Динамизация на структури от данни част 2
22.01.2021

Преговор:
Имаме проблем $Q : T_1 \times 2^{T_2} \to T_3$ и множество $A \subseteq T_2, A$ - крайно.
Имаме *статичен* ($s \equiv static$) алгоритъм, със следните характеристики:
1) $P_s(A)$ - индексиране за време $\mathscr{P}_s(|A|)$
2) $Q_s(x, A)$ - решава проблема Q за време $\mathscr{Q}_s(|A|)$
3) $\mathscr{S}_s(|A|)$ - необходима памет

Имаме допълнителна функция $\sqcup : T_3 \times T_3 \to T_3$, която се изчислява за константно време и има хубавото свойство, че е разложима:

$$Q(x, A \cup B) = Q(x, A) \sqcup Q(x, B), \text{ за } A \cap B \neq \emptyset.$$

При тези предположения, това което успяхме да направим е, че разделихме представянето на едно множество A от елементи в T_2 на блокове, като грубо казано следвахме степените на двойката. Това ни позволяваше да извършваме търсенето за логаритмично време използвайки комбинатор. Имаме логаритмичен брой множества, чието обединение дава представяното множество. Извършваме $\log n$ заявки $Q_S(x, M_i)$ и това води до време $O\left(\log n, Q_s(A)\right)$. От време на време се налага да строим нови множества, но големината на множествата, които се налага да строим е обратно пропорционална на това колко често се налага да ги строим. Поради тази причина видяхме, че *амортизирано* добавянето на елемент ще изисква $O\left(\frac{P_s\left(|A|\right)}{|A|}\right)$ време, а след това влизайки в дълбочина - успяхме да постигнем време в лошия случай.

Хубавото на добавянето на елемент, спрямо операцията $\sqcup$, е че това е монотонна операция - т.е. множествата само нарастват. Когато обаче искаме и да трием елементи - нещата вече се променят, защото това нарушава монотонността. Това налага въвеждането на още едно понятие, като условие за това кога може и да трием елементи от нашата структура от данни.

Интуицията е подобна на тази при хешовете или при динамичните масиви. Идеята е, че когато получим елемент, който трябва да бъде изтрит от нашето множество, ние не го изтриваме реално от множеството, а вместо това го маркираме като изтрит. Това маркиране обаче, трябва да бъде такова, което да не разваля времето за търсене. Когато стигнем до търсения елемент - връщаме $\{\text{true}, \text{false}\}$ според това дали е маркиран или не.

Дефиниция: (*Разложим за триене проблем*) Разложим за търсене проблем Q заедно със статична структура S се нарича разложим за триене, ако винаги когато S е построена оригинално за n елемента - елемент от S може да бъде изтрит за време $\mathscr{E}_s(n)$ без това да нарушава сложността на заявките за търсене или на други заявки за триене.

(Дефиницията звучи малко противоречиво, тъй като имаме статична структура, от която може да трием елементи, но това е така защото имаме уговорката, че вместо реално триене ще маркираме по подходящ начин.)

Твърдение: Ако имаме статично решение на разложим за триене проблем за търсене $Q : T_1 \times 2^{T_2} \to T_3$ с характеристики:

1) $\mathcal{Q}_s(n)$ - времето за заявки е ненамаляваща функция на n
2) $\frac{\mathcal{P}(n)}{n}$ - времето за построяване на статична структура е ненамаляваща функция на n (средно на един елемент)
3) $\mathcal{E}_s(n)$ - времето за изтриване от статична структура, която отигинално е построена за n елемента също е ненамаляваща функция
4) $\frac{\mathcal{S}_s(n)}{n}$ - средно паметта за елемент е ненамаляваща функция

то може да динамизираме проблема Q със следните сложности:

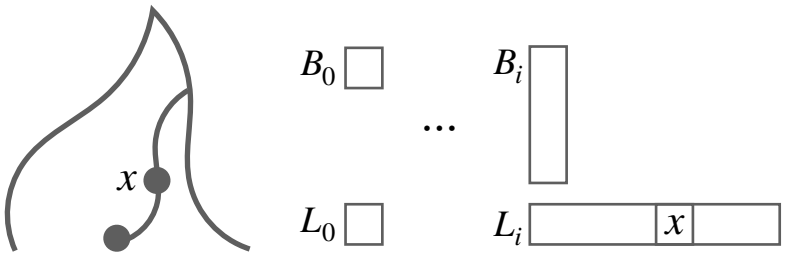
1) $Q_{\mathcal{D}}(n) \in O\left(Q_s(8n) \times \log n\right)$
2) $\mathcal{S}_{\mathcal{D}}(n) \in O\left(Q_s(8n)\right)$
3) $\underbrace{\overline{I_{\mathcal{D}}}(n)}_{\text{амортизирано добавяне на ел.}} \in O\left(\frac{P_s(8m)}{m} \times \log n\right)$
4) $\underbrace{\overline{\mathcal{E}_{\mathcal{D}}}(n)}_{\text{амортизирано изтриване на ел.}} \in O\left(\frac{P_s(4n)}{n} \times \log n + D_s(8n)\right)$

