Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3232

Копалина М.А.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2024

Задача №A «Агроном-любитель»

```cpp
#include <iostream>

int main() {
    int n;
    std::cin >> n;
    int start_idx = 0, end_idx = 0, begin_idx = 1;
    int curr_val;
    int prev_prev_val = 0, prev_val = 0;
    int max_length = 0;
    for (int i = 1; i <= n; i++) {
        std::cin >> curr_val;
        if (n == 1) {
            std::cout << 1 << " " << 1 << std::endl;
            return 0;
        }
        else {
            if (curr_val == prev_prev_val && curr_val == prev_val) {
                begin_idx = i - 1;
            }
            int new_length = i - begin_idx;
            if (new_length > max_length) {
                max_length = new_length;
                start_idx = begin_idx;
                end_idx = i;
            }
            prev_prev_val = prev_val;
            prev_val = curr_val;
        }
    }
    printf("%d %d", start_idx, end_idx);
    return 0;
}
```

Пояснение к примененному алгоритму:

1) Переменные:

- n: количество цветков на грядке.

- start_idx: индекс начала участка без трех одинаковых цветков подряд.

- end_idx: индекс конца участка без трех одинаковых цветков подряд.

- begin_idx: индекс начала текущего потенциального участка без трех одинаковых цветков подряд.

- curr_val: значение текущего цветка.

- prev_prev_val: значение предпредыдущего цветка.

- prev_val: значение предыдущего цветка.

- max_length: максимальная длина участка без трех одинаковых цветков подряд.

2) Программа проходит по всем цветкам и при обнаружении трех одинаковых цветков подряд обновляет начало участка без трех цветков. После завершения проверки всех цветков, программа выводит начало и конец самого длинного участка без трех цветков одного вида подряд.

3) Важно проверить случай, когда n = 1. Тогда выводится максимальная длина размером 1.

4) Необходимо вставить цикл в программу для обновления переменных.

- Если текущее значение равно предыдущим двум значениям, обновляем begin_idx.

- Вычисляем новую длину участка без трех одинаковых цветков подряд.

- Если новая длина больше максимальной длины, обновляем max_length, start_idx и end_idx.

- Обновляем значения prev_prev_val и prev_val.

4) Выводим найденный участок без трех одинаковых цветков подряд.

Сложность алгоритма:

- Временная сложность: O(n), где n - количество цветков на грядке.

- Пространственная сложность: O(1), так как используется постоянное количество дополнительной памяти.

Задача №B «Зоопарк Глеба»

```cpp
#include <iostream>
#include <string>
#include <vector>

bool check(char l, std::vector<char> &letters) {
    if (letters.empty()) {
        return false;
    }
    return (l != letters.back()) &&
           letters.back() == l - 32 || letters.back() - 32 == l;
}
// isupper(l); l >= 'A' && l <= 'Z'
// islower(l); //l >= 'a' && l <= 'z'

int main() {
    std::string str;
    std::cin >> str;
    int n = str.size();

    std::string no = "Impossible";
    std::string yes = "Possible";

    std::vector<char> letters;
    std::vector<int> indexes;
    std::vector<int> animals, boxes;

    int animal_counter = 0;
    int box_counter = 0;
    std::vector<int> result(n);
    for (int i = 0; i < n; i++) {
        char l = str[i];
        if (islower(l)) {
            animals.push_back(++animal_counter);
            boxes.push_back(box_counter);
        }
        if (isupper(l)) {
            boxes.push_back(++box_counter);
            animals.push_back(animal_counter);
        }
    }
    for (int i = 0; i < n; i++) {
        char l = str[i];
        if (check(l, letters)) {
            if (islower(l)) {
                result[boxes[indexes.back()]] = animals[i];
            } else {
                result[boxes[i]] = animals[indexes.back()];
            }
            letters.pop_back();
            indexes.pop_back();
        } else {
            letters.push_back(l);
            indexes.push_back(i);
        }
    }
    if (letters.empty()) {
        std::cout << yes << std::endl;
        for (int i = 0; i < n/2; i++) {
            std::cout << result[i+1];
            if (i != n/2 - 1) {
```

```
                std::cout << " ";
            }
        }
    } else {
        std::cout << no << std::endl;
    }
    return 0;
}
```

Пояснение к примененному алгоритму:

Каждое животное имеет свою ловушку, и программа будет определять возможность пути животного к его ловушке без пересечения с другими животными.

1) На вход подается строка символов, представляющая собой совокупность животных и ловушек.

2) В массиве animals будут храниться индексы животных, а в массиве boxes - индексы ловушек, а также счетчики для животных и ловушек. В процессе выполнения циклов хранилища заполняются.

Например, если в массиве animals в позиции i будет число 3, это означает, что животное с индексом i должно попасть в ловушку с номером 3 (которая будет представлена в массиве boxes).

3) Затем программа проверяет последовательность символов на условие, что два символа не могут быть одновременно верхним и нижним регистром и не могут быть одновременно соседними по алфавиту (например, 'a' и 'A').

4) Если условие нарушено, программа выводит "Impossible". В противном случае она выводит "Possible" и показывает схему поимки животных.

