Lab2

Sheharyar Alam Khan

February 23, 2019

Contents

1 partition.cpp

Lab2 partition

The Purpose

The purpose of this lab is to make a quick sort function. The quick sort function uses divide and conquer to sort the array. its faster and is known is the most efficient way to sort a large list of data. merge sort also uses divide and conquer and is very efficient but it uses more memory as it makes a new sorted array. In quick sort we sort the array itself using swap.

```
#include <iostream>
#include "catch.hpp"
```

Partition

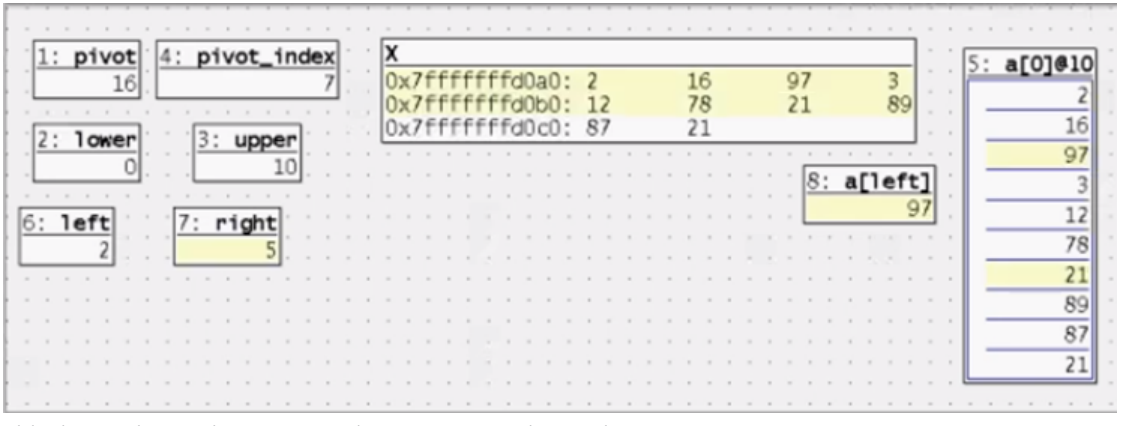
In this function we want to partition the array in a way so that all the numbers less than the pivot are on the left and all the values greater than the pivot are on the right side of the pivot. If the value on the right is greater than the pivot then its swapped with the value on the left.

```
int partition(int a[], int lower, int upper, int pivot_index)
{
        int pivot = a[pivot_index]; int left = lower; int right = upper-2;
        //std::swap(a[pivot_index], a[upper - 1]);
        while(left <= right)
        {
                if(a[left] <= pivot)
                        left++;
                else {
                        std::swap(a[left],a[right]);
                        right--;
                        }
    }
```

```
    std::swap(a[left],a[upper - 1]);
        return left;
}

Select : 0.000683342 seconds
Sort : 0.000461662 seconds

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
select is a Catch v2.0.1 host application.
Run with -? for options

-------------------------------------------------------------------------------
Partition
-------------------------------------------------------------------------------
partition.cpp:32
...............................................................................

partition.cpp:41:
PASSED:
  CHECK( a[i] == b[i] )
with expansion:
  2 == 2

partition.cpp:41: FAILED:
  CHECK( a[i] == b[i] )
with expansion:
  16 == 12

partition.cpp:41: FAILED:
  CHECK( a[i] == b[i] )
with expansion:
  12 == 3
```

```
partition.cpp:41: FAILED:
  CHECK( a[i] == b[i] )
with expansion:
  3 == 16

partition.cpp:41: FAILED:
  CHECK( a[i] == b[i] )
with expansion:
  21 == 78

partition.cpp:41:
PASSED:
  CHECK( a[i] == b[i] )
with expansion:
  97 == 97

partition.cpp:41:
PASSED:
  CHECK( a[i] == b[i] )
with expansion:
  21 == 21

partition.cpp:41:
PASSED:
  CHECK( a[i] == b[i] )
with expansion:
  89 == 89

partition.cpp:41:
PASSED:
  CHECK( a[i] == b[i] )
```

```
with expansion:
  87 == 87

partition.cpp:41: FAILED:
  CHECK( a[i] == b[i] )
with expansion:
  78 == 21

===============================================================================
test cases:  2 | 1 passed | 1 failed
assertions: 10 | 5 passed | 5 failed
```



