04-hw-PA

vladrus13rus

October 2021

1 Парсинг выражений

Воспользуемся первой задачей, проигнорируем для каждого нескобочного символа его. У нас есть для каждой скобочки есть пара! Теперь хочу сделать так, что бы можно было попарсить выражение. Заведем для каждого символа свой тредик за log(n) parallel-for'ом.

Каждая открывающая скобочка говорит символу слева кто она такая. Если там какой либо знак операции, мы говорим этому знаку операции, что теперь мы его правая операция. Если там скобочка, то мы говорим ей, что мы левый ребенок операции, что содержит эта скобочка (при желании, можем перейти к правой скобочке, узнать, кто такая наша операция, и сказать, что мы ее левая скобочка)

Если мы закрывающая скобочка, то говорим символу справа, кто мы такие. Если там скобочка, то мы говорим ей, что мы правый ребенок операции, что содержит эта скобочка (опять же, можем повторить подобную операцию - перейти к парной скобочке, перейти к ее операции, и сказать, что мы правая скобочка)

Если мы число, то мы ведем себя как скобочка. То есть если нас спрашивают скобочка ли мы, мы говорим, что мы скобочка.

Итог: для каждой скобочки-операции мы знаем, кто правая и левая ее операции. Дерево построено. $work = O(n) + work(findPairBracket)$, $span = O(1) + span(findPairBracket)$. То есть, если первая задача решена правильно, то $work = O(n)$, $span = O(sqrt(n)polylog(n))$

2 Парсинг скобочек

У нас есть правильная скобочная последовательность. Первое – я хочу посчитать глубину каждой скобочки. То есть, как только скобочка началась - глубина увеличилась, иначе – закончилась.

Рисунок 1. Это я, когда решал задачу поиска глубины

Давайте поставим +1 там, где скобочка открывается, и -1, там где скобочка закрывается. Теперь сделаем префиксные суммы. Теперь на позиции i находится глубина i скобочки.

Теперь наша задача это стабильно отсортировать это все! В итоге будет так: каждая левая скобочка в этом массиве иметь правее себя свою пару. Почему? Потому что ее пара будет иметь ту же глубину и будет первой, кто будет иметь ту же глубину. Давайте сделаем как нибудь сортировку. Конечно же, нам нужно будет применить магическую accelerating cascades. Разделим все на куски размера $log(n)$ и посортим каждое. Заметим, что каждое из них отличается не больше чем на $log(n)$, так как соседние отличаются не более, чем на 1. Давайте найдем минимум в каждом с помощью алгоритма из лекции ($work = O(log(n) \cdot log(log(n))), span = O(log(log(n)))$). Мысленно вычтем из каждого по минимуму в каждом из блоков и сделаем сортировку подсчетом ($work = O(\frac{n}{log(n)} \cdot log(n)), span = O(\frac{n}{log(n)})$). Теперь обьединяем их по $sqrt(n)$ и сортируем.