Разбор задач лабораторной работы А. А. Шалыто

Якутов Дмитрий

14 января 2017 г.

1 Задача A. Мутация

Вероятность того, что строка A перейдет в строку B при мутации — это произведение вероятностей по всем позициям i от 1 до n, где n — длина строк, того, что i-й символ строки A перейдет в i-й символ строки B. Если символы различаются, то эта вероятность равна $\frac{1}{n}$, иначе она равна $1 - \frac{1}{n}$.

2 Задача B. Кроссовер

Во-первых, проксорив все строки из множества со строкой-образцом, мы сведем задачу к следующей: из этого множества нужно выбрать две строки, из которых с помощью соответствующего типа кроссовера можно получить строку из нулей.

Сделать одноточечный кроссовер можно так: взять префикс одной строки, суффикс другой, сконкатенировать их и получить строку из нулей. Понятно, что выгоднее всего брать в качестве первой строки строку с самым длинным префиксом из нулей, а в качестве второй — с самым длинным суффиксом из нулей. Если сумма длин этих префикса и суффикса хотя бы n, то ответ для одноточечного кроссовера — "YES", иначе "NO".

Двухточечный кроссовер делается так: мы берем подстроку одной строки и вставляем её в соответствующее место другой строки. Если мы зафиксируем строку t, в которой будем делать замену, то понятно, что выгоднее всего взять у нее наибольший суффикс из нулей и наибольший префикс из нулей, а все, что между ними постараться чем-то заменить. То, что между ними, задается позицией начала этой подстроки и длиной подстроки. Давайте для каждой строки s из множества и для каждой позиции i преподсчитаем величину $a[s][i]$ — наибольшую длину подстроки этой строки, целиком состоящей из нулей, начинающейся в позиции i. Это делается за линейное время с помощью двух указателей или ещё как-нибудь. А теперь для каждого i посчитаем величину $b[i] = \max\limits_{s}(a[s][i])$. Теперь с помощью преподсчитанного массива b легко понять, можно ли использовать строку t в качестве той, где мы делаем замену.

Однородный кроссовер по-нормальному делается как-то сложно, я не знаю, как. Я сделал хитро, еле уложившись в ограничения. Для начала заметим, что для того, чтобы из двух строк можно было сделать строку из нулей с помощью однородного кроссовера, необходимо и достаточно, чтобы *and* этих строк как битовых векторов образовывал нулевую строку. Давайте переберем пару строк, проэндим их, проверим, что получился нулевой вектор, если да, то ответ "YES". Это требует порядка $m^2n/2$ действий. Заметим, что если сделать битовое сжатие, например, в одной 64-битной переменной хранить сразу 64 бита, то нам понадобится порядка $m^2n/128$ операций. В задаче ограничения таковы: $1 \le m, n \le 10^5, 1 \le m \cdot n \le 10^6$, описанное решение работает, когда $n > 20$. При $n \le 20$ можно решить задачу за порядка $2^n \cdot n + m$ операций: посчитаем динамику на векторах из множества (их всего не более 2^n различных): $dp[i] = true$, если существует вектор в множестве, являющийся подмаской битового представления числа i, $dp[i] = false$ иначе. Такая динамика считается за $2^n \cdot n$: инициализируем ее векторами из множества, а потом для каждого i такого, что $dp[i] = true$ говорим, что $dp[i \text{ or } (1 << x)] = true$ для всех x от 1 до n. Потом по этой динамике легко восстановить ответ. Перебираем один из векторов, который мы возьмем, пусть он соответствует битовому представлению числа i, тогда нам нужно взять к нему второй вектор, у которого *and* с нашим даст 0. Для этого нам нужно взять любой вектор, являющийся подмаской числа $((1 << n) - 1)$ xor i — отрицания вектора i. Соответствующая ячейка dp скажет нам, есть ли такой вектор.

