Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Алгоритмам и структурам данных»

Базовые задачи / Timus

Выполнил:

Студент группы P3234

Баянов Р. Д.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Яндекс контест

Задача A. Агроном-любитель

Ход решения:

Для решения этой задачи я запоминаю два предыдущих цветках при каждой итерации по списку цветов и сравниваю их с текущим. Если мы встречаем три одинаковых цветка подряд, мы проверяем нашли ли мы строку, в которой нет повторений больше чем была до этого, если да, то присваиваем максимальной длине текущую длину строки. И, конечно же, сбрасываем в 2 текущий счётчик строки и обозначаем новое начало для нашей новой строки, которую мы будем считать. Иначе увеличиваем счётчик строки на 1. После массива цветов мы делаем ещё одну проверку на максимальность текущей строки, так как массив мог закончиться не встретив три цветка подряд. В конце выводим начало строки и к началу строки прибавляем максимальную длину нашей строк, чтобы получить номер последнего цветка в самой длинной строке без трёх идущих подряд цветков.

Оценка сложности решения:

Время: O(N)

Память: O(N)

Код решения:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    int count;
    cin >> count;
    vector<int> flowers(count);
    for (int i = 0; i < count; i++) {
        cin >> flowers[i];
    }

    int prevNum = -1;
    int prevPrevNum = -1;
    int maxLength = 0;
    int curLength = 0;
    int index = 0;
    int startIndex = 1;
    int maxStartIndex = 1;
    for (int flower : flowers) {
        if (flower != prevNum || flower != prevPrevNum) {
            curLength++;
        }
```

```
        else {
            if (maxLength < curLength) {
                maxLength = curLength;
                maxStartIndex = startIndex;
            }
            curLength = 2;
            startIndex = index;
        }
        prevPrevNum = prevNum;
        prevNum = flower;
        index++;
    }
    if (maxLength < curLength) {
        maxLength = curLength;
        maxStartIndex = startIndex;
    }
    /*cout << maxLength << endl;*/
    cout << maxStartIndex << " ";
    cout << maxStartIndex + maxLength - 1;
}
```

Задача B. Зоопарк Глеба

Ход решения:

Для решения этой задачи я использовал три стека. Один стек для индексов животных, один для индексов ловушек и другой для символов из строки. Также я завожу массив result_ids, который мы постепенно будем заполнять индексами животных. Алгоритм работает так, что мы для каждой итерации в самом начале определяем мы встретили ловушку или животное и, встретив ловушку мы увеличиваем счётчик ловушек на 1 каждый раз и кладём этот счётчик на стек с индексами ловушек, а встретив животное мы вычитаем из текущего элемента строки количество подсчитанных ловушек и получаем номер животного и сразу же кладём этот номер на стек с индексами животных. После всего этого мы смотрим, если же стек с символами пустой или же буквы одинаковы и на стеке и в текущей итерации строки или же если буквы в увеличенном регистре разные, то мы просто кладём их на стек с символами. Другая ситуация происходит, если в верхнем регистре обе буквы (и на стеке и текущая в строке) совпадают, то мы убираем вершинку из трёх стеков, но перед этим в массив result_ids, по номеру ловушки записываем номер животного. И в конце мы просто проходимся по всему массиву и получаем наш ответ.

Оценка сложности решения:

Время: O(N)

Память: O(1)

Код решения:

```
#include <iostream>
#include <string>
#include <stack>
using namespace std;

int main()
{
	string input ;
	cin >> input;

	const int size = input.size();
	const int n = size / 2;
	stack<char> letters;
	stack<int> traps_ids;
	stack<int> animals_ids;
	string result = "Possible\n";
	int count = -1;
	int* result_ids = new int[n];

	for (int i = 0; i < size; i++) {
		if (islower(input[i])) {
			animals_ids.push(i - count + 1);
		}
		else {
			count++;
			traps_ids.push(count + 1);
		}
		if (letters.empty() || input[i] == letters.top()) {
			letters.push(input[i]);
		}
		else if (toupper(input[i]) == toupper(letters.top())) {
			result_ids[traps_ids.top() - 1] = animals_ids.top()  - 1;
			letters.pop();
			traps_ids.pop();
			animals_ids.pop();
		}
		else {
			letters.push(input[i]);
		}
	}

