Задача A. Генеральная уборка

Нетрудно заметить, что достаточно убрать каждую точку в сетке размера $(N+1) \times (M+1)$. Заметим, что убирая одну такую точку, убираются только все соседние. Значит, если покрасить все точки в чёрный и белые цвета в виде шахматной раскраски, то будет достаточно убраться только в точках одного цвета. Поскольку $N+1 \geqslant 2$ и $M+1 \geqslant 2$, то и ответ будет равен $\lfloor \frac{(N+1)\times(M+1)}{2} \rfloor$. Доказательство оценки оставим в качестве упражнения для любопытного читателя.

Задача B. Или...?

Заметим, что если $A\ OR\ x = A$, то все единичные биты x являются единичными битами A. Это означает, что нужно посчитать количество битов, которые являются единичными для каждого из данных в вводе чисел, пусть оно равно k. Заметим, что тогда во всех решениях x_i все единичные биты будут являться подмножеством этих k битов. Значит, количество решений равно 2^k.

На самом деле, можно решить задачу чуть проще, для этого можно посчитать «побитовое И» у всех данных в вводе чисел и сосчитать количество единичных битов у полученного числа. Нетрудно догадаться, что это значение также будет равно k.

Задача C. Древняя игра

Вначале решим задачу, чтобы ни разу не проиграть. Очевидно, что если или $A_1 > A_2 + C_2$, или $B_1 > B_2 + A_2$, или $C_1 > C_2 + D_2$, то Валя проиграет, так как ему не хватит предметов, чтобы ни разу не проиграть либо камням, либо ножницам, либо бумагам соответственно.

Также можно показать, что если и $A_1 \leqslant A_2 + C_2$, и $B_1 \leqslant B_2 + A_2$, и $C_1 \leqslant C_2 + B_2$, то Валя при оптимальной стратегии не проиграет. Доказательство этого факта нетрудное и предоставляется интересующимся в качестве упражнения.

Теперь покажем, как Валя может разгромить: для достаточно победить бумагой камень ($A_1 - 1$ и $C_2 - 1$), ножницами бумагу ($C_1 - 1$ и $B_2 - 1$) или камнем ножницы ($B_1 - 1$ и $A_2 - 1$) а затем не проиграть на новых значениях.

Задача D. Конь в отпуске

Назовём два острова *соседними*, если можно как-либо перепрыгнуть с одного на другой. Назовём *границей острова* $x_{Li}, y_{Li}, x_{Ri}, y_{Ri}$ все клетки, принадлежащие этому острову с координатой левого нижнего угла $(x_{ei}; y_{ei})$ такой, что либо $x_{ei} = x_{Li}$, либо $y_{ei} = y_{Li}$, либо $x_{ei} = x_{Ri} - 1$, либо $y_{ei} = y_{Ri} - 1$. Заметим, что если конь может добраться до любой границы острова из любой другой клетки острова, кроме случая, когда он находится в центре доски размера 3×3. Также заметим, что если два острова соседние, то любой ход, в течение которого может произойти переход, будет с границы одного острова на границу другого. Эти два замечания означают, что если начальная клетка является центром острова размера 3×3, до добраться ни до какого другого острова нельзя.

Обозначим границы острова за *южную*, *северную*, *западную* и *восточную* границы, если клетки на ней имеют координату левого нижнего угла $x_{ei} = x_{Li}$, $x_{ei} = x_{Ri} - 1$ и $y_{ei} = y_{Ri} - 1$ соответственно. Заметим, что из южной границы одного острова, можно попасть только в северную другого (или угол, который одновременно является и северной, и восточной, либо и северной, и западной), из северной — только в южную, из западной — только в восточную, из восточной — только в западную.

Заведём для каждой возможной координаты по оси абсцисс и по оси ординат множество соответствующих границ: для оси абсцисс — западные и восточные, для оси ординат — южные и северные. Тогда, для каждого острова можно найти множество соседних, придя в южную границу и посмотрев в множество ординат на две клетки ниже, в северную — в множество ординат на две клетки выше, в западную — в множество абсцисс на две клетки левее, в восточную — в множество абсцисс на две клетки правее. Если хранить все границы в упорядоченных множествах, то можно найти всех соседей с соответствующей стороны за $O(K \cdot \log(N))$, где K — количество соседей с этой стороны.

