28. Дерево отрезков, групповые операции, двумерное дерево

28.1 Общая информация

Дерево отрезков (ДО) - структура данных, которая позволяет считать некоторую ассоциативную функцию f на полуинтервале упорядоченного множества A, при условии наличия нейтрального элемента e относительно этой функции. А так же позволяет изменять один элемент или целый отрезок.

Время запроса, изменения: $\mathcal{O}(\log n)$
Кол-во памяти: $\mathcal{O}(n)$
Построение за: $\mathcal{O}(n)$

28.2 Структура

ДО представляет из себя бинарное дерево, в листьях которого находятся элементы исходного множества, а в остальных вершинах результат f от левого и правого сына. Таким образом высота дерева составляет $\mathcal{O}(\log n)$.

28.3 Построение

Построение на основе массива: Пусть $a_i = f(a_{2i}, a_{2i+1})$, проще говоря - вершины $2i, 2i+1$ являются потомками i-ой вершины. Найдем наименьшую степень двойки len, не меньшую $|A|$. Зарезервируем массив a размера $2len$. Заполним $a[len; len+n)$ исходными данными, а $a[len+n; 2len)$ заполним e. Далее двигаемся и считаем f от $a[len-1]$ до $a[1]$. Асимптотика по времени построения и памяти выполнена. т.о. $a[1] = f(A)$ в силу ассоциативности f.

28.3 Запрос f на полуинтервале

28.3.1 Запрос сверху

В качестве параметров рекурсии rec возьмем pos - номер текущей вершины в a и l, r - запрашиваемый полуинтервал, а так же L и R. Пусть $L = 0, R = len$ для $pos = 1$, проще говоря - это границы, за которые отвечает текущая вершина. Границы обновляются с обновлением вершины.

- Если $[L; R)$ не пересекается с $[l; r)$ - возвращаем e.
- Если $[L; R)$ лежит внутри $[l; r)$ - возвращаем $a[pos]$.
- Иначе возвращаем $f(rec(2pos, l, r, L, (R+L)/2), rec(2pos+1, l, r), (R+L)/2, R)$.

На каждом уровне дерева рекурсия дошла (вернули значение из этой вершины) до не более, чем двух вершин, так как иначе среди этих вершин нашлись бы две соседние, что невозможно, так как мы бы вернули значение из общего предка намного раньше. Всего уровней $\mathcal{O}(\log n)$, а значит и операций было $\mathcal{O}(\log n)$.

28.3.2 Запрос снизу

Реализация этого запроса проще, но не применима с массовыми операциями. Пусть $res = e$. Повторять, пока границы не соприкоснутся:

1. Если l указывает на правого сына - $res = f(res, a[l]), l \mathrel{+}= 1$, аналогично с правым концом.
2. $l \mathrel{/}= 2, r \mathrel{/}= 2$.

28.4 Изменение элемента

Пусть значение $a[pos]$ нужно поменять на val, тогда: $a[pos] = val$, пока $pos \neq 0$: $pos /= 2, a[pos] = f(a[2pos], a[2pos + 1])$. Время работы $\mathcal{O}(\log n)$. По сути делаем то же, что и при построении.

28.5 Массовые операции на полуинтервале

28.5.1 Введем несколько понятий

- Введем функцию g по которой будет происходить обновление.
  1. g должна быть ассоциативна.
  2. g на f должна быть дистрибутивна: $g(x, f(y, z)) = f(g(x, y), g(x, z))$.
  3. e_g - нейтральный элемент относительно g.
- В каждой вершине i будем хранить несогласованность $d[i]$. При всем этом настоящее значение $f(a[i]) = g(a[i], d[i])$. Несогласованность в вершине - это то, что мы пока не успели применить на данном полуинтервале.
- Идея в том, что если мы хотим применить g ко всем потомкам $a[i]$, то мы запомним это в $d[i]$ и сделаем только по необходимости запроса. Важно, что теперь $res(a[i]) = g(a[i], d[i])$.
- Проталкивание несогласованности вершины pos - $d[2pos] = g(d[2pos], d[pos])$, $d[2pos + 1] = g(d[2pos + 1], d[pos]), d[pos] = e_g$. Сложность $\mathcal{O}(1)$.

28.5.2 Обновление

В качестве параметров рекурсии upd возьмем pos - номер текущей вершины в a, arg - элемент, по которому будем обновлять и l, r - запрашиваемый полуинтервал, а так же L и R - границы ответственности текущей вершины. Сложность не изменилась.

- Если $[L; R)$ не пересекается с $[l; r)$ - return.
- Если $[L; R)$ лежит внутри $[l; r)$ - $d[pos] = g(d[pos], arg)$. Запоминаем несогласованность.
- Иначе:
  1. Проталкиваем несогласованность.
  2. Запускаем upd от детей.
  3. Пересчитываем текущее значение: $a[pos] = f(g(a[2pos], b[2pos]), g(a[2pos+1], b[2pos+1]))$.

28.5.3 Запрос

Отличия от обновления (те же пункты):

- return e.
- return $g(a[pos], d[pos])$.
- Иначе:
  1. Проталкиваем несогласованность.
  2. Пересчитываем текущее значение.
  3. return $f(get(2pos), get(2pos + 1))$.

Сложность не изменилась.

28.6 n-мерное ДО

28.6.1 Ассимптотика

Пусть $p[i]$ - размерность дерева по i-ому измерению.

Время запроса, изменения: $\mathcal{O}(\prod_{i=0}^{n} \log p[i])$

Кол-во памяти: $\mathcal{O}(\prod_{i=0}^{n} p[i])$

Построение за: $\mathcal{O}(\prod_{i=0}^{n} p[i])$

28.6.2 Основная идея

n-мерное ДО - обычное ДО, в вершинах которого лежат $(n-1)$-мерные ДО. Идея в том, что для начала мы по первому измерению находим в основном дереве деревья, которые отвечают за следующие измерения, и.т.д. Реализация ничем не отличается от обычной, но нужно грамотно определить функции.

28.6.3 Обновление

Заметим, что функции мы применяем только к деревьям одинаковой глубины и одинакового количества элементов, тогда аналогичным для обычного ДО проходом (сверху, снизу или массовым) обновляем элементы.
Принцип объединения двух деревьев a и b в дерево c (функция $merge$):

1. Если только одно из деревьев является нейтральным элементом, то вернуть дерево.
2. Если размерность деревьев 0 - вернуть $f(a, b)$ или $g(a, b)$ (в зависимости от того, что мы сейчас делаем).
3. Иначе для каждой вершины $c[i] = merge(a[i], b[i])$.

28.6.3 Запрос

Пусть нужно посчитать f на гиперпрямоугольнике, описанным r. $r[i]$ - начало и конец прямоугольника по i-ой координате.
На i-ой глубине делаем запрос к деревьям диапазона $r[i]$ и считаем f по ответу.
Сама механика запроса аналогично обычному случаю.