	if (letters.empty()) {
		cout << result;
		for (int i = 0; i < n; i++) {
			cout << result_ids[i];
			if (i + 1!= n) {
				cout << " ";
			}
		}
	}
	else {
		cout << "Impossible";
	}
	return 0;
}
```

Задача C. Конфигурационный Файл

Ход решения:

Для решения этой задачи я использовал две структуры данных, а именно: неупорядоченную мапу, ключами которой являются строки, а значения целочисленные стеки, стек с листами, в которых находятся строки. Мы в мапе будем хранить стек для каждой нашей переменной, а в этом стеке будут по порядку в ввиду изменений храниться значения этой самой нашей переменной. А собственно в стеке будем хранить списки строк так же ввиду изменений и потом с помощью строки находящейся на вершине по окончании каждого блока мы будем обращаться к мапе для восстановления значений на переменных до начала блока. Мы начинаем считывать строки и запускаем цикл while пока не встретим пустую строку. На каждой итерации мы проверяем встретили ли мы фигурную скобку, если открывающая, то закидываем новый лист строк на стек, если закрывающая восстанавливаем старые значения переменных. Если же мы видим выражения для присваивания переменных, то мы пользуемся мапой, и на стек для каждой переменной кладём новое значение (либо целочисленное значение, либо из мапы добываем значение переменной справа от знака “=”). И не забываем выводить значения при встрече с выражением вида var1=var2.

Оценка сложности решения:

Время: O(N)

Память: O(N)

Код решения:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <stack>
#include <fstream>
#include <list>
using namespace std;

int main()
{
	unordered_map<string, stack<int>> variables;
	stack<list<string>> variablesStack;
	string str;
	list<string> l;
	variablesStack.push(l);
	ifstream input("input.txt");
	if (!input.is_open()) {
		return 1;
	}
	while (getline(input, str)) {
		if (str == "{") {
```

```
                list<string> arr;
                variablesStack.push(arr);
            }
            else if (str == "}") {
                for (const string& variable : variablesStack.top()) {
                    variables[variable].pop();
                }
                variablesStack.pop();
            }
            else {
                int symbol = str.find('=');
                string var1 = str.substr(0, symbol);
                string var2 = str.substr(symbol + 1);

                if (isdigit(var2[0]) || var2[0] == '-') {
                    variables[var1].push(stoi(var2));
                }
                else {
                    if (variables.find(var2) == variables.end() || vari-
ables[var2].empty()) {
                        variables[var2].push(0);
                    }
                    variables[var1].push(variables[var2].top());
                    cout << variables[var1].top() << endl;
                }
                variablesStack.top().push_front(var1);
            }
        }
        input.close();
        return 0;
}
```

Задача D. Доктор Хаос

Ход решения:

Для решения этой задачи я решил использовать простую рекурсию. Я создал отдельную функцию под названием experiment. На вход эта функция принимает наши параметры эксперимента. При каждом вызове этой функции мы проверяем не закончилось ли количество наших дней k, если закончилось то мы возвращаем значение бактерий и заканчиваем программу, если же всё еще мы не на последнем дне нашего эксперимента, то с помощью простых нетрудных описанных в задаче действий мы каждый раз получаем новое кол-во бактерий и уменьшаем кол-во дней, чтобы дальше идти по эксперименту. Если же мы видим что кол-во бактерий после использования меньше c, мы возвращаем ноль, если же нет то рекурсивно вызываем функцию experiment. И также не забываем проверить не равно ли кол-во бактерий в конце какого то дня предыдущему кол-ву бактерий, так как это вызовет нарушение глубины рекурсии и алгоритм естественно можно закончить раньше.

Оценка сложности решения:

Время: O(N)

Память: O(1)

Код решения:

```
#include <iostream>
using namespace std;

int experiment(int a, int b, int c, int d, int k) {
    int l = a;
    if (k > 0) {
        k--;
        if (a * b <= c) {
            return 0;
        }
        a = max(min(a * b - c, d), 0);
        if (a == l) {
            return a;
        }
        return experiment(a, b, c, d, k);
    }
    else {
        return a;
    }
}

int main()
{
    int a, b, c, d, k;
    cin >> a >> b >> c >> d >> k;

    cout << experiment(a, b, c, d, k);
    return 0;
}
```

Timus

Задача 1005. Куча Камней

Ход решения:

Для решения этой задачи я использовал двоичный код (грубо говоря полный перебор). Мы получаем несколько разных кодов до 2^(N-1) и проходимся по ним. 0 – i-ый камень в 0-ую кучу, 1 – i-ый камень в 1-ую кучу. Для каждого определённого случая мы подсчитываем разницу камней и сравниваем с минимальной найденной разницей, если меньше то сохраняем, если нет то идём дальше. Тем самым мы сможем найти минимальную разницу.

