Олимпиада «Ломоносов» по информатике. 2022-2023 учебный год
Работа участника с логином inf23f_102

Прайморадичная система счисления

Посылка по задаче 1 «Прайморадичная система счисления»

```
easy_nums = [1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89]
p_nums = [1, 2, 6, 30, 210, 2310, 30030, 510510, 9699690, 223092870, 6469693230, 200560490130, 7420738134810, 304250263527210, 13082761331670030, 614889782588491410,
 32589158477190044730, 1922760350154212639070, 117288381359406970983270, 7858321551080267055879090, 557940830126698960967415390, 40729680599249024150621323470,
 3217644767340672907899084554130, 267064515689275851355624017992790, 23768741896345550770650537601358310]
N = int(input())
f_nums = []
for i in range(N):
    f_num = [int(x) for x in input().split(':')]
    num = 0
    for i in range(len(f_num)):
        num += f_num[i] * p_nums[len(f_num) - i-1]
    f_nums.append(num)
max_pr = 1
for f_num in f_nums:
    for pr in p_nums:
        if f_num%pr==0:
            if pr>max_pr:
                max_pr = pr
new_prs = []
for f_num in f_nums:
    if f_num%max_pr==0:
        new_prs.append(f_num)
a = max(new_prs)
b = 0
for f_num in new_prs:
    if f_num==a:
        b+=1
print(b)
```

Протокол проверяющей системы по задаче 1 «Прайморадичная система счисления»

см. файл report1.txt

Посылка по задаче 2 «Сундуки»

```
#include <iostream>
#include <algorithm>
#include <vector>

struct Case {
        std::uint32_t id;
```

```
        std::uint32_t money_in_monet;
};

std::uint32_t StrToMoney(const std::string& str) {
        std::uint32_t result = 0;
        for (const auto& ch : str) {
                if (ch == 'a') result += 1;
                else if (ch == 'b') result += 5;
                else if (ch == 'c') result += 10;
                else if (ch == 'd') result += 50;
                else if (ch == 'e') result += 100;
                else if (ch == 'f') result += 200;
                else if (ch == 'g') result += 500;
                else if (ch == 'h') result += 1000;
                else if (ch == 'i') result += 2500;
        }
        return result;
}

int main()
{
        std::uint32_t N;
        std::cin >> N;
        std::vector<Case> cases(N, { 0,0 });
        std::string temp_str;
        for (std::uint32_t i = 0; i < N; ++i) {
                cases[i].id = i;
                std::cin >> temp_str;
                cases[i].money_in_monet = StrToMoney(temp_str);
        }
        std::sort(cases.begin(), cases.end(), [](const Case& a, const Case& b) {return a.money_in_monet < b.money_in_monet; });
        std::vector<std::uint32_t> min_ids, max_ids;
        min_ids.reserve(N);
        max_ids.reserve(N);
        for (auto it = cases.begin(); it != cases.end(); ++it) {
                if (it->money_in_monet != cases.front().money_in_monet) break;
                min_ids.push_back(it->id);
        }
        for (auto it = cases.rbegin(); it != cases.rend(); ++it) {
                if (it->money_in_monet != cases.back().money_in_monet) break;
                max_ids.push_back(it->id);
        }
        std::sort(min_ids.begin(), min_ids.end());
        std::sort(max_ids.begin(), max_ids.end());
```

```
        std::uint32_t  max_id_of_mins = min_ids.back();
        std::uint32_t max_id_of_maxs = max_ids.back();
        if (max_id_of_mins == max_id_of_maxs) {
                std::uint32_t prev_a = 0, prev_b = 0;
                if (min_ids.size() > 1) prev_a = *std::next(min_ids.rbegin());
                if (max_ids.size() > 1) prev_b = *std::next(max_ids.rbegin());
                if (max_id_of_mins - prev_a < max_id_of_maxs - prev_b) max_id_of_mins = prev_a;
                else max_id_of_maxs = prev_b;
        }
        std::cout << std::min(max_id_of_mins, max_id_of_maxs) + 1 << "\n" << std::max(max_id_of_mins, max_id_of_maxs) + 1;

}
```

Протокол проверяющей системы по задаче 2 «Сундуки»

```
OK
28 total tests runs, 28 passed, 0 failed.
Score gained: 100 (out of 100).
```