Найдя для каждого острова соседей, можно построить неориентированный граф, где вершинами будут острова, а рёбрами — отношение соседства, в таком графе можно запустить поиск в глубину, чтобы посчитать количество островов, которые может посетить конь. Заметим, что вершин будет ровно N, а рёбер, на самом деле, $O(N)$ (доказательство этого факта оставим интересующимся читателям). Это означает, что для каждого острова найти всех соседей суммарно будет занимать $O(N \cdot \log(N))$ времени. В итоге, суммарная сложность решения будет равна $O(N \cdot \log(N))$.

Задача E. Хранение урана

Заметим, что если для каждого контейнера запомнить его индекс, отсортировать массив пар (v_i, i) — вместимость и индекс контейнера по возрастанию, то подходящий контейнер для минерала можно найти бинарным поиском пары $(a_j, 0)$, однако затем нужно удалить этот контейнер из массива. Для выполнения таких операций подойдёт упорядоченное множество.

Задача F. КНФ

Вспомним, что в каждой скобке находится ровно 500 попарно различных литералов. Оценим снизу, сколько символов займёт одна скобка. В этой скобке точно находится 500 символов `'x'`, 499 символов `'|'`, 1 символ `'('`, 1 символ `')'`, а также не меньше, чем $9 \times 1 + 90 \times 2 + 401 \times 3$ цифр (все числа от 1 до 500). Суммарно, одна скобка занимает не меньше, чем 2 390 символов. Значит, в строке длины 1 000 000 будет не больше, чем 499 различных скобок. Поскольку в каждой находится 500 различных литералов, то для каждой скобки можно выбрать уникальный литерал, то есть такой, чтобы их индексы не повторялись. Рассмотрим одну из скобок. Если литерал в нём без логического отрицания, то можно подставить соответствующей переменной истинное значение, если с логическим отрицанием — ложное значение.

На самом деле, есть решение, которое проще написать. Можно показать, что если подставить для каждой переменной случайное значение, то вероятность прохождения любого теста не меньше $1 - \frac{500}{2^{500}}$. Доказательство этого факта оставляется на упражнение интересующимся.

Задача G. Критичные вершины

Заметим, что вершина с номером 1 всегда является критической, назовём её корнем. Пусть мы удалили вершину с номером v, из которой выходили рёбра в вершины $u_1, u_2, \dots u_k$.

Заметим, что вершина v критическая тогда и только тогда, когда в какую-то вершину u_i входит ровно одно ребро (из вершины v). Докажем это. Из того, что в u_i входит ровно одно ребро, следует то, что при удалении v в неё не будет ничего входить, а, значит, она недостижима из корня.

Теперь покажем, что если v — критическая, то такая u_i существует. Вначале заметим, что в изначальном графе в любую вершину, кроме корня входит какое-то ребро. Пусть предположение не верно, удалим v, тогда в каждую вершину u_i входит какое-то другое ребро из вершины p_i. Это значит, что во все вершины всё также входит какое-то ребро. Пусть u_a недостижима из корня. Если p_a достижима из корня, то и u_a также достижима, противоречие, значит p_a также недостижима из корня. Найдём то ребро, которое ведёт в p_a, если эта вершина достижима из корня, то p_a и u_a должны быть достижимы, но это не так, значит эта вершина недостижима. Таким образом можно подниматься вверх. Если мы ни разу не попали в корень через N таких операций, то либо, мы остановились в какой-то вершине, что невозможно, так как в любую вершину кроме корня входит какое-то ребро, либо мы зациклились, что также невозможно, так как изначально дан граф без циклов и при удалении вершин и рёбер новых циклов появиться не может. Противоречие.

Таким образом, нужно посчитать количество вершин, у которых степень вхождения (количество входящих рёбер) какого-либо u_i равна единице. Это можно сделать за $O(N + M)$.

