Алгоритмы. ДЗ на неделю 6.

Прохоров Юрий, 771

Задача 6 (ДЗ 5)

Алгоритм и корректность

Обозначим подаваемый на вход многочлен как $f(x)$. Все коэффициенты в нем натуральные, поэтому функция $f : \mathbb{N} \longrightarrow \mathbb{N}$ является строго возрастающей. Тогда для нахождения корня можно воспользоваться бинарным поиском.

Заметим, что $\forall x \geq \lceil \log_n y \rceil \longmapsto f(x) > a_n y \geq y$, поэтому искать корень в этом диапазоне нет смысла. Корень искать будем на отрезке $[1, 2, \ldots, \lfloor \log_n y \rfloor]$. На каждой итерации поиска придется вычислять значение многочлена в выбранной точке. Для этого будет пользоваться стандартным алгоритмом возведения в степень, потому что нам нужно вычислить каждую степень числа x.

Таким образом, если мы при поиске оказалось, что какое-то значение многочлена совпало с y, то аргумент и есть его корень. Если бинарный поиск результатов не дал, то натуральных корней у такого уравнения нет.

Оценка по времени

Корень ищется в массиве длиной $\Theta(\log y)$, на каждом шаге поиска производится $n - 1$ умножений для возведения в степень, n умножений на коэффициенты перед степенями и n сложений. Считая арифметические операции за $O(1)$, общая сложность алгоритма есть $O(\log \log y \cdot n)$

Задача 1 (ДЗ)

```
Function push(Stack, x) :
    Enqueue(Q1, x);
    while Q2 not empty do
        enqueue(Q1, dequeue(Q2));
    end
    t = Q1;
    Q1 = Q2;
    Q2 = t;
end
```

```
Function pop(Stack) :
    dequeue(Q2);
end
```

Задача 2 (ДЗ)

Будем считать, что дерево хранится в массиве с номерами $[0, 1, \ldots, n]$. Корень дерева будем считать нулевым уровнем. Номер первого узла на k-м уровне есть $3^0 + 3^1 + \ldots + 3^{k-1} = \frac{3^k-1}{2}$. Рассмотрим i-ый по порядку элемент на этом уровне (считаем крайний левый нулевым), т.е. элемент с номером $\frac{3^k-1}{2} + i$.

Найдем номер его детей на следующем уровне.

Перед рассматриваемым элементом стоит i других узлов, у каждого из которых по 3 детей. Уровень под номером $k+1$ начинается с номера $\frac{3\cdot 3^k-1}{2}$. Сначала на этом уровне находятся $3i$ детей других узлов, поэтому "нужные"дети начинаются на этом уровне с номера $\frac{3\cdot 3^k-1}{2}+3i$.

Таким образом, если номер узла $n=\frac{3^k-1}{2}+i$, то номера его детей начинаются с $\frac{3\cdot 3^k-1}{2}+3i=\frac{3\cdot 3^k-3}{2}+3i+1=3n+1$. Поэтому детьми узла $[n]$ являются элементы $[3n+1],[3n+2],[3n+3]$.

Найти родителя по номеру ребенка можно по формуле $\left\lfloor\frac{n-1}{3}\right\rfloor$.

Задача 3 (ДЗ)

Будем предполагать, что все ключи элементов различны, иначе в случае $y=x$, y вообще может быть потомком x, что не соответствует условию.

Найдем, где относительно элемента x находится определенный в условии элемент y. Он не может быть потомком x, так как $y>x$. Пусть y не является предком x. Тогда существует узел z (это может быть и корень), такой что z — самый нижний общий предок элементов x и y. Вершины x и y лежат в разных поддеревьях z, иначе бы существовал их общий предок, лежащий ниже z. x лежит в левом поддереве z, а y — в правом поддереве, потому что $x<y$ (иначе — противоречие условию). Но тогда $x<z<y$, и y не является последующим за x — тоже противоречие.

Таким образом, y является предком x, причем x лежит в левом поддереве y. Допустим, левым потомком y является элемент w, чей левый дочерний узел v также является предком x или самим x. Но тогда $x<w<y$, и мы опять приходим к противоречию. Тогда y обладает всеми описанными в условии задачи свойствами.

Задача 4 (ДЗ)

Чтобы введенное в условии определение последующей и предшествующей вершин было корректно (это вершина определялась однозначно), необходимо, чтобы все ключи узлов в бинарном дереве поиска были различны.

Пусть bl и br — дети узла b. Будем для краткости обозначать ключ вершины b просто за b. Тогда $bl<b<br$.

Пусть c — последующая за b вершина, тогда она лежит в правом поддереве от b. Пусть у нее есть левый дочерний узел cl. Но тогда cl тоже лежит в правом поддереве b, и $b<cl<c$, и это противоречит выбору элемента c.

Доказательство для предшествующей вершины абсолютно аналогично.

Задача 5 (ДЗ)

1. $s(n,k,1)=n$.
Пусть в таблице хватает $n-1$ строк. Тогда по принципу Дирихле для каких-то двух элементов множества запросы будут одинаковые, то есть для них будет одинаковый ответ. Но один из них может лежать в подмножестве A, а другой — не лежать. Поэтому меньше n строк быть не может, если $0<k<n$ (иначе строк вообще не нужно).

Задача 6 (ДЗ)

Пусть $(k_1, k_2, \ldots, k_n)$ — искомая перестановка клиентов. Обозначим $p_i = t_{k_i}$, т.е. времена обслуживания клиентов в порядке приема $(p_1, p_2, \ldots, p_n)$. Суммарное время ожидания в таком случае

$$T = (p_1) + (p_1 + p_2) + \ldots + (p_1 + p_2 + \ldots + p_n) = np_1 + (n-1)p_2 + (n-2)p_3 + \ldots + p_n$$

Покажем, что T минимально, когда

$$p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_n.$$

Пусть имеется произвольный порядок приема клиентов. Допустим, мы поменяли местами клиентов с временами p_i и p_j $(i < j, p_i < p_j)$, то есть их времена теперь нарушают порядок возрастания, хотя до этого не нарушали. Новое время

$$T' = np_1 + \ldots + (n-i+1)p_j + \ldots + (n-j+1)p_i + \ldots + p_n,$$

$$T' - T = (n-i+1)p_j + (n-j+1)p_i - (n-i+1)p_i - (n-j+1)p_j = -ip_j - jp_i + ip_i + jp_j = (p_j - p_i)(j - i) > 0$$

Таким образом, если в некоторой последовательности приема клиентов какая-то пара времен ожидания нарушает порядок возрастания, то, поменяв их местами, мы уменьшим суммарное время ожидания. Проводя эту операцию, пока это возможно, мы отсортируем массив по возрастанию.

В общем случае необходимо найти отсортированную последовательность времен ожидания клиентов, поэтому для этого потребуется, например, применения алгоритма *QuickSort* и время $O(n \log n)$.