HW [makeup], [spring/2019]MCS 253P

Name: Yu Qin

Table of Contents

==========Results Screenshots==========
==========start of the write-up==========

Lab 8.3: Max Avg Subtree

Code: mainly by divide and conquer:

```
26 ▼ struct Result FindMaxAverageSubtree(int indexi){
27       // renew sumbinarytree, countbinarytree
28       ////cout<<"processing #"<<indexi<<endl;
29       // divide:
30       int lchild = LeftChild(indexi + 1 );
31       int rchild = RightChild(indexi + 1);
32       struct Result lresult, rresult, result;
33
34       if (lchild) {
35           lresult = FindMaxAverageSubtree(lchild);
36 ▼     } else {
37           lresult.sum = 0;
38           lresult.count = 0;
39       }
40
41       if (rchild) {
42           rresult = FindMaxAverageSubtree(rchild);
43 ▼     } else {
44           rresult.sum = 0;
45           rresult.count = 0;
46       }
47
48       // conquer:
49       result.sum = binarytree[indexi] + lresult.sum + rresult.sum;
50       result.count = 1 + lresult.count + rresult.count;
51 ▼     if (resultmaxaverage < result.sum * 1.0 / result.count) {
52           resultmaxaverage = result.sum * 1.0 / result.count;
53           resultindex = indexi;
54           ////cout<<"changed at #"<<indexi<<": "<<binarytree[indexi]<<" with sum = "<<result.sum<<" and count = "<<result.
55       }
56       ////cout<<"return #"<<indexi<<endl;
57       return result;
58   }
59 ▼ void ConstructTree(string filename) {
```

test case 1, 2 , 3:
1:

```
$ valgrind ./main 1test.txt
==5405== Memcheck, a memory error detector
==5405== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5405== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5405== Command: ./main 1test.txt
==5405==
input number: 7
0: 1
1: -5
2: 11
3: 1
4: 2
5: 4
6: -3
maximum_average_subtree is at node #2: value 11 with average of 4
==5405==
==5405== HEAP SUMMARY:
==5405==     in use at exit: 72,704 bytes in 1 blocks
==5405==   total heap usage: 7 allocs, 6 frees, 81,524 bytes allocated
==5405==
==5405== LEAK SUMMARY:
==5405==    definitely lost: 0 bytes in 0 blocks
==5405==    indirectly lost: 0 bytes in 0 blocks
==5405==      possibly lost: 0 bytes in 0 blocks
==5405==    still reachable: 72,704 bytes in 1 blocks
==5405==         suppressed: 0 bytes in 0 blocks
==5405== Rerun with --leak-check=full to see details of leaked memory
==5405==
==5405== For counts of detected and suppressed errors, rerun with: -v
==5405== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) yuq8@andromeda-21 12:44:39 ~/253P/hw_lab/Makeup/8.3_MaxAvgSubtree
$
```

2:

```
$ valgrind ./main input2.txt
==5437== Memcheck, a memory error detector
==5437== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5437== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5437== Command: ./main input2.txt
==5437==
input number: 16
0: 1
1: 2
2: 3
3: 4
4: 5
5: 6
6: 7
7: 8
8: 9
9: 10
10: 11
11: 12
12: 13
13: 14
14: 15
15: 16
maximum_average_subtree is at node #15: value 16 with average of 16
==5437==
==5437== HEAP SUMMARY:
==5437==     in use at exit: 72,704 bytes in 1 blocks
==5437==   total heap usage: 8 allocs, 7 frees, 81,588 bytes allocated
==5437==
==5437== LEAK SUMMARY:
==5437==    definitely lost: 0 bytes in 0 blocks
==5437==    indirectly lost: 0 bytes in 0 blocks
==5437==      possibly lost: 0 bytes in 0 blocks
==5437==    still reachable: 72,704 bytes in 1 blocks
==5437==         suppressed: 0 bytes in 0 blocks
==5437== Rerun with --leak-check=full to see details of leaked memory
==5437==
==5437== For counts of detected and suppressed errors, rerun with: -v
==5437== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) yuq8@andromeda-21 12:45:16 ~/253P/hw_lab/Makeup/8.3_MaxAvgSubtree
$
```

3:

```
$ valgrind ./main input3.txt
==5495== Memcheck, a memory error detector
==5495== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5495== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5495== Command: ./main input3.txt
==5495==
input number: 15
0: 50
1: 40
2: 40
3: 30
4: 30
5: 30
6: 30
7: 20
8: 20
9: 20
10: 20
11: 20
12: 20
13: 20
14: 20
maximum_average_subtree is at node #0: value 50 with average of 27.3333
==5495==
==5495== HEAP SUMMARY:
==5495==     in use at exit: 72,704 bytes in 1 blocks
==5495==   total heap usage: 8 allocs, 7 frees, 81,588 bytes allocated
==5495==
==5495== LEAK SUMMARY:
==5495==    definitely lost: 0 bytes in 0 blocks
==5495==    indirectly lost: 0 bytes in 0 blocks
==5495==      possibly lost: 0 bytes in 0 blocks
==5495==    still reachable: 72,704 bytes in 1 blocks
==5495==         suppressed: 0 bytes in 0 blocks
==5495== Rerun with --leak-check=full to see details of leaked memory
==5495==
==5495== For counts of detected and suppressed errors, rerun with: -v
==5495== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) yuq8@andromeda-21 12:46:20 ~/253P/hw_lab/Makeup/8.3_MaxAvgSubtree
$
```

