Project 2 B10815057

1. PLA files

input1.pla : B'CD'+ABC+A'BC+ABC'D

```
.i 4
.o 1
.ilb a b c d
.ob F
.p 4
-010 1
111- 1
011- 1
1101 1
.e
```

input2.pla : AB'CD + B'D'E + A'CD + AD + BE' + CE

```
.i 5
.o 1
.ilb a b c d e
.ob F
.p 6
1011- 1
-0-01 1
0-11- 1
1--1- 1
-1--0 1
--1-1 1
.e
```

2. DOT files

output1.dot:

```
digraph ROBDD{
    {rank=same 1}
    {rank=same 2 3}
    {rank=same 4 5 7}
    {rank=same 9 14}
    0 [label=0, shape=box]
    1 [label="a"]
    2 [label="b"]
    3 [label="b"]
    4 [label="c"]
    5 [label="c"]
    7 [label="c"]
    9 [label="d"]
    14 [label="d"]
    16 [label=1, shape=box]
    1->2 [label="0", style=dotted]
    1->3 [label="1", style=solid]
    2->4 [label="0", style=dotted]
    2->5 [label="1", style=solid]
    3->4 [label="0", style=dotted]
    3->7 [label="1", style=solid]
    4->0 [label="0", style=dotted]
    4->9 [label="1", style=solid]
    5->0 [label="0", style=dotted]
    5->16 [label="1", style=solid]
    7->14 [label="0", style=dotted]
    7->16 [label="1", style=solid]
    9->16 [label="0", style=dotted]
    9->0 [label="1", style=solid]
    14->0 [label="0", style=dotted]
    14->16 [label="1", style=solid]
