Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Алгоритмам и структурам данных»

Timus

Выполнил:

Студент группы P3231

Савон Г.К.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2021

Задача №1005 «Куча камней»

```
#include <iostream>
#include <vector>
#include <math.h>
#include <stdlib.h>

using namespace std;

vector<int> rocks;

int summing(int i, int sum1, int sum2) {
  int rock = rocks[i];
  if (i == rocks.size() - 1) {
    return abs(rock - abs(sum1 - sum2));
  } else {
    return min(summing(i + 1, rock + sum1, sum2),
               summing(i + 1, sum1, sum2 + rock));
  }
}

int main() {
  int n;
  int rock;
  cin >> n;
  for (int i = 0; i < n; i++) {
    cin >> rock;
    rocks.push_back(rock);
  }
  cout << summing(0, 0, 0);
  return 0;
}
```

Пояснение к примененному алгоритму:

O(2^(logn))

Считываем количество камней. Считываем и запоминаем все их значения.

После вызываем функцию, которая рекурсивно вызывает себя же для следующего камня, предположив, что текущий положен в одну из куч. И затем возвращает минимальную по модулю разницу между весом камней в кучах.

(с ограничением на последний камень, естественно, там она сравнивает, положив в какую кучу разница будет меньше и соответственно возвращает это значение)

Задача №1296 «Гиперпереход»

```
#include <iostream>

using namespace std;

int main() {
  int n;
  int max_sum = 0;
  int now_sum = 0;
  int number;
  cin >> n;
  for (int i = 0; i < n; i++) {
    cin >> number;
    now_sum += number;
    if (now_sum > max_sum)
      max_sum = now_sum;
    if (now_sum < 0)
      now_sum = 0;
  }
  cout << max_sum;
  return 0;
}
```

Пояснение к примененному алгоритму:

O(n)

Считываем длину последовательности гипер-интенсивностей и поочередно читая значения совершаем следующие действия:

-добавляем к текущей сумме новое значение

-сравниваем с максимальной имеющейся суммой (если превышает – обновляем максимальную сумму)

-если значение текущей суммы становится отрицательным, обнуляем его (т.е считаем, что все обработанные элементы мы далее учитывать не будем).

А теперь методом разделяй-властвуй(оно даже сдалось на тимусе)

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

using namespace std;

vector<int> numbers;

vector<int> recursion(int start, int finish)
{ // 0-левый край(макс), 1 - правый край(макс), 2-вцелом, 3-вся сумма
    vector<int> to_return;
    if (start == finish) {
        to_return.push_back(numbers[start]);
        to_return.push_back(numbers[start]);
        to_return.push_back(numbers[start]);
        to_return.push_back(numbers[start]);
    }
    else {
        vector<int> return_left = recursion(start, (finish + start) / 2);
        vector<int> return_right = recursion((finish + start) / 2 + 1, finish);
        to_return.push_back(max(return_left[0], max((return_left[3] + max(return_right[0], return_right[3])), 0)));
        to_return.push_back(max(return_right[1], max(return_right[3] + max(return_left[1], return_left[3]), 0)));
        to_return.push_back(max(return_left[2], max(return_right[2],
                                                    max(max(return_left[3], return_left[1]) + max(return_right[0], return_right[3]), 0))));
        to_return.push_back(return_left[3] + return_right[3]);
    }
    return to_return;
}

int main()
{
    int n;
    cin >> n;
    if (n == 0)
        cout << 0;
    else {
        int number;
        for (int i = 0; i < n; i++) {
            cin >> number;
            numbers.push_back(number);
        }
        cout << recursion(0, n - 1)[2] << endl;
    }
    return 0;
}
```

Пояснение к примененному алгоритму:

O(2^log(n))

Считываем длину последовательности гипер-интенсивностей, запоминаем в векторочек и отправляем в функцию (которая принимает края используемой части(номера чисел), а возвращает вектор со значениями, эти значения описаны в комментарии)

А функция делает вот что:

-если длина обрабатываемого куска – 1 – возвращает обратно просто 4 значения обрабатывающегося числа (правда тут есть ошибка одна(она легко исправима), из-за которой крайний случай обрабатывается неправильно (при одном отрицательном числе), но такого теста видимо не было, хаха)

-иначе – делим имеющуюся часть пополам и для каждой из них вызываем ту же функцию. Из вернувшихся значений составляем ответ для текущей функции