09-hw-PA

vladrus13rus

November 2021

1 Оптимизация константы

Рисунок 1. Я, когда понял, что можно не разбирать полностью соответствие строки шаблону, что бы решить дз

Давайте напишем код, и поймем, что же мы можем оптимизировать.

```
(1)  for i = 0..|s|:
(2)    ID[0][i] = s[i]
(3)  for it = 1..log(|s|):
(4)    pfor i = 0..|s|:
(5)      BB[ID[it][i]][ID[it][i + 2^it]]] = i
(6)    pfor i = 0..|s|:
(7)      ID[it + 1][i] = BB[ID[it][i]][ID[it][i + 2^it]]]
```

Понятно, что строки 1-2 ничему не мешают, их можно оставить. Так же понятно, что внешний цикл никак не убрать, так как результаты шага x влияют на результаты шагов $x + 1$. Таким образом, мы должны сделать так, что бы внутренний цикл был сложности не более $log(|s|)$ span. Так же мы не должны сломать work – его составляет составляет $|s|$.

1 способ) Можно просто запихивать все в хешмапу, там достанется за O(1) и загрузится за O(1). То есть обращение будет как в массив, просто вместо пары индексов мы пихаем туда по хешу. Памяти там будет не больше чем O(nlogn)

2 способ) Довольно логично, что мы должны воспользоваться каким то параллельным обходом (pfor, map, reduce, scan, filter) или их некоторой комбинацией. Заметим, что ни на какую сортировку чистых пар у нас времени не хватит - их $nlogn$. Так же следует заметить, что количество различных пар ID может быть равно n^2, то есть не следует явно сохранять все пары по разделам. Мы можем провести сортировку подсчетом только для первого индекса или только для второго. Тогда мы узнаем, какое количество индексов выпадает на каждую

одинаковую первую часть строки. Можно так же сделать со второй частью строки. ~~Построим граф и просто построим парсеч.~~. Совершим ход конем - сделаем первое, затем пройдем по массиву подсчетов и собирая с помщью ассоциативной операции присоеднинить двусвязный список к другому можем собрать такой массив - a[i] = все элементы, первая строка которых с индексом i. То есть по сути мы делаем radix sort внутри которого есть counting sort. Выглядит отвратительно, Да. Мы знаем $a[i].length$. Сделам по ним массив префикс сумм $sum[i]$ - индекс первой по индексу строки для строк из $a[i]$. То есть все строки из $a[i]$ будут иметь индексы от $sum[i]$ до $sum[i+1]$. Аналогично сделаем в каждом массиве из $a[i]$. То есть radix sort будет по двум числам от 1 до n. Внутри каждого radix sort будет counting sort от 1 до n. Заметим, что любые такие сортировки идут за линейное время. То есть radix sort будет за n, сумма всех counting sort будет тоже n, так как количество различных индексов будет n. Span тоже будет за $logn$, так ка мы всегда будем пользоваться pfor (количество этапов в radix sort будет 2, то есть константа). Память будет затрачена по количеству элементов, то есть $n * log(n)$