02-hw-CS

vladrus13rus

September 2021

1 Задача о какой то сумме

Условие: Доказать, что $\sum_{i=0}^{logn} log(n/2^i) = Theta(log^2 n)$.

Решение: $\sum_{i=0}^{log(n)} log(\frac{n}{2^i}) = log(\frac{n^{log(n)}}{\prod_{i=0}^{log(n)} 2^i}) = log(\frac{n^{log(n)}}{\frac{1}{2}(log^2(n)+log(n))}) = log(n^{log(n)}) - log(\frac{1}{2}(log^2(n) + log(n))) = log^2(n) - log(\frac{1}{2}(log^2(n) + log(n))) = Theta(log^2(n))$

2 Задача о fork-join-scan

Условие: Написать код алгоритма scan в fork-join модели с $O(n)$ work и $O(logn)$ span

Решение: На самом деле код был почти полностью написан на лекции, нужно было только обработать крайние случаи

```
up(a, l, r, f):
  if (r - l == 0):
    return a[l]
  m = (l + r) / 2
  fork2join(
    left <- up(a, l, m, f)
    right <- up(a, m, r, f)
  )
  tree[(l, r)] = left
  return f(left, right)

down(a, l, r, f, left):
  if (r - l == 0):
    b[l] = f(left, a[l])
  else:
    m = (l + r) / 2
    down(a, l, m, f, left)
    down(a, m, r, f, f(left, tree[(l, m)]))
```

3 Задача о простых

Условие: Опишите алгоритм нахождения всех простых до N за $O(NloglogN)$ work и $O(polylogN)$ span. (Желательно за $O(logN \cdot loglogN)$ span). Тут можно пользоваться parallel for, map, scan и filter. Псевдокод даже необязательно.

Решение: Что бы решить задачу, нам вообще нужно сначала понять, что мы можем распараллелить. Возможно, что все части алгоритма зависят от друг друга, и мы никак не сможем ничего с этим поделать. Конечно же, мы можем как то распарелллелить внутреннюю часть, но это пока что нам ничем не поможет (внешний цикл по n, дает как минимум O(n) span). Давайте разберемся во внешнем цикле. Какие величины не зависят от друг друга? Понятно, что если два числа стоят рядом, то не могут делиться на друг друга, а значит, что не могут сделать друг друга составными.

Отлично, мы можем взять параллельный цикл (одна итерация пойдет по четным, нечетные захватим в той же итерации) и ускорить алгоритм как бы вдвое. Но... ассимтотика не изменилась... (Доказательство ассимтотики Решета Эратосфена основана на рукомахании и интегралах, у нас оно совсем не будет отличаться).

Давайте еще немного подумаем. Быть может, не только соседние числа независимые (в плане делимости)? Конечно же, если мы берем x, и число y, $\frac{x}{2} < y < x$, то они тоже независимы! Опять же, мы помним, что если у нас прошло несколько (y) итераций, и затем мы проходим по x числам решетом, то время работы $O(xlog(log(n)) = time(x) - time(y)$. То есть, если мы распределим внутренний цикл поровну на все треды, то выйдет loglogn span. То есть мы берем сначала числа 2..3, проводим для них решето параллельно, затем берем 4..7, и так далее... Количество таких итераций будет logn, а значит, итоговое время - $O(log(n)log(log(n)))$. Недостаточно!

Может быть, есть такие случаи, когда мы можем не всегда брать только независимые числа? Может ли быть такое, что мы его уже когда то взяли? Конечно же да! Допустим, у нас есть $z = x * y$, $x < y$. Тогда мы узнаем, что z составное сначала из x, а потом зачем то из y. Давайте попробуем сделать так, что бы мы не получали z из y, то есть назначить их в один паралельный проход. Как же должны выглядеть z, x, y? То есть, если y $>$ всех возможных таких x, то положим их рядом с z. Понятно, что самый большой подобный x является корнем z. То есть, нам нужно начинать с какого то z и идти вплоть до z^-1 в одной итерации. Посмотрим, сколько же таких кусочков мы сможем получить. Каждый раз количество увеличвается с i до i^2. Тогда если количество кусочков d:

$2^{2^d} = n$
$2^{2^d} = 2^{log(n)}$
$2^{2^d} = 2^{2^{log(log(n))}}$

Итого, количество кусочков будет $log(log(n))$. Каждый кусочек обрабатывается за $O(log(log(n)))$, значит, итоговая ассимтотика будет $O(log^2(log(n)))$. При этом, дополнительных действий мы не делаем, значит, work остается таким же

Рисунок 1. Секретный мем. Когда перепутал fork2join и forkjoin