实验环境:

```
OS:ubuntu 18.1.04 LTS
compiler:c++11,g++
MakeTool:CMake
```

实验原理

贪心.jpg

实验过程

```
#include <map>
#include <queue>
#include <arpa/nameser.h>
#include <ostream>

struct Node {
    double value = 0;
    Node *left = nullptr, *right = nullptr;
    char c = 0;

    Node() = default;

    Node(double v, char c1) : value(v), c(c1) {}

    bool isChar() { return !left; }

    Node(const Node &node) : value(node.value), left(node.left), right(node.right),
c(node.c) {}

    bool operator<(const Node &node) const { return !(value < node.value); }
};

struct cmp {
    bool operator()(const Node *node1, const Node *node2) { return *node1 < *node2; }
};

Node *build_huffman(const std::map<char, double> &weight) {
    Node *leaves;
    if (weight.size() == 1)
        return new Node(weight.begin()->second, weight.begin()->first);
    std::priority_queue<Node *, std::vector<Node *>, cmp> queue; // todo
    for (auto &item:weight)
        queue.push(new Node(item.second, item.first));
    while (true) {
        if (queue.size() == 2) {
            leaves = new Node();
            leaves->right = queue.top();
            leaves->value += queue.top()->value;
            queue.pop();
```

```
            leaves->left = queue.top();
            leaves->value += queue.top()->value;
            queue.pop();
            return leaves;
        } else {
            leaves = new Node();
            leaves->right = queue.top();
            leaves->value += queue.top()->value;
            queue.pop();
            leaves->left = queue.top();
            leaves->value += queue.top()->value;
            queue.pop();
            queue.push(leaves);
        }
    }
}

#include <stack>
#include <iostream>

std::string binary_print(int temp) {
    std::string string;
    if (!temp)return string = std::string("0");
    while (temp) {
        string += temp & 1 ? "1" : "0";
        temp >>= 1;
    }
    std::reverse(string.begin(), string.end());
    return string;
}

void huffman_print(const Node *root, const int temp = 0) {
    if (root->left->isChar())  // node of left tree is char
        std::cout << root->left->c << " : " << binary_print(temp) << std::endl;
    else huffman_print(root->left, temp << 1);

    if (root->right->isChar())  // node of right tree is char
        std::cout << root->right->c << " : " << binary_print(temp + 1) << std::endl;
    else huffman_print(root->right, (temp + 1) << 1);
}


int main(int argc, char **argv) {

    double value;
    char ch;
    int N;
    std::cin >> N;
    std::map<char, double> map;
    while (N--) {
        std::cin >> ch >> value;
        map[ch] = value;
    }
    huffman_print(build_huffman(map));
    return 0;
}
```

运行结果

input:

```
5
a 10.5
b 12.2
c 30.0
d 11
e 33
```

output:

```
e : 0
c : 1
d : 100
a : 101
b : 11
```