Сложность алгоритма: O(n), где n - количество символов во входной строке с символами животных и ловушек. Это связано с тем, что программа проходит по строке только дважды, выполняя простые операции на каждом шаге.

Задача №C «Конфигурационный файл»

```cpp
#include <vector>
#include <map>
#include <iostream>

std::map<std::string, int> m; // Глобальное хранилище переменных и их значений
std::vector<std::map<std::string, int>> y; // Стек для хранения локальных переменных

void getChars(const std::string &ch, std::string &s1, std::string &s2) {
    size_t p = ch.find('=');
    if (p != std::string::npos) {
        s1 = ch.substr(0, p);
        s2 = ch.substr(p + 1);
    }
}

int main() {
    std::string ch;
    y.push_back(std::map<std::string, int>());

    while (getline(std::cin, ch)) {
        switch(ch[0]) {
            case '}':
                for (const auto& it : y.back()) {
                    m[it.first] = it.second;
                }
                y.pop_back();
                break;
            case '{':
                y.emplace_back();
                break;
            default:
                std::string s1, s2;
                getChars(ch, s1, s2);

                if (atoi(s2.c_str())) {
                    if (y.back().find(s1) == y.back().end()) {
                        y.back().insert_or_assign(s1, m[s1]);
                    }
                    m.insert_or_assign(s1, atoi(s2.c_str()));
                } else {
                    int res;
                    if (m.find(s2) == m.end()) {
                        res = 0;
                    } else {
                        res = m[s2];
                    }
                    if (y.back().find(s1) == y.back().end()) {
                        y.back().insert_or_assign(s1, m[s1]);
                    }
                    m.insert_or_assign(s1, res);
                    std::cout << std::to_string(res) << std::endl;
                }

                break;
        }
    }
    return 0;
}
```

Пояснение к примененному алгоритму:

Этот код представляет собой простой интерпретатор для обработки конфигурационных файлов. Он позволяет парсить строки из файла, выполнять присваивание переменных и выводить значения переменных при необходимости.

1) В функции getChars происходит разделение строки на две части: имя переменной и значение (константа или другая переменная).
2) В основной функции main читается строка из ввода. В зависимости от первого символа строки («{», «}» или любой другой символ) происходит определенное действие:

- Если встречается «}», то значения локальных переменных сохраняются в глобальном хранилище, и удаляется последний блок из стека.

- Если встречается «{», то добавляется новый пустой блок в стек.

- В остальных случаях происходит присваивание значения переменной: если значение константное, оно сохраняется в глобальном хранилище; если значение другой переменной, то текущее значение этой переменной копируется в текущий блок.

3) После каждого присваивания переменной значения происходит вывод этого значения.

Сложность алгоритма:

- Чтение строки из ввода и разбор каждой строки происходит за константное время O(1).

- Вставка/поиск элемента в std::map занимает O(log n) времени, где n - количество элементов в коллекции.

- Вставка элемента в std::vector также занимает O(1) времени.

Таким образом, общая сложность алгоритма будет O(N * log M), где N - количество строк во входном файле, M - количество различных переменных..

Задача №D «Профессор Хаос»

```cpp
#include <iostream>

int main() {
    int a, b, c, d, n;
    std::cin >> a >> b >> c >> d >> n;
    int currentA = a;
    for (int i = 0; i < n; ++i) {
        int result;
        switch (a * b > c) {
            case true:
                result = a * b - c;
                break;
            case false:
                result = 0;
                break;
        }
        switch (result >= d) {
            case true:
                a = d;
                break;
            case false:
                a = result;
                break;
        }
        if (currentA == a)
            break;
        else
            currentA = a;
    }
    std::cout << a;
    return 0;
}
```

Пояснение к примененному алгоритму:

1) В начале у профессора есть определенное количество бактерий a, которые каждый день делятся на b новых бактерий. После этого из них используется c для опытов и остальные оставляются. Однако в контейнер можно поместить не более d бактерий.

- a: количество опасных бактерий, которые есть у профессора Хаоса в начале первого дня эксперимента.

- b: количество новых бактерий, образующихся из одной особо опасной бактерии каждый день.

- c: количество бактерий, которые используются для опытов и затем уничтожаются.

- d: максимальное количество бактерий, которое можно поместить в контейнер. Если число оставшихся бактерий больше d, то в контейнер помещаются d бактерий, а остальные уничтожаются.

- n: номер дня, на который профессор Хаос хочет узнать количество опасных бактерий в контейнере.

2) Моделирование каждого дня эксперимента с помощью цикла с бактериями, проводимым профессором Хаосом.

3) Внутри цикла вычисляется количество оставшихся бактерий после разделения и проведения опытов. Если это количество больше или равно d, то в контейнер помещается d бактерий, иначе все оставшиеся.

4) Программа повторяет эти действия n раз или до тех пор, пока количество бактерий не перестанет изменяться и выводит в консоль количество особо опасных бактерий в контейнере после n-го дня эксперимента.

Сложность:

- Считывание входных данных и их обработка занимают O(1) времени.
- Цикл проходит n раз, поэтому сложность цикла O(n).
- Каждая итерация цикла выполняется за константное время O(1).

Таким образом, общая сложность алгоритма составляет O(n), где n - количество дней эксперимента.