Algorithms HW2

화학부 2016-14043 이준영

1. **Environment**
   a. Language : C++17
   b. Compiler : clang++
   c. IDE : Visual Studio Code
   d. OS : MacOS Monterey 12.3
   e. tasks.json (for build)

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "cppbuild",
      "label": "C/C++: clang++ build active file",
      "command": "/usr/bin/clang++",
      "args": [
        "-std=c++17",
        "-fdiagnostics-color=always",
        "-g",
        //"${workspaceFolder}/inputMaker/*.cpp",
        "${workspaceFolder}/*.cpp",
        "-o",
        //"${workspaceFolder}/inputMaker/inputMaker"
        "${workspaceFolder}/main",
      ],
      "options": {
        "cwd": "${fileDirname}"
      },
      "problemMatcher": [
        "$gcc"
      ],
      "group": {
        "kind" : "build",
        "isDefault": true
      },
      "detail": "compiler: /usr/bin/clang++"
    }
  ]
}
```

   f. execution code (in terminal)

```
./main
```

2. **Code**
   a. Used libraries

```
#include <iostream> //printing result on console
#include <fstream> //reading input and writing output
#include <string>
#include <vector>
#include <array> //containing input data
```

b. Overview

I made two classes for implementation, osTree and *Checker*. osTree reads input file with “readInput” function and builds tree with “buildTree” function.

```
int main(){
    osTree tree;
    Checker checker;
    tree.readInput("input.txt");

    tree.buildTree();

    tree.makeOutput("output.txt");

    checker.inCommand = tree.inCommand;
    checker.outputSequence = tree.outputSequence;
    checker.makeOutput("checker.txt");

    return 0;
}
```

c. osTree

```
class osTree{
public:
    Node* root = nullptr;
    vector<int> a;
    vector<array<int ,2>> inCommand;    // {command, key}   command : [I: 0 / D: 1 / S: 2 / R: 3]
    vector<int> outputSequence;
};
```

d. treeNode

```
class Node{
public:
    Node* parent = nullptr;
    Node* left = nullptr;
    Node* right = nullptr;
    int key = 0;
    int size = 1;
    bool RED = true;    // RED = false if black node
};
```

3. **Input**

a. inputMaker

I made *inputMaker* program to make input files with various numbers of elements. I made seven input files with different number of elements. All elements are in range [1 : 9999]. I used <random> library to choose element randomly. I made 10 input files simultaneously to check the correctness. I made several input files in “inputMaker” directory. The implementation follows as:

```
#include <iostream>
#include <fstream>
#include <string>
#include <random>

#define N 1000  //Number of elements
#define MAX 9999  //Range[1 : MAX]

using namespace std;

int main(){
    random_device rd;
    mt19937 mersenne(rd());
    uniform_int_distribution<> ran(1, MAX);
    uniform_int_distribution<> idx(1, N);
    uniform_int_distribution<> command(0, 3);

    for(int i = 0; i < 10; i++){
        ofstream Input;
        string fileName = "input" + to_string(i) + ".txt";
        Input.open(fileName);

        for(int i = 0; i < N; i++){
            int tempCom = command(mersenne);
            int tempKey = ran(mersenne);
            if(tempCom == 0) Input << "I " << tempKey << endl;
            else if(tempCom == 1) Input << "D " << tempKey << endl;
            else if(tempCom == 2) Input << "S " << tempKey << endl;
            else if(tempCom == 3) Input << "R " << tempKey << endl;
        }

        Input.close();
    }

    return 0;
}
```

b. Input Files Summary

	Number of Elements (N)
input1	same as the sample input
input2	1000
input3	1000
input4	1000
inpu5	1000

4. **Checker**

Checker takes boolean array to check correctness. The following codes are algorithms how it works.

```
class Checker{
public:
    array<bool, 10000> checkArray = {false};
    vector<array<int ,2>> inCommand;    // {command, key}   command : [I: 0 / D: 1 / S: 2 / R: 3]
    vector<int> outputSequence;
    vector<int> checkSequence;
};
```

```
void processCheckSeq(){
        for(auto itr = inCommand.begin(); itr != inCommand.end(); itr++){
            int tempCom = (*itr)[0];
            int tempKey = (*itr)[1];
            if(tempCom == 0) {
                if(!checkArray[tempKey]) {
                    checkArray[tempKey] = true;
                    checkSequence.push_back(tempKey);
                }
                else checkSequence.push_back(0);
            }
            else if(tempCom == 1) {
                if(checkArray[tempKey]) {
                    checkArray[tempKey] = false;
                    checkSequence.push_back(tempKey);
                }
                else checkSequence.push_back(0);
            }
            else if(tempCom == 2) {
                int count = 0;
                for(int i = 1; i < checkArray.size(); i++){
                    if(checkArray[i]){
                        count++;
                        if(count == tempKey){
                            checkSequence.push_back(i);
                            break;
                        }
                    }
                }
                if(count != tempKey) checkSequence.push_back(0);
            }
            else if(tempCom == 3) {
                int count = 0;
                if(!checkArray[tempKey]) checkSequence.push_back(0);
                else{
                    for(int i = 1; i <= tempKey; i++){
                        if(checkArray[i]) count++;
                    }
                    checkSequence.push_back(count);
                }
            }
        }
    }
```

5. **Example Running**
    a. input.txt
        I 17
        I 3
        I 22
        I 44

I 19
I 21
I 6
I 10
D 4
D 22
I 22
I 3
S 2
S 10
R 5
R 3

b. output.txt

I 17
I 3
I 22
I 44
I 19
I 21
I 6
I 10
D 4
D 22
I 22
I 3
S 2
S 10
R 5
R 3
17
3
22
44
19
21
6
10
0
22
22
0
6
0
0
1

c. checker.*txt*

Correct

(If result is wrong “Wrong” is printed)