Домашнее задание №12

Томинин Ярослав, 778 группа

6 мая 2018 г.

1.

Назовем эту последовательность A. Последовательности будем выводить в лексикографическом порядке: изначально стоит 1,1,1,...,1; в конце n,n,...,n. Далее(для того, чтобы составить следующую последовательность) будем искать справа число i: A[i]<A[i+1]>...>A[n]. Далее аналогично изученному алгоритму поменяем A[i] на с A[i]+1, а все элементы справа сделаем равными 0. Повторим этот алгоритм до тех пор, пока наша последовательность не станет из n.

Корректность: Наш алгоритм рассматривает всевозможные перестановки. Для доказательства мы можем провести аналогию с числами, основание которых равно n+1. Тогда наш алгоритм будет перебтрать подряд все числа от 1111..1 до nnn...nnn. Поэтому это и будут всевозможные последовательности длины к.

Сложность по времени: На каждой итеррации алгоритм тратит O(k), а всего итерраций n^k поэтому обзая сложность $O(n^k k)$

2.

Алгоритм: Научимся находить стоимость разреза строки U[i,j](строка, состоящая из $u_i, ..., u_j$). Сделаем это с помощью рекуррентного соотношения: Стоимостью разреза U[i,j] (i<j) назовем $A[i, j] = min(A[i, k] + A[k, j] + \sum_{q=i}^{j} u_q)$ для всех i<к<j . Учтем, что стоимость еденичного разреза равна 0.(U[i,i+1]) Создадим таблицу, в которой в клетке i,j будет стоять A[i,j]. Заполнять таблицу будем справа налево и так мы дойдем до U[1,n].

Корректность: В нашем алгоритме мы рекурсивно находим U[i,j]. Рекурсивная формула разбирает все возможные случаи, поэтому мы можем доказать по индукции, что U[1,n] будет найден корректно.(База: все A[i,i+1]=0)

Осталось понять, что высчитывая A[i,j] у алгориьма будут высчитаны все нужные разрезы. Для этого достаточно понять, что алгоритм использует только правые и верхние элементы. Докажем что он не использует левые и нижние: если бы это было правдой, то для нахождения A[i,j] нам бы потребовалось A[i+q,j+w], где одно из q,w больше 0. Но тогда это означает, что наш разрез содержит элементы, которые не входят в него. Противоречие.

Сложность по времени: На каждой итеррации наш алгоритм находит min из O(n) элементов. Всего итерраций $O(n^2)$. Из этого следует, что сложность алгоритма $O(n^3)$.

3.

Алгоритм: Применим сортировку и посмотрим на ненулевую кучу, которая содержит нечетное количество монет и при этом содержит монеты наибольшего достоинства. Если такая куча найдется, то Алиса выйграла: она возьмет монету из этой кучи и во всех кучах останется четное количество монет, а потом будет повторять действия за Бобом. У нее всегда будет возможность сходить, так как количество монет в куче четно. Поэтому у Алисы всегда будет ход, а в силу того, что количество монет конеччно, то проиграет Боб. Если же не нашлось нечетной кучки, то Алиса проиграет, так как Боб будет повторять за Алисой и аналогично предыдущему доказателству у него всегда будет ход.

Корректность: Только что мы доказали, что у Алисы всегда будет ход, если сначала есть хотя бы одна нечетная кучка и, если же таковой не будет, то мы доказали, что всегда будет ход у Боба. Следовательно, алгоритм работает корректно.

Сложность по времени: Сначала мы делаем сортировку подсчетом за O(n), а потом ищем нечетную кучку за O(n). Следовательно, общая сложность O(n).

2)

Этот же алгоритм можно перефразировать как жадный: Алиса всегда берет монету максимального достоинства из нечетной кучи(если таковая в начале есть, то у нее всегда будет ход), если же такой нет, то Боб придерживается аналогтчной строатегии и берет монету максимального достоинства из нечетной кучи(мы доказали выше, что у него всегла будет ход).

Сложность по времени: Каждый раз алгоритм вынужден находить максимальную нечетную кучу, после сортировки подсчетом,если есть хотя бы одна нечетноая куча, то после хода Боба Алиса всегда будет знать, что максимальная нечетная куча- та, из которой Боб взял монету. В случае, когда нет нечетной кучи Боб аналогично индентифицирует максимальную нечетную кучу. Поэтому сложность алгоритма O(n).

4.

Алгоритм: если мы знаем, что a[1]...a[k] заполняют отрезок от 1 до N, то если a[k+1]>N+1, то ответом будет N+1. Иначе мы сможем покрыть отрезок N'=N+a[k+1]. После этого сравним следующее число с N' и т.д. Если же в какой -то момент a[k+1]>N+1, то выведем N+1, иначе(если мы не нашли такой ситуации и числа закончились) мы выведем сумму всех наших чисел+1.

Корректность: Докажем по индукции, что если мы ни разу не встретили ситуации a[k+1]>N+1, то наши числа покрывают отрезок [1,N], где N-сумма первых i чисел, которые мы сравнивали.

База: Первое число 1(иначе просто выводим 1), оно удовлетворяет нашему условию.

Переход: если

$$N_k = \sum_{i=1}^{k} a[i]$$

то докажем, что при $a[k+1] \leq N+1$

$$N_{k+1} = \sum_{i=1}^{k+1} a[i]$$

Действительно, мы можем покрыть все числа от 1...N не используя a[k+1], а потом с использованием a[k+1] добавляь каждое число из отрезка 1..N(можем составить из предыдущих чисел) и получим

$$N_{k+1} = \sum_{i=1}^{k+1} a[i]$$

Переход доказан.
Теперь докажем, что если a[k+1]>N+1, то N+1 не представимо. Оно может быть представимо числами a[1]...a[k](так как остальные > N). Но сумма этих чисел равна N, следовательно, N+1 не представимо.
Сложность по времени: Алгоритм делает не более n сравнений и n операций сложения, поэтому сложность O(n).
5.
Рассмотрим многоугольник, который образован вершинами k....l(l>k). Он состоит из сторон нашего многоугольника и хорды kl(на самом деле их 2, но мы берем тот, который не содержит сторону 1,n). Обозначим за a(k,l)-стоимость разреза этого многоугольника. Тогда попытаемся выразить его через рекуррентное соотношение: для l=k+1 и l=k+2 : a(k,l)=0, для l>k+2: a(k,l)=min(a(k,i)+a(i,l)+ L(k,i)+L(i,l)) по всем i: k<i<l, L-длина хорды(если она принадлежит прямоугольнику, то длина равна 0). Тогда мы умеем находить a через предыдущие. Если мы нарисуем таблицу, по осям которой к, l, а в клетках значения a, то мы можем Ссылаясь на верхние клетки находить значения нижних.
Тогда мы можем найти значение a(1,n), которое будет находиться в таблице в правом нижнем углу.
Корректность: мы вывели рекуррентное сооэтношение, которое перебирает всевозможные варианты:для l=k+1 и l=k+2 : a(k,l)=0, для l>k+2: a(k,l)=min(a(k,i)+a(i,l)+ L(k,i)+L(i,l)) по всем i: k<i<l. Так же мы умеем находить нижние через верхние. Следовательно мы сможем найти значение a(1,n).

Сложность по времени: Алгоритм делает $O(n^2)$ итерраций, га каждой он находит минимум, не более, чем от n чисел, поэтому сложность алгоритма $O(n^3)$.