}
```

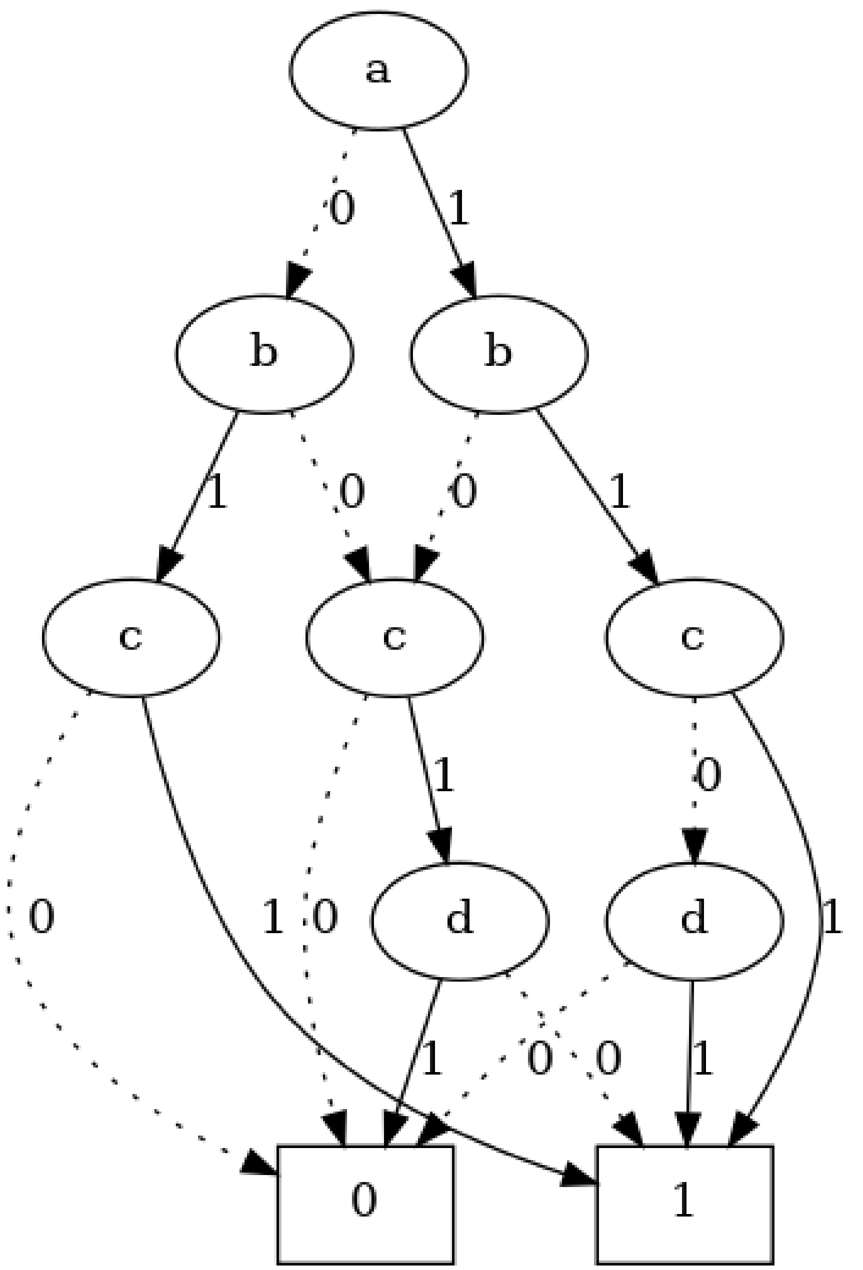
output2.dot:

```
digraph ROBDD{
    {rank=same 1}
    {rank=same 2 3}
    {rank=same 4 5 7}
    {rank=same 8 9 14}
    {rank=same 16 20}
    0 [label=0, shape=box]
    1 [label="a"]
    2 [label="b"]
    3 [label="b"]
    4 [label="c"]
    5 [label="c"]
    7 [label="c"]
    8 [label="d"]
    9 [label="d"]
    14 [label="d"]
    16 [label="e"]
    20 [label="e"]
    32 [label=1, shape=box]
    1->2 [label="0", style=dotted]
    1->3 [label="1", style=solid]
    2->4 [label="0", style=dotted]
    2->5 [label="1", style=solid]
    3->9 [label="0", style=dotted]
    3->7 [label="1", style=solid]
    4->8 [label="0", style=dotted]
    4->9 [label="1", style=solid]
    5->20 [label="0", style=dotted]
    5->32 [label="1", style=solid]
    7->14 [label="0", style=dotted]
    7->32 [label="1", style=solid]
    8->16 [label="0", style=dotted]
    8->0 [label="1", style=solid]
    9->16 [label="0", style=dotted]
    9->32 [label="1", style=solid]
    14->20 [label="0", style=dotted]
    14->32 [label="1", style=solid]
    16->0 [label="0", style=dotted]
    16->32 [label="1", style=solid]
```