Доказателство:

Конструкция:

1) Имаме статични блокове $B_0, B_1, B_2, \ldots, B_k$, които представят $L_0, L_1, L_2, \ldots, L_k : L_i \cap L_j = \emptyset$ за $i \neq j$ и $\bigcup_{i=1}^{k} L_i = M$
2) B_i е създадено оригинално за $\leq 2^i$ елемента и $|L_i| \geq 2^{i-3}$, ако $L_i \neq \emptyset$
3) Балансирано дърво за търсене T за елементите, които реално са в M. За всеки елемент $x \in M$: $\begin{cases} ind(x) = i \Leftrightarrow x \in L_i \\ ptr(x) \text{ - указател към позицияна на елемента } x \text{ в } L_{ind(x)} \end{cases}$

T (търсим x по ключа му в T)



Предполагаме, че $k = 0$ или $L_k \neq \emptyset$. Тогава реалния брой елементи $|M| \geq |L_k| \overset{def.}{\geq} 2^{k-3} \Rightarrow 2^k \leq 8n$, т.е. $k \leq \log(8n)$.

В частност, всеки от блоковете B_i $(i \leq k)$ оригинално е бил построен за $\leq 2^i$ елемента, което е $\leq 8n$.

От тук може да изведем:

1) **Заявки**:

$Q_{\mathscr{D}}(x, \mathscr{D})$
    $r \leftarrow Q_s(x, \emptyset)$
    for $i = 0$ **to** k **do**
        if $L_i \neq \emptyset$ **then**
            $r \leftarrow r \sqcup Q_s(x, S_i)$
done
return r

Времева сложност:

$$O\left(Q_s(n)\right) \leq O\left(\sum_{i=0}^{k} Q_s(2^i)\right) \leq Q_s\left(8n \log(k+1)\right) \leq O\left(Q_s(8n) \times \log(8n+1)\right)$$

2) **Сложност по памет**:

$$\mathscr{S}_{\mathscr{D}} \in O\left(n + \sum_{i=0}^{k} \mathscr{S}_s\left(|B_i|\right)\right) = O\left(n + \sum_{\substack{i=0 \\ |B_i| \neq 0}}^{k} \frac{\mathscr{S}_s(B_i)}{2^i} \times 2^i\right) = O\left(n + \sum_{\substack{i=0 \\ |B_i| \neq 0}}^{k} \frac{\mathscr{S}_s(8n)}{8n} \times 2^i\right) =$$

$$= O\left(n + \frac{\mathscr{S}_s(8n)}{8n}\left(\underbrace{2^{k+1}}_{\leq 16n} - 1\right)\right) = O\left(n + \mathscr{S}_s(8n)\right) = O\left(\mathscr{S}_s(8n)\right).$$

3) **Добавяне** (амортизиран случай):

Разликата тук е, че не знаем дали броя на елементите може да стане точна степен на двойката (ще бъде между две поредни степени на двойката)

$Insert(x, \mathscr{D})$ // $x \notin \mathscr{D}$
    $L \leftarrow \{x\}$
    $j \leftarrow 0$
    while $|L| + |L_j| > 2^j$ **do** } $O(\log n)$
        $L \leftarrow L \cup L_j$
        discard B_j **and** L_j
        $j \leftarrow j + 1$

done

$L \leftarrow L \cup L_j$
$B_j \leftarrow P_s(L)$ } $2^{j-1} < |L| \leq 2^j$

$O(2^j)$ **for** $y \in L$ **do**
$ind(y) \leftarrow j$
$ptr(y) \leftarrow$ адрес на y в L
done

$L_j \leftarrow L$
$InsertKey(x, T)$ } $O(\log n)$

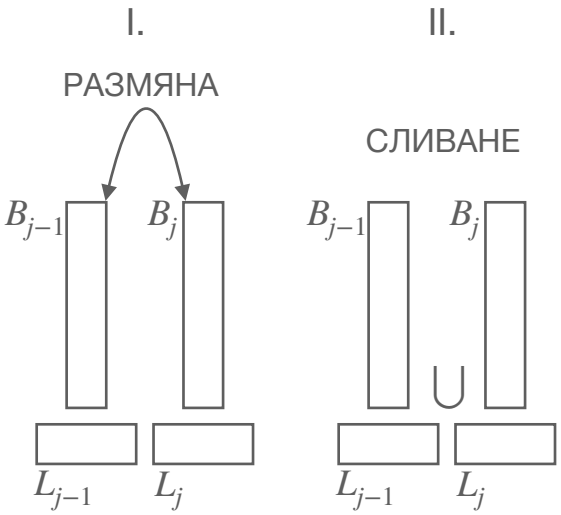
Наблюдение: $Insert$ създава структури B_j с $2^{j-1} < |L_j| \leq 2^j$ елемента.

