Одиннадцатая домашняя работа по алгоритмам

Ситдиков Руслан

30 апреля 2018 г.

Задача 1

Алгоритм:
Введем обозначения: S_i - сумма всех элементов от 1 до i-ого, S_{min} - минимальная сумма для обработанной последовательности, $indexMin$ - ее индекс, p, q - начало и конец искомой подпоследовательности для обработанной в данный момент последовательности, max - сумма эл-тов искомой подпоследоватеотности, для уже обработанной последовательности.
1) Инициализируем переменные: $p = 0,\ q = 0$, $max = -\infty,\ S_{min} = 0$, $indexMin = 0$.
2) Пошагово вводим элементы последовательности.
3) На каждому шагу:
Ввели a_i, вычислили S_i;
Если $S_i - S_{min} > max \ \Rightarrow \ max = S_i - S_m in$ и $p = indexMin + 1,\ q = i$.
$S_{min} = min(S_{min}, S_i)$, а также если надо переприсваиваем $indexMin$.
4) Выводим ответ: отрезок $p,\ q$.

Корректность:
Предположим, что для некоторой последовательности найдены $S_m in$ и max (т.е искомая подпоследовательность). Тогда, если новый элемент a_i образует искомую подпоследовательность (с суммой елементов большей max) для нового мн-ва, то сумма элементов этой новой подпоследоваетльности равна $S_i - S_{min} > max$ (из определения S_i), если же новая сумма оказалась меньше $max \Rightarrow a_i$, не образует новой искомой последовательности, т.е искомая подпоследовательность осталась без изменений. Далее поддержав инвариантами $S_m in$ и max индукцией несложно убедиться в корректности данного алгоритма.

Ассимптотика:
Очевидно онлайн алгоритм - $O(n)$

Задача 2

Обозначим через $f(p, q)$ максимальную длину общей подпоследовательности последовательностей $x[1] \ldots x[p]$ и $y[1] \ldots y[q]$. Тогда

$$x[p] \neq x[q] \Rightarrow f(p, q) = max(f(p, q-1), f(p-1, q));$$

$$x[p] = x[q] \Rightarrow f(p, q) = max(f(p, q-1), f(p-1, q), f(p-1, q-1) + 1);$$

Поскольку $f(p-1, q-1) + 1 \geq f(p, q-1), f(p-1, q)$, во втором случае максимум трёх чисел можно заменить на третье из них. Поэтому можно заполнять таблицу значений функции f, имеющую размер $n \cdot k$. Можно обойтись и памятью порядка k (или n), если индуктивно (по p) вычислять $\langle f(p,0), \dots f(p,q)\rangle$ (как функция от p этот набор индуктивен).

Задача 3

Алгоритм:
1) Объявляем массив $isAchievable$ размера n, элемент которого $isAchievable[i]$ характеризует "достижимость"(возможность получить данную сумму из заданных купюр) суммы равной i. При этом присваиваем $isAchievable[0] = true$.
2) Организуем цикл по i от 0 до s. На каждом шаге работы цикла вычисляем $isAchievable[i]$ по правилу: присваиваем $isAchievable[i] = true$ только в том случае, если $\exists V_j : isAchievable[i - V_j] = true$.
3) По завершении цикла смотрим на значение $isAchievable[s]$. Если оно истино выводим "Достижима иначе "Недостижима".

Корректность: Предроложим что для всех $j < i$, $isAchievable[j]$, тогда i-ую сумму мы можем получить как добавление к какой-то меньшей сумме одного из номиналов, очевидно что если существует какая-то достижимая сумма $j < i$, и какой-то номинал $V_k : \ V_k + j = i$, то сумма i также достижима, в противном случае не достижима. Далее по индукцией получаем корректность алгоритма.

Ассимптотика: Всего рассмотрим s сумм, для каждой суммы будут проверены все суммы связанные с ней с помощью всех имеющихся номина лов, т.е n сумм. Получаем оценку - $O(ns)$.

Задача 4

Алгоритм:
1) Запускаем поиск в ширину от любой вершины, находим листья и от них запускаем дальнейший алгоритм.
2)Для каждого поддерева ищем два покрытия: P - минимальное покрытие вообще, и Q - минимальное покрытие, содержащее корень (для листьев $P = 0, Q = 1$). Пусть T - наше дерево, X - его поддеревья, растущие из детей корня. Тогда:

$$Q(T) = 1 + sum(P(X))$$

(корень включён, все рёбра, идущие к нему, учтены, в поддеревьях можно выбирать любые покрытия)

$$P(T) = min(Q(T), sum(Q(X)))$$

(либо корень не включён - и надо брать покрытия поддеревьев, содержащие корни, либо взять уже построенное покрытие Q).
3) Завершаем алгоритм, когда дойдем до корня, выводим $P(root)$.

Корректность:
Предположим, что для поддеревьев X, дерева T верно вычислены $Q(X)$ и $P(X)$, тогда очевидным образом $Q(T) = 1 + sum(P(X))$. Если же корень не был обязательно взят в искомое мн-во, то возможны два варианта - взять его или нет, если мы его не взяли, то из условия покрытия мы должны взять все корни его поддеревьев, иначе делать мы этого не обязаны - т.е достаточно взять все $P(X)$, далее, находя минимум получаем наш инвариант. Из индукции с вышеописанным переходом следует корректность.

Ассимптотика:
Формирование дерева и подъем от дерева к корню очевидно $O(V)$.

Задача 5

1) Заметим, что для того чтобы найти минимальное расстояния редактирования нам необходимо вычислить только последнюю строчку таблицы $m*n$ (а именно элемент mn). Также заметим, что для вычисления новой строки нашей таблицы необходимо знать только предыдущую строку и первый элемент строки (он очевидно равен номеру строки -1). Поэтоу из этих предположений очевидным образом вытекает алгоритм:
1) Очевидным образом заполняем первую строку таблицы.(0, 1, 2, ...)
2) Новую строку вычисляем из предыдущей по стандартному правилу, аналогичному в неоптимизированном алгоритме (Первый элемент очевиден).
3) Затираем старую строку нововычесленной. И повторяем операцию пока не вычислим m-ую строку.
Таким образом, данному алгоритму необходимо $2n + 1$ ячейка памяти - $O(n)$.
2) Аналогично пункту выше доходим до строки под номером $m/2$ далее создаем массивы предков $p[n]$ и $pNew[n]$. Инициализируем $p[n]$ строкой $1, 2, 3...$, в $pNew$ при вычислении значений новой строки записываем элемент $p[i]$, если рассматриваемый элемент данной формирующейся строки

наследуется от элемента предыдущей строки с индексом i. После вычисления очередной строки записываем в p значения из $pNew$. Процесс останавливается, когда достигается строка m, несложно показать индукцией, что в $p[n]$ лежит искомое k.

Задача 6