Задача H. Геометрический контест имени Вадима Баринова

Вначале решим задачу без запросов на отрез. Назовём многоугольник из входных данных A, его вершины $A_1, A_2, \dots, A_N$, а их радиус-вектора — $\vec{A_1}, \vec{A_2}, \dots, \vec{A_N}$. Поскольку многоугольник задан в порядке обхода против часовой стрелки, то его площадь равна $\frac{\vec{A_1} \times \vec{A_2} + \vec{A_2} \times \vec{A_3} + \dots + \vec{A_{N-1}} \times \vec{A_N} + \vec{A_N} \times \vec{A_1}}{2}$. Научимся быстро находить площадь многоугольника $A_L A_{L+1} \dots A_{R-1} A_R$, где $L < R$ (назовём такой многоугольник $A_{[L,R]}$). Для этого можно хранить префиксные суммы векторных произведений. Так $sum[1] = 0$, а $sum[i] = sum[i-1] + \frac{\vec{A_{i-1}} \times \vec{A_i}}{2}$. Тогда площадь многоугольника $A_{[L,R]}$ равна $sum[R] - sum[L] + \frac{\vec{A_R} \times \vec{A_L}}{2}$, что можно посчитать за $O(1)$ на запрос и $O(N)$ на предподпосчёт. На запрос замера тогда можно отвечать так: площадь всего многоугольника A равна площади многоугольника $A_{[1,N]}$, её можно быстро найти по описанному выше способу, тогда ответом на i-й запрос будет $max(S(A_{[min(l_i,r_i),max(l_i,r_i)]}), S(A_{[1,N]}) - S(A_{[min(l_i,r_i),max(l_i,r_i)]}))$.

Теперь добавим запросы на отрез. Пусть во всех запросах $L_j = min(l_j, r_j), R_j = max(l_j, r_j)$. Вначале сделаем простое замечание — в любой момент из многоугольника вырезано какое-то множество многоугольников $A_{[L_j,R_j]}$, а также, возможно, один какой-то многоугольник $A_{R_s}A_{R_s+1}\dots A_N A_1 \dots A_{L_s-1}A_{L_s}$ (назовём этот многоугольник *отрезающим стартовую вершину*). Это означает, что к вершинам с номерами $L_j + 1, L_j + 2, \dots, R_j - 2, R_j - 1$, а также $R_s + 1, R_s + 2, \dots, N, 1, \dots, L_s - 2, L_s - 1$ запросов поступать не будет.

Заведём ещё один массив префиксных сумм $cut[i]$, в котором будет храниться площадь, отрезанная от $A_{[1,i]}$, не учитывая отрезающий стартовую вершину многоугольник. Также, будем поддерживать площадь всего многоугольника S, для этого можно просто вычитать отрезанное из предыдущего значения площади. Заметим, что тогда при запросе можно считать только площадь многоугольника, в котором нет вершин с порядковым номером меньшим или большим, чем порядковые номера вершин из запроса (в то же время второй многоугольник будет являться *отрезающим стартовую вершину*), а для подсчёта площади второго многоугольника достаточно вычесть из площади всего многоугольника посчитанную площадь первого многоугольника. Тогда, храня префиксные суммы векторных произведений и префиксные суммы отрезанного от $A_{[1,i]}$, можно посчитать эту площадь по формуле $sum[R_j] - sum[L_j] + \frac{\vec{A_{R_j}} \times \vec{A_{L_j}}}{2} - (cut[R_j] - cut[L_j])$.

Научимся обновлять массив cut. Назовём площадь, посчитанную в формуле из предыдущего абзаца, S_0, тогда площадь второго многоугольника равна $S - S_0$. Пусть $S_0 < S - S_0$, тогда нужно отрезать этот многоугольник, после этого отрезания весь многоульник $A_{[L_j,R_j]}$ окажется отрезанным. Как было показано выше, к вершинам с номерами $L_j + 1, L_j + 2, \dots, R_j - 2, R_j - 1$ запросов больше поступать не будет, поэтому нам не важно, что лежит в них в массиве cut. Значит, нам достаточно прибавить к каждому значению cut на отрезке $[R_j, N]$ значение S_0 (причём нас не волнует, если мы обновим значение в какой-то уже удалённой вершине, потому что обращаться к ней больше не будут). Пусть теперь $S_0 \geqslant S - S_0$, тогда удаляется какой-то многоугольник, *отрезающий стартовую вершину*. Нетрудно заметить, что при его удалении массив cut обновлять не нужно, потому что он не учитывается в нём по определению.