//~/Labs/Lab2/partition$ g++ -g -I/home/debian -o "select" "select.cpp" "partition.cpp" ~/catch2.o -DUNIT_TEST

2 select.cpp

```
// cs124 lab
#include <iostream>
#include <algorithm>
#include <stdlib.h>
#include <fstream>
#include "catch.hpp"
#include <chrono>


using namespace std;

int partition(int arr[], int lower, int upper, int pivot_index);

select
    This is a recursive function that finds the median value of the array.

int select(int arr[], int pivot_index, int lower, int upper)
{
        int p = partition(arr, lower, upper, pivot_index);
        int n1 = p - lower;
        int n2 = 1;// this wastes memory. I suggest hardcoding 1 instead.
        int n3 = upper - (n1 + n2);// I feel that this line has no purpose.
            it also wastes memory.
        if (pivot_index < n1)
        {
                select (arr, pivot_index, 0, n1);//recursive case
        }
        else if (pivot_index > n1 + n2)
        {
                select(arr, pivot_index, n1 + n2, upper);//recursive case
```

```
        }
        else
        {
                return (arr[p]);//base case
        }
}
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
select is a Catch v2.0.1 host application.
Run with -? for options

-------------------------------------------------------------------------------
Select
-------------------------------------------------------------------------------
select.cpp:42
...............................................................................

select.cpp:46:
PASSED:
  REQUIRE( select(a,n/2,0,n) == 21 )
with expansion:
  21 == 21

select.cpp:48:
PASSED:
  REQUIRE( a[n/2] == 21 )
with expansion:
  21 == 21

===============================================================================
All tests passed (2 assertions in 1 test case)
```

Reading input from file

In order to read the input from a text file i used ifstream and just stored everything in an array of size n (in this case 1000). I'm using a large value so we can see the diference in times between our sort function and the std libraries sort function.

measuring time

To measure the time I used chronos::high_resolution_clock and increased the number of items in the array until it took long enough for a proper time output.

```
TEST_CASE("difference between select and sort")
{
        const int n = 1000;

        //int a[] = {2,87,21,3,12,78,97,16,89,21};
        int a[n];

        ifstream file;
        file.open("shuffle.txt");

        for(int i = 0; i < n; i++)
        {
                file >> a[i];
        }

        chrono::high_resolution_clock::time_point t1 = chrono::
            high_resolution_clock::now();
        select(a, n / 2, 0, n - 1);
        chrono::high_resolution_clock::time_point t2 = chrono::
            high_resolution_clock::now();
        chrono::duration<double> time_span1 = chrono::duration_cast<chrono::
            duration<double>>(t2 - t1);
```

```
        t1 = chrono::high_resolution_clock::now();
        std::sort(a,a+n);
        t2 = chrono::high_resolution_clock::now();
        chrono::duration<double> time_span2 = chrono::duration_cast<chrono::
            duration<double>>(t2 - t1);

        cout << "Select : " << time_span1.count() << " seconds" << endl;
        cout << "Sort : " << time_span2.count() << " seconds" << endl;
}
```

```
Select : 0.23898 seconds
Sort : 0.00959879 seconds
===============================================================================
test cases: 1 | 1 passed
assertions: - none -
```

3 shuffle.txt

```
787
495
708
388
721
639
602
849
81
903
230
301
393
414
338
760
356
188
208
445
776
407
703
51
542
257
585
928
194
769
```

```
736
434
823
945
29
85
615
198
513
696
632
766
217
352
236
185
2
366
314
999
663
438
837
381
32
726
665
412
421
325
674
582
```

844
281
240
229
345
749
225
671
752
858
988
376
41
436
764
967
866
317
990
481
605
506
870
879
863
750
406
93
819
968
119
504

177
633
628
836
134
38
715
607
300
691
369
658
56
653
454
917
189
526
514
834
620
857
756
591
343
909
538
916
242
26
694
477

411
235
404
873
827
359
706
636
508
96
252
133
264
815
678
157
997
270
974
190
461
272
920
789
871
918
423
267
145
214
771
54

