 << '\n';

    return 0;
}

void swap(int arr[], int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}

void printArr(int arr[], int n) {
    for(int i = 0; i < n; i++)
        cout << arr[i] << '\t';
    cout << '\n';
}

int med(int a, int b, int c) {
    if (a >= b && a <= c || a >= c && a <= b) return a;
    if (b >= a && b <= c || b >= c && b <= a) return b;
    return c;
}

void fillArr(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        arr[i] = rand() % 100;
```

```
    }
}

void quickSort(int arr[], int a, int b) {
    if(a < b) {
        int pivot = arr[b];

        int i = a - 1;
        for(int j = a; j < b; j++)
            if(arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        swap(arr, i + 1, b);

        quickSort(arr, a, i);
        quickSort(arr, i + 2, b);
    }
}

void quickSortMod(int arr[], int a, int b) {
    if(a < b) {
        int pivot;
        if(b - a <= 7) pivot = (a + b) / 2;
        else if(b - a <= 40) pivot = med(arr[a], arr[(a + b) / 2], arr[b]);
        else med(med(arr[0 * (b - a - 1) / 8], arr[1 * (b - a - 1) / 8], arr[2 * (b - a - 1) / 8]),
                 med(arr[3 * (b - a - 1) / 8], arr[4 * (b - a - 1) / 8], arr[5 * (b - a - 1) / 8]),
                 med(arr[6 * (b - a - 1) / 8], arr[7 * (b - a - 1) / 8], arr[8 * (b - a - 1) / 8]));

        pivot = arr[b];

        int i = a - 1;
        for(int j = a; j < b; j++)
            if(arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        swap(arr, i + 1, b);

        quickSortMod(arr, a, i);
        quickSortMod(arr, i + 2, b);
    }
}
```

Listing 2: main.cpp

3 Overall comparison

Time(μs)

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

```
// Ryan Jacoby

/*
I'm not running this on my PC, only my server..... it would take wayyyy too long.


Done! (Intel Core-i9 9900k@5.00GHz)

--Quicksort--
Min: 85304      Max: 91291      Avg: 85691

--Quicksort(Modified)--
Min: 87103      Max: 92961      Avg: 87492

--Merge Sort--
Min: 7586       Max: 8408       Avg: 7606

--Insertion Sort--
Min: 4191381    Max: 4211034    Avg: 4201999
*/

#include<iostream>
#include<ctime>
#include<chrono>

using namespace std;

void printArr(int[], int);
void swap(int[], int, int);
void quickSort(int[], int, int);
void quickSortMod(int[], int, int);
int med(int, int, int);
void fillArr(int[], int);
void insertionSort(int[], int);
void mergeSort(int[], int, int);

int main() {
    srand(time(NULL));

    const int RUNS = 1000;

    int a[100000];

    int qS[RUNS];
    int qM[RUNS];
    int mS[RUNS];
    int iS[RUNS];

```

```
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;
```

```
    for(int i = 0; i < RUNS; i++) {
        if(qM[i] > max) max = qM[i];
        if(qM[i] < min) min = qM[i];
        sum += qM[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';

    cout << "\n--Merge Sort--\n";
    min = mS[0];
    max = mS[0];
    sum = 0;
    for(int i = 0; i < RUNS; i++) {
        if(mS[i] > max) max = mS[i];
        if(mS[i] < min) min = mS[i];
        sum += mS[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';

    cout << "\n--Insertion Sort--\n";
    min = iS[0];
    max = iS[0];
    sum = 0;
    for(int i = 0; i < RUNS; i++) {
        if(iS[i] > max) max = iS[i];
        if(iS[i] < min) min = iS[i];
        sum += iS[i];
    }

    cout << "Min: " << min << "\tMax: " << max << "\tAvg: " << sum / RUNS << '\n';

    return 0;
}

void swap(int arr[], int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}

void printArr(int arr[], int n) {
    for(int i = 0; i < n; i++)
        cout << arr[i] << '\t';
    cout << '\n';
}

int med(int a, int b, int c) {
    if (a >= b && a <= c || a >= c && a <= b) return a;
    if (b >= a && b <= c || b >= c && b <= a) return b;
    return c;
}

void fillArr(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        arr[i] = rand() % 100;
    }
}
```

```
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 * (
    b - a - 1) / 8]),
190                  med(arr[3 * (b - a - 1) / 8], arr[4 * (b - a - 1) / 8], arr[5
    * (b - a - 1) / 8]),
191                  med(arr[6 * (b - a - 1) / 8], arr[7 * (b - a - 1) / 8], arr[8
    * (b - a - 1) / 8]));
192 
193         pivot = arr[b];
194 
195         int i = a - 1;
196         for(int j = a; j < b; j++)
197             if(arr[j] < pivot) {
198                 i++;
199                 swap(arr, i, j);
200             }
201         swap(arr, i + 1, b);
202 
203         quickSortMod(arr, a, i);
204         quickSortMod(arr, i + 2, b);
205     }
206 }
207 
208 void insertionSort(int arr[], int n) {
209     int i, j, k;
210 
211     for(int i = 1; i < n; i++) {
212         k = arr[i];
213         j = i - 1;
214 
215         while(j >= 0 && arr[j] > k) {
216             arr[j + 1] = arr[j];
217             j = j - 1;
218         }
219 
220         arr[j + 1] = k;
221     }
222 }
```

```
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[i]) {
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