3 Задача C. Деревья решений

Давайте обходить дерево обходом в глубину, запоминая для всех переменных, что мы с про них знаем. Знать мы можем одно из трех: что она равна 1, что она равна 0, или ничего. Каждый раз, когда мы приходим во внутренную вершину, на которой написана переменная, про которую мы уже что-то знаем, мы можем утверждать, что эта вершина бесполезная: ребро из её предка можно сразу провести в то её поддерево, которое соответствует значению этой переменной. Второе же поддерево можно удалить.

4 Задача D. Стартовое состояние

Напишем динамику. $dp[v][len] = true$, если из вершины v могла получиться последовательность выходных действий, равная суффиксу длины len подследовательности z. Ответом будут являться вершины автомата v такие, что $dp[v][m] = true$. Чтобы влезть в память нужно хранить не весь массив динамики, а только два столбца. Это возможно сделать, так как состояние динамики $dp[v][i]$ зависит только от состояний $dp[u][i-1]$ для некоторых u.

5 Задача E. Дискретные выходные воздействия

Давайте для каждого ребра (пары из состояния автомата v и входного события x) и каждого выходного воздействия y посчитаем, какой профит мы получим, если поставим на это ребро именно выходное воздействие y. Это делается втупую прогулкой по автомату для каждого из тестов. После этого для каждого ребра поставим на него то выходное воздействие, которое дает больше всего профита.

6 Задача F. Непрерывные выходные воздействия

В этой задаче выходное воздействие — действительное число, а функция приспособленности — некоторая квадратичная функция от $2 \cdot n$ аргументов, этих самых выходных воздействий. Эту функцию мы можем найти в явном виде прогулкой по автомату. Если мы зафиксируем все переменные, кроме одной, то мы получим либо параболу с ветвями, направленными вниз, либо константу. Второе может получиться, если ни одна из последовательностей входных событий не проходила через ребро, соответствующей переменной, которую мы не зафиксировали. Чтобы максимизировать значение функции, нужно, как минимум, максимизировать её значение при всех фиксированных переменных, кроме одной. А для этого нужно обнулить частную производную этой функции по этой переменной. Производная там — линейная функция от наших $2 \cdot n$ аргументов. По каждому из аргументов при фиксированных остальных наша функция либо имеет один локальный максимум, являющийся также и глобальным, либо является константой, так что обнулить все частные производные не только необходимо, но и достаточно. Это делается решением системы линейных уравнений $2n \times 2n$. Гаусс с выбором максимума в столбце заходит, наверное, метод простых итераций или ему подобные тоже зайдут.

Очень внимательно читайте формат ввода, сэмпл там очень слабый, ничего не покрывает.

7 Задача G. Умный муравей

В этой задаче нужно вырастить 10 автоматов. Некоторые из них можно сделать простым перебором всех автоматов, но пройти так все тесты достаточно проблематично. Почитать о выращивании автоматов можно, например, здесь: ссылка на сайт А.А. Шалыто. Там есть даже сама задача “Умный муравей”.

8 Задача H. Интерактивная минимизация

Простое, но важное наблюдение: задача независима по всем переменным, то есть если мы зафиксируем все переменные, кроме x_i, например, скажем, что они равны 0, и найдем a_i — минимум функции $g_i(y) = f(0, 0, \ldots, y, \ldots, 0)$, то ответом будет являться $f(a_1, a_2, \ldots, a_n)$.

Осталось научиться минимизировать функцию одной переменной вида $g(x) = q \cdot (x - a) + q_0 \cdot cos(2\pi t(x - a))$ за 1000 операций. Это можно сделать чем угодно, например так. Заметим, что период косинуса в функции не меньше $\frac{1}{10}$. Тогда, если мы разобъем область определения функции, то есть отрезок [0, 1], на отрезки длиной не более $\frac{1}{20}$, то на каждом из них функция будет выпукла. А тогда её минимум на каждом из отрезков можно найти тернарным поиском.

9 Задача I. Пять минимумов

Думаю, эта ссылка расскажет больше, чем могу рассказать я: differential evolution.