Очевидно, что обновлять значения массива cut на отрезке поэлементно будет довольно долго, поэтому можно использовать дерево отрезков с операциями прибавления на отрезке и получения значения по номеру позиции. Тогда общая сложность работы решения будет равна $O(N+Q\cdot log(N))$, и занимать $O(N)$ памяти.

Задача I. Привлекательный плейлист

Заметим, что привлекательность трека — это длина наибольшего строго возрастающего подотрезка, назовём такой подотрезок *интересным*, содержащего этот трек. Нетрудно догадаться, что весь массив разбивается на наибольшие строго возрастающие подотрезки единственным способом. Пусть их длины равны $l_1, l_2, \dots, l_k$, тогда привлекательность плейлиста будет равна $l_1^2 + l_2^2 + \dots + l_k^2$.

Посмотрим, что происходит с привлекательностью треков при прибавлении на отрезке с l_j по r_j. Проще смотреть не на треки в отдельности, а на интересные подотрезки. Нетрудно заметить, что при таком изменении не больше одной пары соседних интересных подотрезков могли объединиться в один и не больше одного интересного подотрезка могло распасться на два, причём это могло произойти только с интересными подотрезками, содержащими l_j и/или r_j трек.

Посмотрим, что происходит с привлекательностью треков при присвоении на отрезке с l_j по r_j. Заметим, что с $l_j + 1$ по $r_j - 1$ привлекательность станет равна 1, а l_j-й и r_j-й треки также могут либо стать обособленными интересными подотрезками, как и все остальные на отрезке, либо объединиться с соседними.

Такие изменения можно подсчитывать в дереве отрезков, в каждой вершине которого будут хранится следующие значения:

- сумма привлекательностей треков на этом отрезке;
- значение, стоящее на самой левой позиции отрезка;
- длина наибольшего строго возрастающего подотрезка внутри этого отрезка, включающего самый левый элемент;

- значение, стоящее на самой правой позиции отрезка;
- длина наибольшего строго возрастающего подотрезка внутри этого отрезка, включающего самый правый элемент;
- длина всего отрезка.

Нетрудно понять, как пересчитывать каждую из величин при каждом из обновлений, осталось реализовать сливание двух соседних отрезков в один, и это также нетрудно расписать, что останется интересующимся упражнением.

Задача J. Оптимист и пессимист

Заметим, что поскольку каждая фигура — это увеличенный плюс или минус, то можно с помощью поиска в глубину или в ширину сосчитать количество клеток в каждой, пусть у текущей это количество равно s_i, если $\frac{s_i}{5}$ — это квадрат целого неотрицательного числа, то это плюс, если $\frac{s_i}{3}$ — это квадрат целого неотрицательного числа, то это минус.

Также есть другое решение. Для каждой фигуры можно сосчитать ширину w_i и высоту h_i — разницу между координатой x у самой правой и самой левой клетки и разницу между координатой y у самой нижней и самой верхней клетки. Это можно сделать также с помощью поиска в глубину или в ширину. Если $w_i = h_i$, то это плюс, иначе — минус.

Задача K. Восьмёрки и девятки

Заметим, что итоговое произведение будет равно $(10^N - 1) \times (10^M - 1) \times \frac{8}{9}$. Это означает, что можно поменять местами N и M так, чтобы $M \leqslant N$.

Также заметим, что произведение также будет равно $\underbrace{888\ldots88}_{M\text{ раз}}\underbrace{000\ldots00}_{N\text{ раз}} - \underbrace{888\ldots88}_{M\text{ раз}}$. Поскольку $M \leqslant N$, то можно легко заметить закономерность в этой разности (например, вычитая столбиком):

- Если $D < M$, то на D-м месте слева стоит 8;
- Если $D = M$, то на D-м месте слева стоит 7;
- Если $M < D \leqslant N$, то на D-м месте слева стоит 9;
- Если $N < D < N + M$, то на D-м месте слева стоит 1;
- Если $D = N + M$, то на D-м месте слева стоит 2.