Университет ИТМО
Кафедра ВТ

Задача 4 (Divide And Conquer)

Алгоритмы и Структуры Данных

Выполнил: Федоров Сергей
Группа: P3212

Санкт-Петербург
2020 г.

- # Задача 4 - Гиперпереход - 1296

Прелюдия: Долго не мог решить именно путем Divide And Conquer. Получалось с помощью ДП и рекурсии. Но наверное в итоге получил что-то такое.

Решение:
Имеем массив заданных чисел, если за отправную точку считать центр, то максимальная последовательность может либо проходить через центр, либо быть слево, либо справо.
Как и гласит D&C, рекурсивно спускаемся из центра влево и вправо, до того момента пока не дойдем до индивидуальных чисел.
Мержим числа путем выбора максимальной последовательности от центра (включительно) влево и от центра (не включительно) вправо и сложением их сумм.
Выбираем максимум из трех чисел: левого числа, правого числа или полученной merge суммы.

Сложность:
Сложность от рекурсивных вызовов T(n/2)
Сложность от мержа чисел в центре O(n/2)

T(n) = 2T(n/2) + O(n)

Пример:

```
(5  5  3  -10  -19  3  4  5) max_merge_sum = 3 (left) + (-7) (right) = -4
(5  5  3  -10)(-19  3  4  5) max_merge_sums (left) = 13 | (right) = 12 ← 13 is right!
(5  5)(3  -10)(-19  3)(4  5) max_merge_sums (1) = 10, (2) = 3, (3) = 3, (4) = 9
(5)(5)(3)(-10)(-19)(3)(4)(5) max_merge_sums (1) = 10, (2) = 3, (3) = 3, (4) = 9
```

Код:
```c
//
// Created by Sergey Fedorov on 12/03/2020.
//

#include "../../utils/common_utils_string.c"

int max(int a, int b){
    return (a > b) ? a : b;
}

int divide_n_conquer(int* values, int a, int b){
    if (a ≠ b) {
        int left_side = divide_n_conquer(values, a, a + (b-a) / 2);
        int right_side = divide_n_conquer(values, a + (b-a) / 2 + 1, b);

        // Computing mid crossing sum
        int m = (a + b) / 2;

        int temp_sum = 0;
        int mid_left_sum = 0;
        for (int i = m; i ⩾ a; i--)
        {
            temp_sum = temp_sum + values[i];
            mid_left_sum = max(mid_left_sum, temp_sum);
        }

        temp_sum = 0;
        int mid_right_sum = 0;
        for (int i = m+1; i ⩽ b; i++)
```

```
        {
            temp_sum = temp_sum + values[i];
            mid_right_sum = max(mid_right_sum, temp_sum);
        }

        return max(mid_left_sum + mid_right_sum, max(left_side, right_side));
    } else {
        return values[a];
    }
}

int main(){
    // 0 ⩽ N ⩽ 60000 ⟹ max 5 sym
    int n = str_to_int(get_line(5));

    int ps[n];

    for (int i = 0; i < n; ++i) ps[i] = str_to_int(get_line(6));

    if (n > 0){
        printf("%i", divide_n_conquer(ps, 0, n-1));
    } else {
        printf("0");
    }
}
```