04-hw-PA

vladrus13rus

October 2021

1 Сортировочка

Задание: Вам дано n пар (k_i, v_i). *Известно, что* $0 <= k_i < logn$. *Отсортируйте пары по ключу за* $O(n)$ *work и* $O(log^2 n)$ *span*

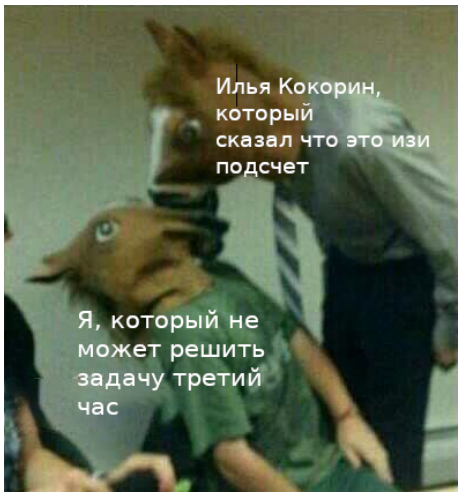

Рисунок 1. Как я себя чувствовал

Действительно, давайте применим просто сортировку подсчетом. Разобьемся parallel-for за $O(log(n))$, заведем массивчик на $log(n)$ элементов, и каждый прибавить по 1 там где нужно...

К сожалению (или к счастью), это так не работает.

Давайте решим это как то без CAS и всякого такого. Но идея хорошая! Конечно же, первым делом давайте разделим все parallel-for'ом. Итог: у нас есть n потоков и каждый держит свое число. Теперь мы хотим совместить все числа в один массивчик. Каждый создает свой массивчик размера $log(n)$ и когда они сливаются, один из потоков сливает два массива в один за $log(n)$. Сколько же будет работы?

У нас будет $O(n)$ слияний, каждое по $log(n)$. Высота дерева - $log(n)$, то есть:

$span = O(log(n)) \cdot O(log(n)) = O(log^2(n))$

$work = O(n) \cdot O(log(n) = O(nlog(n))$

тут должен быть суперкрутой мем, но я не смогу вставить гифку

Может попробуем разкладываться не до 1 элемента? Может лучше до $log(n)$?

Рисунок 2.

Посмотрим на итоги нашей плодотворной работы. Мы делимся пополам наши тредики до тех пор, пока их не станет $O(\frac{n}{log(n)})$. Ура, теперь берет каждый по $log(n)$ и сортирует их втупую за $log(n)$ (работа становится $O(n)$, каждый элемент встает в массив ровно один раз). Теперь давайте пойдем наверх. Теперь у нас будет $O(\frac{n}{log(n)})$ слияний, каждый по $O(log(n))$, высота дерева не больше $O(log(n))$. Итог: $O(n)$ work, $O(log^2(n))$ span.

2 Эйлеров обход?

Задание: Дано подвешенное дерево за вершину r с уже построенным Эйлеровым обходом. Нужно для каждой вершины найти её глубину в дереве за $O(n)$ work и $O(polylogn)$ span

- Давайте пронумеруем все ребра от 1 до n. **Как?** Просто сделаем list ranking по обходу графа. Расстояние до последнего ребра будет его номером (можем для успокоения души сделать номером $n - listranking + 1$).

  $work = O(n), span = O(polylog(n))$

- Давайте найдем предка для каждой вершины. **Как?** А вот это было на лекции!

  $work = O(n), span = O(polylog(n))$

- Найдем, какие ребра нас спускают по дереву, а какие поднимают. **Как?** Теперь, раз мы знаем, какое ребро ведет из предка, поставим там значение +1, а иначе -1. Это делается за проход по всем ребрам, то есть за $O(n)$, но сила параллелизма дает нам parallel-for и мы успешно делаем это!

  $work = O(n), span = O(polylog(n))$

- Найдем высоты вершин. **Как?** Возьмем все ребра в эйлеровом обходе и сделаем по ним префиксные суммы. Заметим, что если мы спустились по ребру, то заработали +1, а если поднялись по ребру, то -1. Что то похожее на высоту, не так ли?) Просто посмотрим на входящее ребро и его величину. Оно будет высотой. Мы решили задачу!

  $work = O(n), span = O(polylog(n))$

Рисунок 3. Когда увидел, что кто то из сдающих не пишет точную ассимтотику алгоритмов, а просто пишет подогнанную под ответ