CSI 3334 Data Structures
Homework 2

Yufan Xu

January 30, 2018

Do your own work for this assignment; do not work with others. Consult the book and your professor for help if you need it. Do not use any other resources. Please write neatly, or preferably type your answers. Use good grammar, correct spelling, and complete sentences.

1. (5 points) A const method is one which does not change the data member(s) of an object when the method runs. It is signified by a "const" after the argument list of the method, e.g. void doSomething() const;. Which of the following methods on a Stack class should be marked as const methods based on what they apparently do?

   (a) void pop()

   (b) void clear()

   (c) void push(const Base & val)

   (d) const Base &top()

   (e) int getSize()

   Solution:
   (d) and (e) can be marked as const after the argument. Because these two functions does not change or manipulate anything within the class, they just store or return some copied information.

2. (10 points each, 30 points total) Analyze these functions. For each function, give the complete T(n) (or T(m,n), or whatever is appropriate), then find the tightest time complexity you can, in Theta notation. You need not show every step in getting from T(n) to Theta. You should assume all inputs are > 0.

   (a)
```
int f1(int n) {
        int i = 1;                      // c1
        for (int j = 1;                 // c2
                        j <= n;         // c3
                                j++) {  // c4
```

```
        i = i * j;                  // c5
    }
    return i;                       // c6
}
```

Solution to (a):
$c_1(1)$
$c_2(1)$
$c_3(n+1)$
$c_4(n)$
$c_5(n)$
$c_6(1)$
$T(n) = c_1(1) + c_2(1) + c_3(n+1) + c_4(n) + c_5(n) + c_6(1)$
$T(n) = c_1 + c_2 + c_3(n+1) + c_4(n) + c_5(n) + c_6$
$T(n) = c_1 + c_2 + c_3n + c_3 + c_4n + c_5n + c_6$
$T(n) = (c_3 + c_4 + c_5)n + c_1 + c_2 + c_3 + c_6$
$T(n) = c_7n + c_8$ where $c_7 = c_3 + c_4 + c_5$ and $c_8 = c_1 + c_2 + c_3 + c_6$
So $T(n) = \Theta(c_7n + c_8)$

(b)
```
int f2(int m, unsigned int n) {
    for (int i = 0;                     // c1
                    i < 2 * m;          // c2
                                i++) {  // c3
        for (int j = n;                 // c4
                         j > 0;         // c5
                                j--) {  // c6
            return j;                   // c7
        }
    }
    return 0;                           // c8
}
```

Solution to (b):
$c_1(1)$
$c_2(2m+1)$
$c_3(2m)$
$c_4(1)$
$c_5(n+1)2m$
$c_6(n)2m$
$c_7(n)2m$
$c_8(1)$
$T(m,n) = c_1(1) + c_2(2m+1) + c_3(2m) + c_4(1)2m + c_5(n+1)2m + c_6(n)2m +$
$c_7(n)2m + c_8(1)$
$T(m,n) = c_1 + c_2 2m + c_2 + c_3 2m + c_4 2m + c_5 2mn + c_5 2m + c_6 2mn + c_7 2mn + c_8$
$T(m,n) = (c_5 + c_6 + c_7)2mn + (c_2 + c_3 + c_4 + c_5)2m + c_1 + c_2 + c_8$

$T(m,n) = c_9mn + c_{10}m + c_{11}$ where $c_9 = 2(c_5 + c_6 + c_7)$, $c_{10} = 2(c_2 + c_3 + c_4 + c_5)$, $c_{11} = c_1 + c_2 + c_8$
So, $T(m,n) = \Theta(c_9mn + c_{10}m + c_{11})$

(c)
```
void f3(int n) {
    for (int i = 0;                          // c1
                    i < n;                   // c2
                           i++) {            // c3
        for (int j = 10;                     // c4
                         j >= 0;             // c5
                                j--) {       // c6
            cout << "i = " << i;             // c7
            cout << ", j = " << j << endl; // c8
        }
    }
}
```

Solution to (c):
$c_1(1)$
$c_2(n+1)$
$c_3(n)$
$c_4(n)$
$c_5(12n)$
$c_6(11n)$
$c_7(11n)$
$c_8(11n)$
$T(n) = c_1(1) + c_2(n+1) + c_3(n) + c_4(n) + c_5(12n) + c_6(11n) + c_7(11n) + c_8(11n)$
$T(n) = (c_2 + c_3 + c_4 + 12c_5 + 11c_6 + 11c_7 + 11c_8)n + c_1 + c_2$
$T(n) = c_9n + c_{10}$ where $c_9 = c_2 + c_3 + c_4 + 12c_5 + 11c_6 + 11c_7 + 11c_8$ and $c_{10} = c_1 + c_2$
So $T(n) = \Theta(c_9n + c_{10})$

3. (10 points) Will the following C++ function always terminate, for any input? Prove your answer. You should assume for this problem that the int type does NOT wrap around (it can count to any integer, positive or negative).

```
int g(int n) {
    int x = g(n - 1);
    if (x > 0) {
        return x + 1;
    } else {
        return 1;
    }
```

```
}
```

Solution:
This recursive function will never terminate no matter what the value of n is. Because the first line blocks the following base cases, the function can only execute the first line and deduce the value of x everytime, then it will start again and deduce the value of x again, this would eventually use up the memory on the stack and cause segmentation fault.

