Домашнее задание 3
по алгебреLaTeX

Попов Николай

5 марта 2018 г.

№1

1)SX(5) =SX(101)= XSSX, y меняется так: 1 3 3^2 3^4 3^5 Значит, F(3,5)=3^5
2)Реализуется здесь алгоритм быстрого возведения в степень слева направо(биты просматриваются от старшего к младшему). Этот алгоритм содержит шаги:
а)Перевести показатель степени n в двоичный вид (у нас в буквенную строку). б)Проходясь от старшего бита к младшему делать:
Если бит - это 1, то текущий результат возводится в квадрат (выполяется S) и затем умножается на x (выполняется X).
Если же 0, то текущий результат просто возводится в квадрат (операция S).
3)Корректность следует из формулы:
$m = 2^k n_k + 2^{k-1} n_{k-1} + ... + 2n_1 + n_0$, n_i - ноль или единица. Будем выносить 2 за скобки, получим:

$$x^m = x^{((...((n_k*2+n_{k-1})*2+n_{k-2})*2+...)*2+n_1)*2+n_0} =$$
$$= ((...(((x^{n_k})^2) * x^{n_{k-1}})^2...)^2 * x^{n_1})^2 * x^{n_0}$$

4)Подсчитаем число шагов, необходимых для "спуска"от значения показателя до 1 делением на 2 если число четно или вычитанием единицы если нечетно. Их количество порядка $\log m$, считая, что показатель - целое число и операции - константы по времени.

№2

Двигаясь от отрезка вложенного во все остальные (самого короткого) к более широким отрезкам (изнутри) можем сказать, что вне 1 отрезка точки покрыты n-1 отрезками (точки также внутри самого большого отрезка), вне 2 - n-2, тогда вне n/3 отрезка - (2/3)n отрезками,а вне n/3+1 отрезка- (2/3)n-1. Т.е. искомые точки находятся между левыми и правыми концами n/3 отрезка и n/3+1 отрезка. Из того, что отрезки строго вложены следует, что их длины образуют строго возрастающую последовательность. Значит, n/3 отрезок будет n/3 порядковой статистикой в массиве длин отрезков, которую посчитаем для каждой пары концов отрезков. Ее найдем за линейное время, а концы n/3+1 отрезка отрезка можно найти либо как n/3+1 статистику по длинам, либо в массиве всех концов найти максимальное число не превосходящее левого конца n/3 отрезка и минимальное, не меньшее его правого конца (за линейное время). В итоге, имея пары концов этих отрезков, в ответ вернем два отрезка с заданным свойством - между левыми и правыми найденными концами. Весь алгоритм линейный.

№3

По аналогии получаем рекурентную формулу:

$$T(n) \leqslant T(\frac{n}{7}) + T(\frac{5n}{7}) + C'n$$

Покажем по индукции, что асимптотика останется линейной. Пусть $T(n) \leqslant Cn$:

$$T(n) \leqslant \frac{C}{7}n + \frac{5C}{7}n + C'n = n(\frac{6}{7}C + C') \leqslant Cn$$

C' - какая то константа алгоритма, для нее всегда можно подобрать $C \geqslant 7C'$, чтобы выполнялся последний знак неравенства. Т.е. алгоритм линейный.

№4

Сделаем устойчивую сортировку. Для этого можно воспользоваться вторым массивом, в начало которого будем добавлять нули, проходясь по данному массиву в первый раз, в порядке их следования, и то же самое для единиц при втором проходе. В итоге, получим в нашем массиве слева нули в том же порядке, за ними 1 в том же порядке.

№5

Воспользуемся расширенным алгоритмом Евклида. Чтобы получить искомый x надо решить $ax + My = -b$. Расширенный алгоритм Евклида отстанавливается при нахождении НОД-а чисел a и M, а значит, шагов в нем 2n (это длина двоичной записи обоих чисел a и M). На каждом шаге производится деление с остатком и умножение, требующие n^2 операций. В итоге, получаем алгоритм, с кубической асимптотикой. Его корректность диктуется решением диофантовых уравнений.

№6

а)В худшем случае, когда массив уже упорядочен, рекурентная формула примет вид: $T(n) = T(n-1) + \Theta(n)$ и делений массива будет болше всего, т.к. делим до тех пор, пока полученный (кусок) массив не будет содержать 1 элемент. При этом вызовов рекурсии будет n штук.
б)Для того, чтобы уменьшить число рекурентных вызовов, надо достичь T(1) - массива из 1 элемента как можно быстрее. Этого можно достигнуть, если делить исходный массив пополам (получим $\log_2 n$ вызовов рекурсии). Для этого необходимо в поданном на сортировку массиве найти медиану, которая и поделит массив примерно пополам в результате partition - а.

№6(Семинар)

Не будем делить массив пополам, а сразу начнем сливать элементы в пары, пары в четверки и так далее. Предположим для простоты, что число элементов в массиве - n - это степень двойки.
Псевдокод:

```
 for(shag = 1; shag <= n/2; shag *=2){
 beg1=1
 end1=shag
 beg2=beg1+shag
 end2=2*shag
 do{
buffer = merge([beg1,end1], [beg2,end2])
[beg1, end2] = buffer
beg1 += 2*shag
end1 += 2*shag
beg2 += 2*shag
end2 += 2*shag
 }
 while(end2 <= n)
 }
```

Используем здесь массив buffer длины n для временного хранения слитых частей. Длины частей (shag) меняются как 1 2 4 8... . Операция merge производит слияние двух массивов, переданных как части исходного массива по их началам и концам beg и end. Результат слияния записывается на место обоих кусков. Т.к. алгоритм проводит те же операции, что и рекурсивный вариант, его асимптотика не изменится.