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

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

Построим ДО. Узел хранит два значения: сумма на интервале и искомую функцию для интервала. В листах значения равны a_i и $\max 0, a_i$. Для не листового узла функция для нового интервала будет равна $\max(f(t.l), sum(t.l) + f(t.r))$ Если оптимальное d лежит в левом сыне ответ уже посчитан в нем. Докажем для правого сына: пусть $g(d)$ ответ при заданном d. $g(d) = sum(t.l) + \min(d : [L, R] : \sum_{i=L}^{d} a_i) = sum(t.l) + f(t.r)$.
1) Изменение: поменяем значение в узле и пересчитаем значения на пути до корня $O(\log n)$.
2) Запрос ответа: ответ уже посчитан в корне дерева.