Conclusion: right result, 0 error in memory leak.
Little notes:

1. If encounter with 2 equal, prefers to the lower level node

Lab 9.3: savingandloading_bt

```
$ cd 9.3_savingandloading_bt/
(base) yuq8@andromeda-21 12:49:16 ~/253P/hw_lab/Makeup/9.3_savingandloading_bt
$ ls
.DS_Store  load1.txt  load2.txt  load3.txt  load.txt  main*  main.cpp  main.dSYM/  Makefile  save1.txt  save2.txt  save3.txt
(base) yuq8@andromeda-21 12:49:16 ~/253P/hw_lab/Makeup/9.3_savingandloading_bt
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main load.txt save.txt
(base) yuq8@andromeda-21 12:49:19 ~/253P/hw_lab/Makeup/9.3_savingandloading_bt
$
```

Test cases 1, 2, 3:
1:

```
$ valgrind ./main load.txt save.txt
==5797== Memcheck, a memory error detector
==5797== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5797== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5797== Command: ./main load.txt save.txt
==5797==
Tree loaded
                7
        3
                        13
                6
                        12
1
                        11
                5
                        10
        2
                        9
                4
                        8
Tree saved after modification
                70
        30
                        130
                60
                        120
10
                        110
                50
                        100
        20
                        90
                40
                        80
==5797==
==5797== HEAP SUMMARY:
==5797==     in use at exit: 72,704 bytes in 1 blocks
==5797==   total heap usage: 36 allocs, 35 frees, 91,700 bytes allocated
==5797==
==5797== LEAK SUMMARY:
==5797==    definitely lost: 0 bytes in 0 blocks
==5797==    indirectly lost: 0 bytes in 0 blocks
==5797==      possibly lost: 0 bytes in 0 blocks
==5797==    still reachable: 72,704 bytes in 1 blocks
==5797==         suppressed: 0 bytes in 0 blocks
==5797== Rerun with --leak-check=full to see details of leaked memory
==5797==
==5797== For counts of detected and suppressed errors, rerun with: -v
==5797== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) yuq8@andromeda-21 12:50:47 ~/253P/hw_lab/Makeup/9.3_savingandloading_bt
$
```

2:

```
$ valgrind ./main load2.txt save2.txt
==5863== Memcheck, a memory error detector
==5863== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5863== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5863== Command: ./main load2.txt save2.txt
==5863==
Tree loaded
                        6
                5
                        null
        4
                        null
                null
                        null
3
                        null
                null
                        null
        null
                        null
                null
                        null
Tree saved after modification
                        60
                50
                        null
        40
                        null
                null
                        null
30
                        null
                null
                        null
        null
                        null
                null
                        null
==5863==
==5863== HEAP SUMMARY:
==5863==     in use at exit: 72,704 bytes in 1 blocks
==5863==   total heap usage: 40 allocs, 39 frees, 92,148 bytes allocated
==5863==
==5863== LEAK SUMMARY:
==5863==    definitely lost: 0 bytes in 0 blocks
==5863==    indirectly lost: 0 bytes in 0 blocks
==5863==      possibly lost: 0 bytes in 0 blocks
==5863==    still reachable: 72,704 bytes in 1 blocks
==5863==         suppressed: 0 bytes in 0 blocks
==5863== Rerun with --leak-check=full to see details of leaked memory
==5863==
==5863== For counts of detected and suppressed errors, rerun with: -v
==5863== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

3:

```
$ valgrind ./main load3.txt save3.txt
==5931== Memcheck, a memory error detector
==5931== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5931== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5931== Command: ./main load3.txt save3.txt
==5931==
Tree loaded
0
Tree saved after modification
0
==5931==
==5931== HEAP SUMMARY:
==5931==     in use at exit: 72,704 bytes in 1 blocks
==5931==   total heap usage: 8 allocs, 7 frees, 90,236 bytes allocated
==5931==
==5931== LEAK SUMMARY:
==5931==    definitely lost: 0 bytes in 0 blocks
==5931==    indirectly lost: 0 bytes in 0 blocks
==5931==      possibly lost: 0 bytes in 0 blocks
==5931==    still reachable: 72,704 bytes in 1 blocks
==5931==         suppressed: 0 bytes in 0 blocks
==5931== Rerun with --leak-check=full to see details of leaked memory
==5931==
==5931== For counts of detected and suppressed errors, rerun with: -v
==5931== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) yuq8@andromeda-21 12:53:08 ~/253P/hw_lab/Makeup/9.3_savingandloading_bt
$
```

Conclusion: right result, 0 error in memory leak.

Previous Quiz: Prime Factorization

Dear TA,

I have studied and re-wrote a file of code. It would be nice if you decide it's enough for some make-up of the quiz.

