

OJ review 1

Huangjie Zheng

04/25/2018

Outline

- Assignment 1,4,5
- How to debug efficiently

Assignment review

Assignment1: Big Integer Class

- Description:
 - Implement a big integer class with overloading some operators (+, -, *, =).
- Remark:
 - Classic operators (in C++) **do not** serve for created class
 - Overloading operators are **redefined** according to our need. (creation, modification, etc. are not allowed)

Assignment1: Big Integer Class

```
class BigInteger
{
private:
    int data[500];
    int length;

public:
    BigInteger(){ length = 1; memset(data,0,sizeof(data)); }
    BigInteger(const char*); //First thing: char -> BigInteger
    BigInteger &operator=(const BigInteger &); //Second: define the result
    BigInteger operator+(const BigInteger &) const; // Third: define operators
    BigInteger operator-(const BigInteger &) const;
    BigInteger operator*(const BigInteger &) const;
    void show(); // Finally, present the result
};
```

Assignment1: Big Integer Class

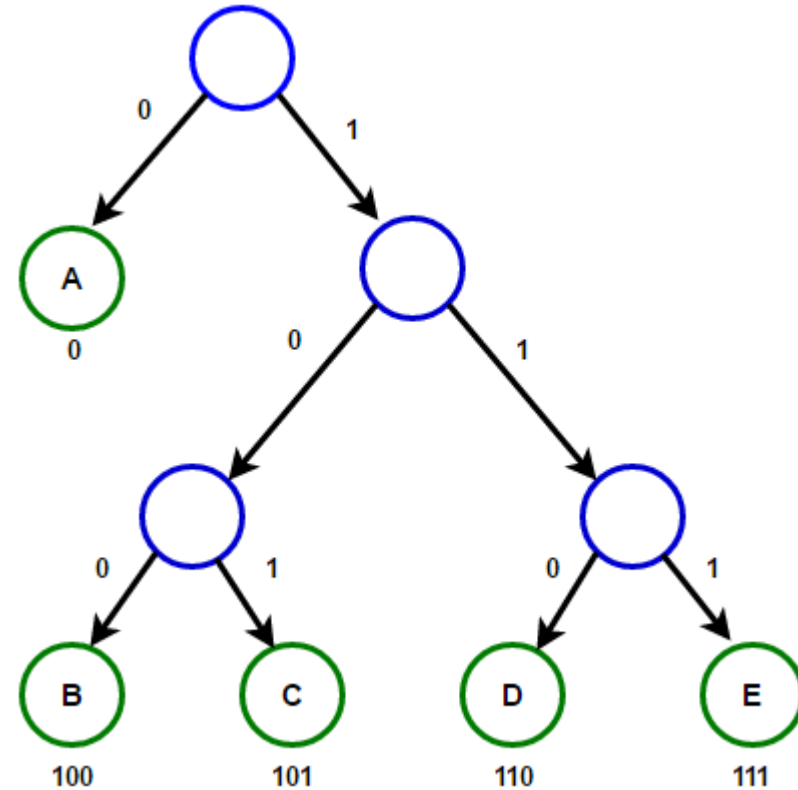
- BigInteger(const char*);
 - “char” to “BigInteger”
 - Length <- strlen
 - data[] <- char*
- Overloading “=”
 - this <- input (length, data)
- Overloading “+,-,*”
 - Past exercise

Some helpful links

- Why return a reference:
<https://www.cnblogs.com/codingmengmeng/p/5871254.html>
- How overloading works:
https://en.wikibooks.org/wiki/C%2B%2B_Programming/Operators/Operator_Overloading

Assignment4: Huffman encoding

- Background:
 - ASCII: 8bit/character
- Huffman:



Assignment4: Huffman encoding

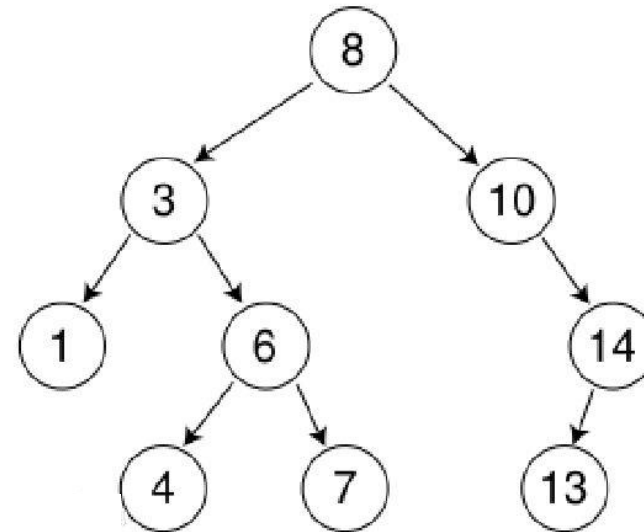
- Construct Huffman coding tree
- Traversal of the tree:
 - Depth of the leave node (code length)
 - Frequency
 - Final length = $\sum$ code length x frequency

Assignment4: Huffman encoding

- Trick: Huffman propriety
 - Optimized **Weighted Path Length of Tree**
- Alternative solution:
 - Compute the weighted path of coding tree
 - Recursively compute the weight of nodes
 - Ex: min-Heap, pop and add every two tops, push back.

Assignment5: Binary Search Tree

- Node:
 - Left < Right
- Inorder traversal:
 - Ordered sequence



Assignment5: Binary Search Tree

- Common ancestor of two given nodes:
 - Check given node “a”, “b”: $a > b$ or $a < b$

Suppose $a > b$:

- Recursively move downward
- Check if current node “c” satisfy : $a < c < b$
- If $c > b$: move to the left child of c
- If $c < a$: move to the right child of c

How to debug

Actually...

- Debug is a very personal thing...
- Everyone has his/her own way
- Experience is precious
- Good coding habits

Helpful habits

- Use “function”
- Use “class”
- Write down some comments (optional)

Some debug methods



Debug: Compile error

- Easiest to deal with
 - Read carefully the error information
- Ex:

```
sample.c:10:1: error: unknown type name 'iint'
  iint main() {
  ^
sample.c: In function 'main':
sample.c:11:13: error: 'bc' undeclared (first use in this function)
    int a = bc;
             ^
```

Debug: Runtime error

- Typically: misuse of memory:
 - Pointer
 - Subscribe of the array is out of range

```
int * p=(int *)malloc(5 * sizeof(int));  
*(p+10)=10;
```

Debug: locate the bug

- Breakpoint: check the running process
- If online judge:
 - add some output mechanism
 - delete(comment) the deployment of some function

Debug: empirical check

- Initialization of algorithm
- Logical order of:
 - Loops (e.g. break/continue)
 - Logic operation
- Is our algorithm strong enough to cover all cases?
- MOST IMPORTANTLY: PATIENCE

Thanks

<https://jegzheng.github.io/teaching/2018-spring-teaching>