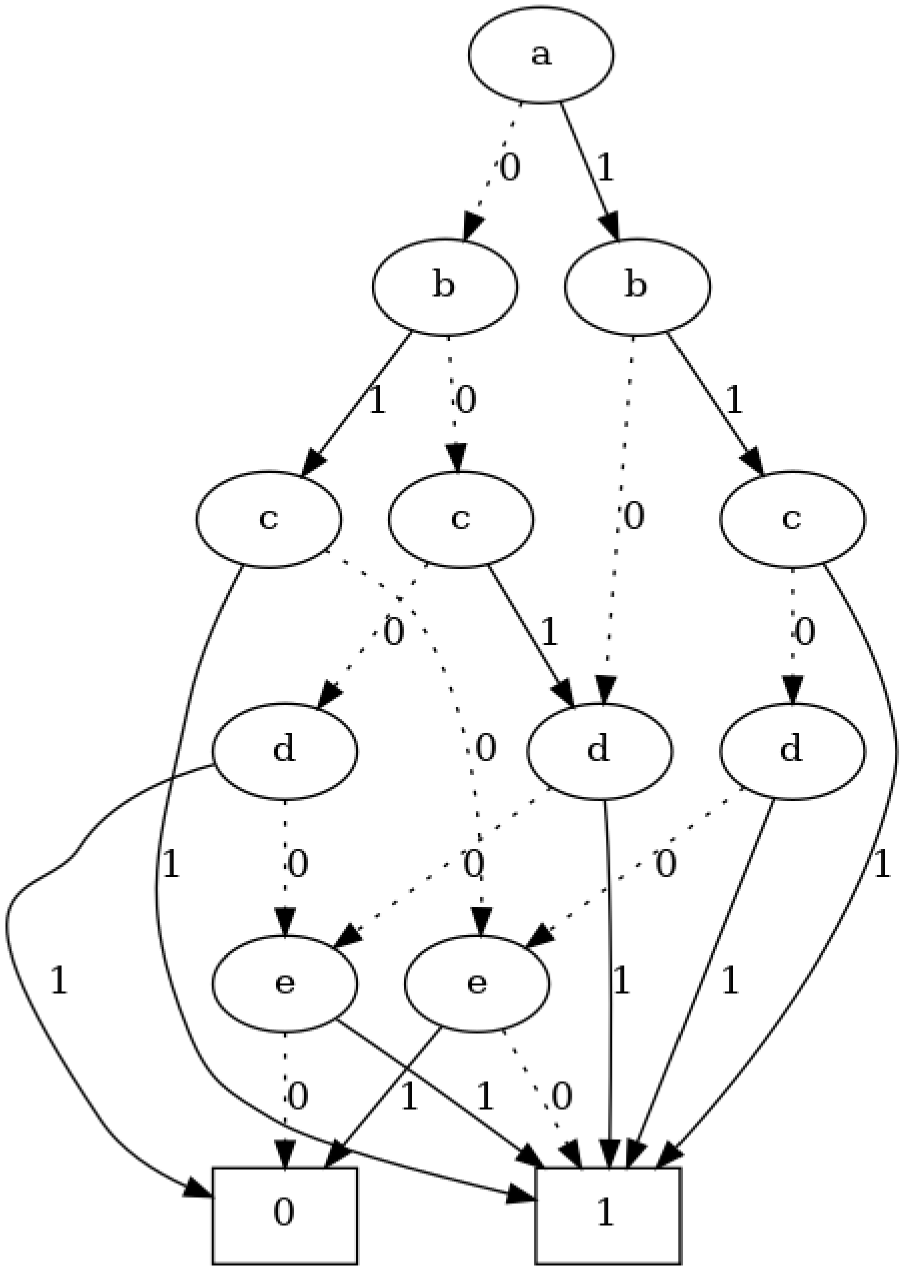
```
    20->32 [label="0", style=dotted]
    20->0 [label="1", style=solid]
}
```