Посылка по задаче 3 «Кубик»

```
#include <iostream>
#include <algorithm>
#include <vector>

void L(std::string& sost) {
        std::swap(sost[0], sost[12]);
        std::swap(sost[2], sost[14]);
        std::swap(sost[4], sost[6]);
        std::swap(sost[8], sost[9]);
}

void R(std::string& sost) {
        std::swap(sost[1], sost[13]);
        std::swap(sost[3], sost[15]);
        std::swap(sost[5], sost[7]);
        std::swap(sost[10], sost[11]);
}

void U(std::string& sost) {
        std::swap(sost[0], sost[15]);
        std::swap(sost[1], sost[14]);
        std::swap(sost[8], sost[10]);
        std::swap(sost[6], sost[7]);
}
```

```
void D(std::string& sost) {
        std::swap(sost[2], sost[13]);
        std::swap(sost[3], sost[12]);
        std::swap(sost[9], sost[11]);
        std::swap(sost[4], sost[5]);
}

std::string temp_sost;
void F(std::string& sost) {
        temp_sost.resize(sost.size(), '0');
        std::copy(sost.begin(), sost.end(), temp_sost.begin());
        sost[0] = temp_sost[2];
        sost[1] = temp_sost[0];
        sost[3] = temp_sost[1];
        sost[2] = temp_sost[3];
        sost[10] = temp_sost[6];
        sost[11] = temp_sost[7];
        sost[5] = temp_sost[10];
        sost[4] = temp_sost[11];
        sost[8] = temp_sost[4];
        sost[9] = temp_sost[5];
        sost[6] = temp_sost[9];
        sost[7] = temp_sost[8];
        sost[14] = temp_sost[12];
        sost[15] = temp_sost[14];
        sost[13] = temp_sost[15];
        sost[12] = temp_sost[13];
}

void MakeCommands(std::string& sost, const std::string& cmd) {
        for (const auto& cm : cmd) {
                if (cm == 'L') L(sost);
                else if (cm == 'R') R(sost);
                else if (cm == 'U') U(sost);
                else if (cm == 'D') D(sost);
                else if (cm == 'F') F(sost);
        }
}

int main()
{
        std::string sost, cmd;
        std::cin >> sost >> cmd;
```

```
        MakeCommands(sost, cmd);
        std::cout << sost;
}
```

Протокол проверяющей системы по задаче 3 «Кубик»

```
OK
51 total tests runs, 51 passed, 0 failed.
Score gained: 100 (out of 100).
```

Посылка по задаче 4 «Codemirror»

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <list>

//std::uint32_t MyStoi(const std::string& str, std::uint32_t start, std::uint32_t end) {
//      std::uint32_t result = 0;
//      for (std::uint32_t i = start; i < end; ++i) {
//              result *= 10;
//              result += str[i] - '0';
//      }
//      return result;
//}
//
//void StringToFNum(const std::string& str, std::pair<std::vector<std::uint32_t>, std::uint32_t>& full_f_num, std::uint32_t& max_last_zero_cnt, bool& is_zero_exists) {
//      std::uint32_t start = 0;
//      auto& f_num = full_f_num.first;
//      f_num.clear();
//      f_num.reserve(25);
//      auto& now_zero_cnt = full_f_num.second;
//      now_zero_cnt = 0;
//      bool is_start = true;
//      for (std::uint32_t end = str.find(':');; start = end + 1, end = str.find(':', start)) {
//              if (end >= str.size()) end = str.size();
//              std::uint32_t now_num = MyStoi(str, start, end);
//              if ((!is_start) || (now_num != 0)) {
//                      is_start = false;
//                      f_num.push_back(now_num);
//                      if (f_num.back() == 0) ++now_zero_cnt;
//                      else now_zero_cnt = 0;
//              }
//              if (end == str.size()) break;
//      }
```

```
//      if (f_num.empty()) is_zero_exists = true;
//      if (now_zero_cnt > max_last_zero_cnt) max_last_zero_cnt = now_zero_cnt;
//}

//struct Case {
//      std::uint32_t id;
//      std::uint32_t money_in_monet;
//};
//
//std::uint32_t StrToMoney(const std::string& str) {
//      std::uint32_t result = 0;
//      for (const auto& ch : str) {
//              if (ch == 'a') result += 1;
//              else if (ch == 'b') result += 5;
//              else if (ch == 'c') result += 10;
//              else if (ch == 'd') result += 50;
//              else if (ch == 'e') result += 100;
//              else if (ch == 'f') result += 200;
//              else if (ch == 'g') result += 500;
//              else if (ch == 'h') result += 1000;
//              else if (ch == 'i') result += 2500;
//      }
//      return result;
//}

//void L(std::string& sost) {
//      std::swap(sost[0], sost[12]);
//      std::swap(sost[2], sost[14]);
//      std::swap(sost[4], sost[6]);
//      std::swap(sost[8], sost[9]);
//}
//
//void R(std::string& sost) {
//      std::swap(sost[1], sost[13]);
//      std::swap(sost[3], sost[15]);
//      std::swap(sost[5], sost[7]);
//      std::swap(sost[10], sost[11]);
//}
//
//void U(std::string& sost) {
//      std::swap(sost[0], sost[15]);
//      std::swap(sost[1], sost[14]);
//      std::swap(sost[8], sost[10]);
//      std::swap(sost[6], sost[7]);
```

