1

Запишем в узлы два значения: сумму элементов $[L, R)$ и сумму значений $a_i * i$. Тогда ответом на запрос для узла будет второе значение минус первое, умноженное на $(L-1)$. Док-во: $a_L + 2a_{(L+1)} + ... + (R-L) * a_R = \sum\limits_{i=L}^{R-1} (i-(L-1)) * a_i = \sum\limits_{i=L}^{R-1} i * a_i - (L-1) * \sum\limits_{i=L}^{R-1} a_i$

2

Сожмем координаты прямоугольников таким образом, чтобы они были в диапазоне $[0, 2n)$ и $x_i < x_j \Rightarrow x'_i < x'_j$ (аналогично для y). Разберем решение для аналогичной задачи для отрезков на прямой. Левый конец отрезка будет соответсвовать 1 в массиве, правый -1. Тогда точка с максимальной префиксной суммой будет ответом на задачу. Построим ДО, узел будет содержать сумму на интервале, а также максимальный префикс на нем (вместе с точкой в которой достгается макисмальное значение). Для узла значения очевидно совпадают с единственным элементом в узле. Для родительских узлов сравним ответ в левом сыне и сумму ответа в правом с суммой на интервале правого и выберем наибольший вариант. Ответ для всего массива можно получить в корне дерева за $O(1)$, изменение одного листа занимает $O(\log n)$. Вернемся к прямоугольникам. Рассмотрим их координаты y в порядке возрастания. Если мы рассматриваем "нижнюю"границу прямоугольника, нужно добавить в ДО 1 в его левой координате и отнять 1 в правой. (наоборот для случая с верхней границей). Таким образом, наибольший ответ среди $2n$ рассмотренных прямых будет ответом для прямоугольник (его координату x мы получим из ДО, y координатой будет та, на которой мы находимся в порядке обхода). Мы рассмотрим $2n$ прямых, каждый раз делая изменения в ДО и получая ответ за $O(\log n)$. Итого: $O(n \log n)$.

3

Пусть k - количество цифр в x. Предложим жадный алгоритм: слева направо брать подстроку максимально возможной длины. Докажем, что полученный ответ будет оптимальным. Число длины $k-1$ всегда меньше x, число длины $k+1$ всегда больше, следовательно наш алгоритм всегда берет либо подстроку длины $k-1$, либо k (кроме случая, когда строка закончилась). Предположим существует разбиение на подстроки, более оптимальное, чем разделение жадным алгоритмом. Пусть $f(x)$ - длина первых x подстрок в оптимальном разбиении ($g(x)$ - соответсвенная длина для жадного алгоритма).
1) Если $f(x) = g(x) \Rightarrow f(x+1) \leq g(x+1)$ Очевидно, т.к из равенства $f(x)$ и $g(x)$ следует что начало $(x+1)$-ой строки в обоих разбиениях будет в

одной позиции в исходной строке $\Rightarrow$ из определения жадного алгоритма его строка не может быть меньше, чем строка в оптимальном разбиении.
2) Если $f(x) < g(x) \Rightarrow f(x+1) \leq g(x+1)$ Из утверждений выше, длина новой строки в оптимальном разбиении $\leq k$, а длина строки в жадном алгоритме $\geq (k-1)$. Следовательно разница между длинами строк $\leq 1 \Rightarrow$ т.к значения f и g целые числа, утверждение доказано.
Т.к $f(0) = g(0) = 0$ то по индукции из первых двух утверждений следует что, $\forall x : f(x) \leq g(x) \Rightarrow$ алгоритм оптимален.
Реализация: Построим ДО. В узле будут храниться количетсво подстрок и начало последней подстроки разбиения для этого ответа для подстрок $[i, R)$ $i : [L, \min(R-1, L+k)]$.
1)Изменение: изменим элемент в текущей строке. Дальше изменим соответсвующие узлы в дереве. Ответ в листе всегда 1. Далее для узла будем считать новые значения так: Предпосчитаем значения чисел в суффиксах длины $\leq k$ в левом сыне и префиксах $\leq k$ в правом сыне. Далее посчитаем новые значения для узла. Возьмем корректный ответ для того же i из левого сына, с помощью предподсчитанных значений на суффиксе левого и префиксе правого определим число d (сколько цифр из правого блока можно приписать к последнему блоку левого) и получим ответ для этого i равный: $t->l[i] + t->r[d]$, начало последнего блока будет совпадать с соответсвующим значением из правого сына. Значение d считаем за $O(1)$ т.к мы должны проверить только два варианта k и $k-1$. В случаях когда в сыновьях элементов $< k$ ответом для i не лежащих в левом сыне будет 1, началом последней подстроки будет соответсвенно само i). Предпосчет в узле $O(k)$. Считаем $O(k)$ значений, каждое за $O(1)$ $\Rightarrow O(k)$ на узел. Узлов – $O(\log n \Rightarrow O(\log n * k) = O(\log n * \log x)$.
2) Ответом в каждый момент времени будет значение при $i = 0$ из вершины дерева. которое мы можем получить за $O(1)$.

4

Пусть j наибольший индекс массива на котором значение s становится равным 0. Тогда ответом на задачу будет $\sum\limits_{i=j+1}^{n-1} a_i$. Построим ДО. В узле будем хранить два значения: сумму на отрезке и ответ на задачу для массива [L, R) (обозначим это значение как $f(t)$. В листах значения будут равны a_i и $\max(0, a_i)$, в остальных узлах
$f(t) = \max(f(t.r), f(t.l) + sum(t.r)$. Докажем: если j принадлежит правому сыну то ответ уже посчитан в нем, если j принадлежит левому, суффикс в родительском узле содержит суффикс левого сына и сумму правого. При изменении мы изменяем значение в листе и пересчитываем значения на пути до корня по формуле за $O(\log n)$. Ответ для заданного массива будет лежать в корне дерева, мы можем получить его за $O(1)$.