```
10  int a[5] = {2,3,5,7,11};
11  vector<int> primeset(a, a + 5);
12  bool isprime(int value ){
13      for (int i = 2; i <= sqrt(value); i++){
14          if (value % i == 0)
15              return false;
16      }
17      return true;
18  }
19  void constructPrimeSet(int num){
20      int size = primeset.size();
21      int i = primeset[size - 1] + 1;
22      do {
23          if ( isprime(i) )
24              primeset.push_back(i);
25          i++;
26      }while(i <= num);
27  }
28  void checkPrimeSet(int num){
29      int size = primeset.size();
30      if (primeset[size - 1] < num)
31          constructPrimeSet(num);
32  }
33  void printPrime(int num){
34      if (num <= -1) {
35          cout << num <<": no prime.\n";
36      }
37      //cout << num <<": has prime of: "<<endl;
38      cout << num <<": has prime of: ";
39      for (int i = 0; primeset[i] <= sqrt(num); i++){
40          //cout<<"i, prime: "<< i << ","<<primeset[i]<<endl;
41          while (num % primeset[i] == 0){
42              if (num/primeset[i] == 1)
43                  printf("%d\n",primeset[i]);
44              else
45                  printf("%d * ",primeset[i]);
46              //cout<<"num, prime: "<< num << ","<<primeset[i]<<endl;
47              num /= primeset[i];
48              //cout<<"new num "<< num << endl;
49          }
50      }
51      if (num > 2)
52          printf("%d\n", num);
53  }
54  int main(){
55      ifstream in("input.txt");
56      int num;
57      string mynum;
58      //cout <<"initial test: " << primeset[6]<<endl;
59      while (getline(in, mynum)){
60          stringstream(mynum) >> num;
61          //cout<<num<<endl;
62          checkPrimeSet(num);
63          //cout << "constructPrimeSet for " << num << " until " << primeset.back() << endl;
64          printPrime(num);
65      }
66      in.close();
67      return 0;
```

test cases 1, 2, 3, 4:

```
(base) yuq8@andromeda-17 16:35:06 ~/253P/hw_lab/Makeup/quiz_4_prime
$ valgrind ./main
==22824== Memcheck, a memory error detector
==22824== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==22824== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==22824== Command: ./main
==22824==
46328: has prime of: 2 * 2 * 2 * 5791
45: has prime of: 3 * 3 * 5
16: has prime of: 2 * 2 * 2 * 2
99: has prime of: 3 * 3 * 11
==22824==
==22824== HEAP SUMMARY:
==22824==     in use at exit: 72,704 bytes in 1 blocks
==22824==   total heap usage: 14 allocs, 13 frees, 122,404 bytes allocated
==22824==
==22824== LEAK SUMMARY:
==22824==    definitely lost: 0 bytes in 0 blocks
==22824==    indirectly lost: 0 bytes in 0 blocks
==22824==      possibly lost: 0 bytes in 0 blocks
==22824==    still reachable: 72,704 bytes in 1 blocks
==22824==         suppressed: 0 bytes in 0 blocks
==22824== Rerun with --leak-check=full to see details of leaked memory
==22824==
==22824== For counts of detected and suppressed errors, rerun with: -v
==22824== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) yuq8@andromeda-17 16:35:16 ~/253P/hw_lab/Makeup/quiz_4_prime
$
```

Little Notes:

1. In the original quiz, the main problem I'm hesitating on is how to test the primality and construct the set of prime, efficiently.
2. There is actually a problem of NP primality test. And I learned some method on this:
   a) sieve of Eratosthenes: delete out the multiply from the small known prime number, 2, 3, 5… Until the sqrt() of the wanted number. O(n log log n)
   But since our job here can reuse the former result of prime number, this method will have some replicated work.
   b) According to the rule: If is number is not in the form of 6x+1 of 6x+5, it must be not a prime number. So we can eliminate a series of number that are not prime.
   c) Here, I firstly decide to use the most intuitive way, simplify the range from the former prime dataset.
   d) But since the primeset should be maintained till the wanted number, which might be large, so I adopt the (b) method with the rule that composite number must be multiply of prime. Here's the optimized function of judging a prime number:

```
12  bool isprime(int value ){
13      if (value % 6 != 1 && value %6 != 5)
14          return false;
15      for (int i = 0; primeset[i] <= sqrt(value); i++){//composite number must be multiply of prime
16          if (value % primeset[i] == 0)
17              return false;
18      }
19      return true;
20  }
```

HW 2: Problem 1 c-String Functions

Dear TA,

You did not give me the scores for report since you said you cannot see the details. I have emailed you to supply for information, but you did not reply to me. So I re-emailed again.

If there are any problems, please contact me at yuq8@uci.edu

Thanks for you work~ Have a nice weekend!
Yu Qin