```
//}
//
//void D(std::string& sost) {
//      std::swap(sost[2], sost[13]);
//      std::swap(sost[3], sost[12]);
//      std::swap(sost[9], sost[11]);
//      std::swap(sost[4], sost[5]);
//}
//
//std::string temp_sost;
//void F(std::string& sost) {
//      temp_sost.resize(sost.size(), '0');
//      std::copy(sost.begin(), sost.end(), temp_sost.begin());
//      sost[0] = temp_sost[2];
//      sost[1] = temp_sost[0];
//      sost[3] = temp_sost[1];
//      sost[2] = temp_sost[3];
//      sost[10] = temp_sost[6];
//      sost[11] = temp_sost[7];
//      sost[5] = temp_sost[10];
//      sost[4] = temp_sost[11];
//      sost[8] = temp_sost[4];
//      sost[9] = temp_sost[5];
//      sost[6] = temp_sost[9];
//      sost[7] = temp_sost[8];
//      sost[14] = temp_sost[12];
//      sost[15] = temp_sost[14];
//      sost[13] = temp_sost[15];
//      sost[12] = temp_sost[13];
//}
//
//void MakeCommands(std::string& sost, const std::string& cmd) {
//      for (const auto& cm : cmd) {
//              if (cm == 'L') L(sost);
//              else if (cm == 'R') R(sost);
//              else if (cm == 'U') U(sost);
//              else if (cm == 'D') D(sost);
//              else if (cm == 'F') F(sost);
//      }
//}

struct TimePoint {
        std::uint32_t time;
        std::uint32_t request_id;
```

```
        std::uint32_t result_id;
};

struct Sost {
        std::list<char> text;
        std::list<char> buffer;
        std::list<char>::iterator cursor; // pos, before that cursor
        std::list<char>::iterator shift_start; // pos, before that is shift_start
        bool is_shift_press = false;
        Sost() {
                cursor = shift_start = text.end();
        }
};

void MakeCommand(Sost& sost, char cmd) {
        if (cmd == '{' || cmd == '}') {
                if (!sost.is_shift_press) {
                        sost.shift_start = sost.cursor;
                        sost.is_shift_press = true;
                }
                if (cmd == '{') { if (sost.cursor != sost.text.begin()) --sost.cursor; }
                else if (cmd == '}') { if (sost.cursor != sost.text.end()) ++sost.cursor; }
        }
        else {
                if (cmd >= 'a' && cmd <= 'z') {
                        if (sost.cursor == sost.text.begin()) {
                                sost.text.emplace_front(cmd);
                                sost.cursor = std::next(sost.text.begin());
                        }
                        else {
                                sost.text.emplace(sost.cursor, cmd);
                        }
                }
                else if (cmd == '<') {
                        if (sost.cursor != sost.text.begin()) --sost.cursor;
                }
                else if (cmd == '>') {
                        if (sost.cursor != sost.text.end()) ++sost.cursor;
                }
                else if (cmd == 'C') {
                        if (sost.is_shift_press) {
                                sost.buffer.resize(std::distance(sost.shift_start, sost.cursor));
                                std::copy(sost.shift_start, sost.cursor, sost.buffer.begin());
                        }
```

```
                        else sost.buffer.clear();
                }
                else if (cmd == 'V') {
                        if (sost.is_shift_press) sost.text.erase(sost.shift_start, sost.cursor);
                        for (const auto& ch : sost.buffer) sost.text.emplace(sost.cursor, ch);
                }
                else if (cmd == 'X') {
                        sost.buffer.clear();
                        if (sost.shift_start != sost.cursor) for (auto it = sost.shift_start; it != sost.cursor; ++it) sost.buffer.emplace_back(*it);
                        sost.text.erase(sost.shift_start, sost.cursor);
                }
                else if (cmd == 'D') {
                        if (sost.is_shift_press) sost.text.erase(sost.shift_start, sost.cursor);
                        else if (sost.cursor != sost.text.begin()) sost.text.erase(std::prev(sost.cursor));
                }
                sost.is_shift_press = false;
        }
}

int main()
{
        std::uint32_t N, K;
        std::string commands;
        std::cin >> N >> commands >> K;
        std::vector<TimePoint> requests(K, { 0,0,0 });
        std::vector<std::string> results(K);
        for (std::uint32_t i = 0; i < K; ++i) {
                requests[i].request_id = i;
                std::cin >> requests[i].time;
        }
        std::sort(requests.begin(), requests.end(), [](const TimePoint& a, const TimePoint& b) { return a.time < b.time; });
        Sost sost;
        std::uint32_t time_pos = 0;
        for (std::uint32_t i = 0; i < K; ++i) {
                if (time_pos < requests[i].time) for (; time_pos < requests[i].time; ++time_pos) MakeCommand(sost, commands[time_pos]);
                results[i].resize(sost.text.size(), '0');
                std::copy(sost.text.begin(), sost.text.end(), results[i].begin());
                requests[i].result_id = i;
        }
        std::sort(requests.begin(), requests.end(), [](const TimePoint& a, const TimePoint& b) {return a.request_id < b.request_id; });
        for (const auto& req : requests) std::cout << results[req.result_id] << "\n";
}
```

Протокол проверяющей системы по задаче 4 «Codemirror»

см. файл report4.txt