Домашнее Задание по алгоритмам №2

Павливский Сергей Алексеевич , 873

17.02.2019

1 Задание №1

1.1 Пункт а)

Т.к. $c > 0$, то $g(n) > 1$, т.е. g(n) ограничено снизу функцией 1·C1 , где C1 = 1 .

По формуле геометрической прогрессии g(n) = $\frac{1-c^{n+1}}{1-c}$. $1 - c^{n+1} < 1$; 1 - c - const; значит g(n) ограничена сверху функцией вида $\frac{1}{C2}$, где C2 - const .

Значит g(n) - θ(1) по определению .

1.2 Пункт б)

$c^n = 1$ при c=1 . Тогда суммируя n единиц получаем n , что равно θ(n) .

1.3 Пункт в)

g(n) ограничена снизу функцией c^n , т.к. $c > 1$.

По формуле геометрической прогрессии g(n) = $\frac{1-c^{n+1}}{1-c} = \frac{c^{n+1}-1}{c-1} < \frac{c^{n+1}}{c-1} = c^n \cdot \frac{c}{c-1}$; значит g(n) ограничена сверху функцией вида $c^n \cdot C2$, где C2 - const .

Значит g(n) - θ(c^n) по определению .

2 Задание №2

2.1 Пункт а)

Заводим переменные s и x. s изначально 0 . Считываем в переменную x число и потока ввода . На k-м шаге умножаем переменную s на k-1 , прибавляем считанный элемент и делим на k .

2.2 Пункт б)

Заводим переменные max, x и count . max изначально 0 , count изначально 0 . Считываем первый элемент в x и присваиваем max значение этого элемента , count увеличиваем на 1. Далее последовательно считываем элементы из потока ввода в x и после каждого считывания выполняем проверки :

- считанный элемент < max - тогда ничего не делаем;

- считанный элемент = max - тогда увеличиваем count на 1;

- считанный элемент > max - тогда max = считанный элемент ; count = 1 .

2.3 Пункт в)

Заводим переменные element, x, max и count . element изначально 0 , count изначально 0 , max изначально 0 . Считываем первый элемент и присваиваем element значение этого элемента , count увеличиваем на 1 . Далее последовательно считываем элементы и после каждого считывания выполняем проверки:

считанный элемент = element - тогда увеличиваем count на 1;

count > max - тогда max = count;

считанный элемент != element - тогда element = считанный элемент; count = 1.

3 Задание №3

Заведем переменные a, b и c . Сохраним в переменную a первый элемент массива A , т.е. A[0]; в переменную b элемент B[0]; в переменную c элемент C[0] . Т.к. массивы отсортированны по возрастанию , то в переменных хранятся минимальные элменты 3-х массивов . На каждом шаге будем выбирать min(a, b, c) , увеличивать на 1 счетчик того массива , соответствующая которому переменная оказалась минимальной , и увеличивать счетчик числа различных элементов . Если переменных с наименьшим значением несколько , то выполняем то же самое , но увеличиваем счетчики тех массивов , переменные которых оказались минимальными . Далее переменной(-ым) , у которой(-ых) оказалось(-ись) минимальные значения присваиваем новое значение того элемента массива , на который теперь указывает счетчик . Если последний элемент некоторого массива оказался на некоторой итерации минимальным , а остальные массивы еще не на последнем элементе , то после выполнения указанных выше процедур этот массив перестает участвовать в сравнении .

Сложность по времени :

Т.к. в худшем случае после каждого сравнения счетчик только одного элемента будет увеличиваться , то суммарно будет произведено количество операций , равное сумме длин массивов , а это время линейное .

Доказательство корректности :

Этим алгоритмом мы гарантированно учтем все различные элементы , т.к. убираемый элемент различен со всеми остальными элементами в совем массиве по условию , а так же и с элементами других массивов , т.к. он <= рассматриваемой тройке элементов на данном шаге , а оставшиеся элементы в других массивах > соответствующих им элементов из рассматриваемой тройки .

Этим алгоритмом мы гарантированно обойдем все элементы , т.к. элемент убирается из рассмотрения только после его обработки .

В совокупности представляется очевидной корректность данного алгоритма.

4 Задание №4

Заведем 3 переменные : a1, a2, a3, a4, sum.

a1 - сумма всех считанных a, a2 - сумма всех считанных b, a3 - количество считанных чисел , sum - сумма всех чисел вида a i-e$\cdot$ b j-e, где i != j, a4 - последнее считанное a . Тогда сам алгоритм :

Считываем число , увеличиваем a3 на 1 .

Если a3 кратно 2 - значит мы считали некоторое a i-e , тогда a4 = считанное число; sum = sum + a2 $\cdot$ a4; a3 = a3 + 1;

Если a3 не кратно 2 - значит мы считали некоторое b i-e , тогда sum = sum + a1 $\cdot$ считанное число; a3 = a3 + 1; a2 = a2 + считанное число; a1 = a1 + a4.

В итоге ответом будет sum .

Оценка по времени :

каждое a будет перемножено со всеми b без одного; аналогично для b; тогда количество операций будет $O(n^2)$

Корректность :

Так как прибавляем последнее считанное a только после перемножения b су суммой , то произведений вида a i-e $\cdot$ b i-e не будет . Из структуры алгоритма понятно , что будут перебраны все произведения вида a i-e $\cdot$ b j-e , которые будут появлятся при умножении a i-го или b j-го на другую сумму . Так как на каждой итерации имеется промежуточный ответ , то это онлайн-алгоритм .

5 Задание №5

5.1 Пункт а)

Заведем массив B , где B[i] - длина максимальной возрастающей последовательности , оканчивающейся на элементе , у которого индекс i . Тогда максимум среди B[i] - это требуемый ответ . Инициализируем B[i] по следующему принципу :

- B[i] = 1 , если последовательность состоит только из одного числа A[i];

- B[i] > 1 , если перед A[i] есть другие элементы подпоследовательности.

Это любой элемент A[j] , где j < i , A[j] < A[i] .

Тогда если на k-м шаге нам нужно посчитать B[k] , то это можно сделать как 1 + max B[j] , где j < k , A[j] < A[i] .

Асимптотика :

Исходя из алгоритма в худшем случае мы на каждом шаге сравниваем элемент массива B со всеми элементами массива B , у которых индексы меньше . Суммарно количество операций - арифметическая прогрессия $\sum_{i=0}^{n-1} i$, что равно $\frac{n-1}{2}$ · n , что является $\theta(n^2)$

Доказательство корректности :

На каждой итерации мы будем находить оптимальное решение из возможных для соответствующего элемента , а значит и решение для конечного элемента будет оптимальным . Решение являются корректным динамического программирования .