4. (10 points) Consider a standard binary tree with n nodes, where every node has a pointer to two children, either of which may be NULL. In this tree, are there more NULL child pointers, or non-NULL child pointers? Prove your answer. Remember that n could be any integer greater than zero, so we're not just talking about one particular tree for some fixed n, but ANY tree.

   Solution:
   The NULL child pointers should be more than non-NULL child pointers. Because a leaf node must have two children point to NULL. For example, if the root is the only node in the tree, then the tree would have two NULL children pointed by the root. If the tree has one root and two children, suppose the left child of the root is not NULL and the right child is NULL, then there is another child node on the left points to two NULL nodes, so in total the tree has two non-NULL child pointers and three NULL child pointers. So, as a result, a binary search tree will always have more NULL child pointers than non-NULL pointers if every nodes had either one child is NULL.

5. (10 points) Exercise 2.11 in your textbook. For this problem and the next one, consider setting up an equation $\frac{T(n1)}{t1} = \frac{T(n2)}{t2}$. Think about what this equation means.

   Solution to (a):
   Because the growth rate is linear,
   $t_0 = \frac{500}{100} * 0.5ms$
   $t_0 = 5 * 0.5ms = 2.5ms$

   Solution to (b):
   Because the growth rate is $O(NlogN)$,
   $t_1 = \frac{500log500}{100log100} * 0.5ms$
   $t_1 = \frac{1349.49}{200} * 0.5ms$
   $t_1 = 6.747 * 0.5ms$
   $t_1 = 3.37ms$

   Solution to (c):

Because the growth rate is N^2,
$t_2 = \frac{500^2}{100^2} * 0.5ms$
$t_2 = \frac{250000}{10000} * 0.5ms$
$t_2 = 25 * 0.5ms$
$t_2 = 12.5ms$

Solution to (d):
Because the growth rate is N^3,
$t_3 = \frac{500^3}{100^3} * 0.5ms$
$t_3 = \frac{125000000}{1000000} * 0.5ms$
$t_3 = \frac{1}{2}5 * 0.5ms$
$t_3 = 62.5ms$

6. (10 points) Exercise 2.12 in your textbook. For part (b), you will need to use some guess-and-check to find the answer.

Solution to (a):
$x = 100 * \frac{1min}{0.5ms}$
$x = 100 * \frac{60000ms}{0.5ms}$
$x = 100 * 120000$
$x = 12000000$
$x = 1.2 * 10^7$

Solution to (b):
$xlogx = 100log100 * \frac{60000ms}{0.5ms}$
$xlogx = 200 * 120000$
$xlogx = 24000000$
$logx^x = 24000000$
$x = 3656808$
$x = 3.656808 * 10^6$

Solution to (c):
$x^2 = 100^2 * \frac{60000ms}{0.5ms}$
$x^2 = 10000 * 120000$
$x^2 = 1200000000$
$x = \sqrt{1.2 * 10^9}$
$x = 34641$
$x = 3.4641 * 10^4$

Solution to (d):
$x^3 = 100^3 * \frac{60000ms}{0.5ms}$
$x^3 = 1000000 * 120000$
$x^3 = 120000000000$

$x = \sqrt[3]{1.2 * 10^{11}}$
$x = 4932$
$x = 4.932 * 10^3$

7. (10 points) Exercise 2.29 in your textbook (Hint: the answer has to do with how we perform algorithm analysis).

   Solution:
   Because the size of an integer that a computer can store is limited. For example, a 32-bit computer can store a integer up to 32 bits, any numbers above 32 bits will cause overflows and gives incorrect result. The largest number that a 32-bit machine can store is 4294967296, which is 2^{32}, and the binary representation of this number is 1111 1111 1111 1111 1111 1111 1111 1111, if we add anything to this, it would cause problem and errors for the computer because it exceeds the 32-bit limit. So, it is very important to assume all integers have fixed size for computers

8. (10 points) Exercise 3.9 in your textbook.

   Solution:
   Because these operations are changing the size of a vector object. For example, when we call pushback or insert, we are adding an element into the array, but this will actually create a new vector object in another memory location, then it will copy all elements in the previous object to the new object and delete the old one. So the iterators were pointing to the memory location that has been removed and they will need to point to the new location. As a result this will invalidates all iterators viewing the vector object.

9. (10 points) Exercise 3.27 in your textbook (here "stack space" means the "space on the runtime stack" the stack that stores function calls and local variables).

   Solution: It is very likely to run out of the stack space if N = 50. Because the main reason is that the amount of memory in the runtime stack runs out very quickly as the value of N gets larger, it causes the stack space grows exponentially whereas the runtime stack has only fixed amount of space. As I am testing the function on my machine, the computer prints incorrect result when N = 50, it is because the function ran out of stack space and caused segmentation faults.

10. (10 points) Exercise 3.36 in your textbook. Think outside the box on this one. The

answer is simple.

Solution:
Suppose ptr is a temporary pointer that points to the node,
The following function can remove a node in constant time:

```
ptr->data = ptr->next->data;
ptr->next = ptr->next->next;
```

The above algorithm does these steps, first it copies the data in the next node to the data in the current node, which is where ptr points to; then it will set the next node to the node after the next node, which is next-¿next. Then, the next node is deleted at O(1) time.