531
215
448
930
652
241
239
35
399
167
168
166
429
587
921
970
521
548
247
286
28
468
557
115
363
944
560
644
55
725
908
216

```
291
294
48
575
332
897
59
79
573
220
927
793
811
712
152
49
316
111
192
395
433
159
169
647
664
262
751
520
966
92
204
572
```

878
307
511
794
629
123
805
861
846
278
900
635
83
5
829
973
199
420
69
491
357
551
798
932
925
202
117
998
773
681
583
350

670
277
540
772
334
765
701
740
246
493
590
956
408
258
668
661
579
16
626
564
622
991
759
237
274
523
282
995
403
941
955
213

66
534
463
596
631
731
120
906
72
856
89
273
698
64
358
114
654
306
899
486
402
768
326
960
324
173
375
553
311
10
544
558

```
124
707
692
940
876
99
418
606
182
676
170
812
197
690
256
431
122
938
23
987
935
975
841
788
71
4
65
435
148
8
882
601
```

643
592
379
922
384
344
957
729
527
385
466
46
984
803
425
370
510
102
361
816
528
179
685
782
924
308
755
360
516
75
295
82

623
87
686
14
176
853
47
589
283
546
797
537
848
260
497
78
125
898
742
578
382
276
143
25
462
347
141
996
84
18
950
738

```
27
165
825
351
962
599
614
126
180
838
73
144
611
392
634
401
890
840
255
705
130
455
979
624
593
804
600
318
424
374
373
162
```

```
512
651
136
588
389
346
464
13
320
947
91
484
7
147
289
550
472
467
340
682
377
584
453
475
617
108
164
444
405
655
994
3
```

```
895
439
852
292
153
413
687
576
919
743
734
965
847
581
478
826
106
243
319
865
977
362
76
724
498
554
21
163
929
754
720
981
```

```
943
113
186
901
218
684
763
222
17
561
160
58
779
717
627
982
397
287
741
353
88
313
699
383
828
138
98
910
509
140
154
679
```

645
1000
205
371
137
774
904
809
980
872
303
936
503
285
184
135
800
275
335
612
53
447
820
608
336
331
610
757
174
718
211
621

233
482
801
854
201
6
912
835
490
251
672
875
604
594
39
939
887
450
874
963
244
641
598
640
539
723
821
911
44
253
959
442

```
80
34
571
613
305
700
296
568
868
867
688
422
791
417
105
562
430
97
953
913
249
618
881
728
451
232
563
693
949
761
515
877
```

785
61
677
67
704
473
367
518
488
465
896
471
103
100
284
747
889
843
443
986
737
952
597
958
893
880
711
775
946
719
206
95

```
50
833
268
533
476
638
109
487
36
68
310
883
446
386
94
948
869
432
263
714
683
814
552
321
271
200
485
505
565
926
961
391
```

297
529
795
839
862
666
799
625
269
786
845
322
931
470
142
221
428
193
452
907
851
156
993
77
630
496
337
333
60
171
992
777

501
656
559
722
42
502
507
480
339
713
355
22
746
288
309
673
328
298
545
90
238
312
603
195
341
500
989
394
884
415
43
19

```
702
780
37
762
923
426
802
543
648
646
810
850
808
132
378
209
20
372
15
860
951
580
790
416
902
348
380
536
745
441
530
657
```

```
161
207
937
859
474
280
971
299
888
796
659
266
12
818
976
778
695
499
753
733
650
400
293
524
329
479
45
494
830
954
279
675
```

158
748
535
449
33
390
732
570
983
522
121
315
304
196
469
396
203
457
914
985
616
261
595
569
933
619
697
226
770
680
155
822

964
934
525
290
541
574
427
74
172
758
52
689
150
716
327
458
855
744
784
118
915
259
969
40
178
9
86
577
817
519
116
824

183
387
972
365
566
212
437
24
609
342
127
245
181
398
807
832
250
886
30
669
231
460
323
410
842
440
456
532
492
62
302
709

128
409
556
1
31
891
101
667
735
885
864
978
727
227
894
459
792
224
234
146
330
489
151
228
662
223
567
219
586
139
70
549

248
187
710
57
649
783
354
555
107
63
104
730
942
149
191
767
364
368
831
806
419
781
483
637
265
517
112
349
175
660
110
892

```
131
739
210
813
905
547
11
129
254
642
```