3. Screenshots

output1.png



output2.png



4. Source code: [gist]

```cpp
#include<fstream>
#include<iostream>
#include<string>
#include<vector>
#include<sstream>
#include<bitset>
using namespace std;
class Node {
public:
    int else_edge,then_edge;//index of arrayreduced_order()
    string value="";
    bool redundent = false;
    bool operator==(Node& i) {
        return (this->value == i.value) && (this->else_edge == i.else_edge) && (this->then_edge ==
i.then_edge);
    }
    bool edge_same() {
        return else_edge == then_edge;
    }

};
class ROBDD {
public:
    ROBDD() {}
    ~ROBDD() {
        delete[] btree;
    }
    void create_tree() {
        size = (1 << i_count) + 1;
        btree = new Node[size];
        btree[0].value = "0";
        btree[size-1].value = "1";
    }
    int layer_start(int L) {
        return (1 << L);
    }
    int layer_end(int L) {
        return (1 << (L + 1));
    }
    void create_layer(string input,int layer_count) {
        for (int i = layer_start(layer_count);i < layer_end(layer_count);i++) {
            btree[i].value = input;
            if (layer_count == i_count - 1) continue;
            btree[i].else_edge = i * 2;
            btree[i].then_edge = i * 2 + 1;
        }
    }
    void output(ofstream& dot) {
        for (int i = layer_start(i_count - 1),j = 0;i < layer_end(i_count - 1);i++,j+=2) {
            btree[i].else_edge = get_result(j) ? size - 1 : 0;
            btree[i].then_edge = get_result(j+1) ? size - 1 : 0;
        }
        reduced_order();
        //output to file
        dot << "digraph ROBDD{\n";
        for (int i = 0;i < i_count;i++) {
            dot << "\t{rank=same";
            for (int j = layer_start(i);j < layer_end(i);j++) {
                if (btree[j].redundent) continue;
                dot << ' ' << j;
            }
            dot << "}\n";
        }
        dot << "\t0 [label=0, shape=box]\n";
        for (int i = 1;i < size - 1;i++) {
            if (btree[i].redundent) continue;
            dot << '\t' << i << R"( [label=")" << btree[i].value << "\"]\n";
        }
        dot << '\t' << size - 1 << " [label=1, shape=box]\n";

        for (int i = 1;i < size - 1;i++) {
            if (btree[i].redundent) continue;
            dot << '\t' << i << "->" << btree[i].else_edge << R"( [label="0", style=dotted])" << '\n';
            dot << '\t' << i << "->" << btree[i].then_edge << R"( [label="1", style=solid])" << '\n';
        }
        dot << "}\n";
    }
```

```
    bool get_result(int num) {//get boolean function result
        bitset<8> bin(num);
        bool result = false;
        for (int i = 0;i < product_terms.size();i++) {
            bool line_result = true;
            for (int j = 0;j < i_count;j++) {
                if (product_terms[i][j] == '-') continue;
                if ((!bin[i_count - j - 1] && product_terms[i][j] == '1') || (bin[i_count - j - 1] &&
product_terms[i][j] == '0')) {// found 0
                    line_result = false;
                    break;
                }
            }
            if (product_terms[i].back() == '0') line_result = !line_result;

            if (line_result) {
                result = true;
                break;
            }
        }
        return result;
    }
    void reduced_order() {//main algorithm here
        print_tree();
        for (int i = i_count - 1;i >= 0;i--) {//layer from down to up
            int start = layer_start(i), end = layer_end(i);
            bool change = false;
            for (int j = start;j < end;j++) {
                if (btree[j].redundent) continue;
                if (btree[j].edge_same()) {
                    (j % 2 ? btree[j / 2].then_edge : btree[j / 2].else_edge) = btree[j].else_edge;
                    btree[j].redundent = true;
                    change = true;
                    continue;
                }
                for (int k = j + 1;k < end;k++) {
                    if (btree[k].redundent) continue;
                    if (btree[j] == btree[k]) {
                        (k % 2 ? btree[k / 2].then_edge : btree[k / 2].else_edge) = j;
                        btree[k].redundent = true;
                        change = true;
                    }
                }
            }
            if(change) print_tree();
        }

    }
    void print_tree() {
        cout << "index" << "\t\t" << "Variable" << "\t" << "Else-edge" << "\t" << "Then-edge" << "\t"
<< "Comment" << endl;
        for (int i = 1;i < size-1;i++) {
            cout << i << "\t\t" << btree[i].value << "\t\t" << btree[i].else_edge << "\t\t" <<
btree[i].then_edge << "\t\t" << (btree[i].redundent ? "redundent" : "") << endl;
        }
        cout << "-----------------------------------------------------------------------------" << endl;
    }
    int i_count = 0, o_count = 0 , size = 0;
    Node* btree = nullptr;
    vector<string> product_terms;
};
```

```cpp
int main(int argc, char* argv[]) {
    if (argc != 3) {
        cout << "command error!" << endl;
        return 0;
    }
    ifstream input(argv[1]/*"input.pla"*/);//argv[0] 是本程式名稱
    ofstream dotfile(argv[2]/*"output.pla"*/);
    string line,word;
    ROBDD robdd;
    while (1) {
        //sline.str(line);
        input >> word;
        if (word == ".i") {
            input >> robdd.i_count;
            robdd.create_tree();
        }
        else if(word == ".o"){
            input >> robdd.o_count;
        }
        else if (word == ".ilb") {
            string tmp;
            for (int i = 0;i < robdd.i_count;i++) {
                input >> tmp;
                robdd.create_layer(tmp,i);
            }
        }
        else if (word == ".ob") {
            string tmp;
            input >> tmp;
        }
        else if (word == ".p") {
            int line_count;
            char nl;
            input >> line_count;//讀掉換行
            input.ignore();
            for (int i = 0;i < line_count;i++) {
                getline(input, line);
                robdd.product_terms.push_back(line);
            }
        }
        else if (word == ".e") {
            break;
        }
        else {
            cout << "error reading pla file!" << endl;
            return 0;
        }
    }
    robdd.output(dotfile);
    return 0;
}
```