Оценка сложности решения:

Время: O(2^N *N)

Память: O(N)

Код решения:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>
using namespace std;

int main()
{
	int size;
	cin >> size;

	vector<int> stones(size);
	for (int i = 0; i < size; i++) {
		cin >> stones[i];
	}

	int minDiff = INT_MAX;
	for (int code = 0; code < (1 << (size - 1)); code++) {
		cout << code << " ";
		int sum0 = 0;
		int sum1 = 0;
		for (int i = 0; i < size; i++) {
			cout << " " << (code << i) << endl;
			if (((code >> i) & 1) == 0) {
				sum0 += stones[i];
			}
			else {
				sum1 += stones[i];
			}
		}
		minDiff = min(minDiff, abs(sum0 - sum1));
	}
	cout << minDiff;
	return 0;
}
```

Задача 1155. Дуоны

Ход решения:

Для решения этой задачи заметим, что если две точки отдалены друг от друга ещё одной точкой, то эти точки никак не удалить без появления дуонов в других местах. Поэтому выражение A + C + F + H == B + D + E + G должно быть выполнено иначе выполнить удаление невозможно. Если условие выполнилось, то мы идём по бесконечному циклу, и с помощью условных конструкций удаляем дуоны, и в конце каждого цикла смотрим не стало ли кол-во дуонов во всех наших камерах 0, если нет, то продолжаем удаление, если да выходим из цикла и завершаем программу.

Оценка сложности решения:

Время: O(1)

Память: O(1)

Код решения:

```
#include <iostream>
#include <unordered_map>
using namespace std;

int main()
{
	unordered_map<char, int> cameras;
	for (int i = 0; i < 8; i++) {
		cin >> cameras['A' + i];
	}
	if (cameras['A'] + cameras['C'] + cameras['F'] + cameras['H'] != cameras['B'] + cam-
eras['D'] + cameras['E'] + cameras['G']) {
		cout << "IMPOSSIBLE";
	}
	else {
		int flag = 0;
		while (true) {
			if (cameras['B'] > 0 && cameras['C'] > 0) {
				cout << "BC-" << endl;
				cameras['B']--;
				cameras['C']--;
			}
			if (cameras['B'] > 0 && cameras['A'] > 0) {
				cout << "BA-" << endl;
				cameras['B']--;
				cameras['A']--;
			}
			if (cameras['B'] > 0 && cameras['F'] > 0) {
				cout << "BF-" << endl;
				cameras['B']--;
				cameras['F']--;
			}
			if (cameras['B'] > 0 && cameras['H'] > 0) {
				cout << "CG+" << endl;
				cout << "HG-" << endl;
				cout << "BC-" << endl;
				cameras['B']--;
				cameras['H']--;
			}
			if (cameras['A'] > 0 && cameras['E'] > 0) {
				cout << "AE-" << endl;
				cameras['A']--;
				cameras['E']--;
			}
			if (cameras['A'] > 0 && cameras['D'] > 0) {
				cout << "AD-" << endl;
				cameras['A']--;
				cameras['D']--;
			}
			if (cameras['D'] > 0 && cameras['C'] > 0) {
				cout << "DC-" << endl;
				cameras['D']--;
				cameras['C']--;
			}
			if (cameras['D'] > 0 && cameras['H'] > 0) {
				cout << "DH-" << endl;
```

```
                                        cameras['D']--;
                                        cameras['H']--;
                                }
                                if (cameras['G'] > 0 && cameras['C'] > 0) {
                                        cout << "GC-" << endl;
                                        cameras['G']--;
                                        cameras['C']--;
                                }
                                if (cameras['G'] > 0 && cameras['H'] > 0) {
                                        cout << "GH-" << endl;
                                        cameras['G']--;
                                        cameras['H']--;
                                }
                                if (cameras['G'] > 0 && cameras['F'] > 0) {
                                        cout << "GF-" << endl;
                                        cameras['G']--;
                                        cameras['F']--;
                                }
                                if (cameras['E'] > 0 && cameras['F'] > 0) {
                                        cout << "EF-" << endl;
                                        cameras['E']--;
                                        cameras['F']--;
                                }
                                if (cameras['E'] > 0 && cameras['H'] > 0) {
                                        cout << "EH-" << endl;
                                        cameras['E']--;
                                        cameras['H']--;
                                }
                                if (cameras['D'] > 0 && cameras['F'] > 0) {
                                        cout << "AE+" << endl;
                                        cout << "DA-" << endl;
                                        cout << "FE-" << endl;
                                        cameras['D']--;
                                        cameras['F']--;
                                }
                                if (cameras['C'] > 0 && cameras['E'] > 0) {
                                        cout << "DH+" << endl;
                                        cout << "CD-" << endl;
                                        cout << "EH-" << endl;
                                        cameras['C']--;
                                        cameras['E']--;
                                }
                                if (cameras['A'] > 0 && cameras['G'] > 0) {
                                        cout << "BF+" << endl;
                                        cout << "AB-" << endl;
                                        cout << "GF-" << endl;
                                        cameras['A']--;
                                        cameras['G']--;
                                }
                                flag = 0;
                                for (int i = 0; i < 8; i++) {
                                        if (cameras['A' + i] != 0) {
                                                flag = 1;
                                                break;
                                        }
                                }
                                if (flag == 0) {
                                        break;
                                }
                        }
                }
                return 0;
}
```