4) **Триене** на елементи:

$Delete(x, \mathscr{D})$
$(j, ptr) \leftarrow Search(x, T)$ // $O(\log n)$
$L_j \leftarrow L_j \backslash \{x\}$ // $O(1)$ използваме указатели
if $|L_j| > 2^{j-3}$ **then**
$D_s(x, B_j)$ // $O\left(D_s(2^j)\right)$
else
if $|L_{j-1}| \geq 2^{j-2}$ **then**

Време $P_S(2^{j-3})$ $+2^j$ (доп.) РАЗМЯНА
$B_j \leftarrow B_{j-1}$
$B_{j-1} \leftarrow P_s(L_j)$
$swap(L_{j-1}, L_j)$
for $x \in L_{j-1}$ **do**
$ind(x) \leftarrow j-1$
done
for $x \in L_j$ **do**
$ind(x) \leftarrow j$
done

$2^{j-3} \leq |L_{j-1}| \leq 2^{j-2}$
$2^{j-2} \leq |L_j| \leq 2^{j-1}$



else // $|L_{j-1}| < 2^{j-2}$
$L \leftarrow L_{j-1} \cup L_j$ // $2^{j-3} \leq |L| \leq 2^{j-3} + 2^{j-2} < 2^{j-1}$
$B \leftarrow P_s(L)$
if $|L| < 2^{j-2}$ **then**
$L_{j-1} \leftarrow L$
$B_{j-1} \leftarrow B$
discard B_j **and** L_j
for $x \in L$ **do**
$ind(x) \leftarrow j-1$
$ptr(x) \leftarrow$ подходящо модифициран
done

else // $|L| \geq 2^{j-2}$

$2^{j-2} \leq |L_j| \leq 2^{j-1}$ {
- $L_j \leftarrow L$
- $B_j \leftarrow B$
- **discard** L_{j-1} **and** B_{j-1}
- **for** $x \in L$ **do**
  - $ind(x) \leftarrow j$
  - $ptr(x) \leftarrow$ подходящо модифициран
- **done**

Наблюдение: При изтриване с реорганизация, новите множества L_j са или празни или $2^{j-2} \leq |L_j| \leq 2^{j-1}$.

Анализ на $Delete$ (концентрираме се на един блок с фиксиран номер *j*):

1) Без реорганизация: $O\left(\log n + P_s(2^j)\right)$
2) С реорганизация: $O\left(\log n + P_s(2^{j-1}) + 2^j\right)$

Интересуват ни: броя стъпки от тип 1 броя стъпки от тип 2.
Общо време за триене от блок B_j:
#ст.1 $\times O\left(\log n + D_s(2^j)\right)$ + #ст.2 $\times O\left(\log n + P_s(2^{j-1}) + 2^j\right)$

Да разгледаме два момента от време $t_1 < t_2$, в които прилагаме стъпка 2 за изтриване на елемент от B_j. Без операциите изтриване от стъпка 1 - всички други операции осигуряват $|L_j| \geq 2^{j-2} \Rightarrow t_2 - t_1 \geq 2^{j-3}$ (т.к. 2^{j-3} стъпки от тип 1)
#ст.1 $\geq 2^{j-3}$#ст.2
#ст.1 + #ст.2 = #изтривания от $B_j \Rightarrow$ #ст.2 $\leq \dfrac{\text{\#изтривания}}{2^{j-3}}$, $\mathbf{m}$ = #**изтривания**

$$Time(\textbf{изтривания}) = O\left(\sum_{j=0}^{k} D_s(2^j)\#(\textbf{изтр. от } B_j) + \log n \times m + \sum_j \frac{P_s(2^{j-1}) + 2^j}{2^{j-3}} m\right) =$$

$$= O\left(D_s(8n)m + \log n \times m + 4\sum \frac{P_s(4n)}{4n} m\right) = O\left(D_s(8n)m + \log n \times m + \frac{P_s(4n)}{4n} \log n \times m\right)$$

за m изтривания.

Окончателно средно за изтриване харчим:

$$\overline{\mathscr{C}_{\mathscr{D}}}(n) = O\left(D_s(8n) + \log n + \frac{P_s(4n)}{n} \log n\right)$$

5) **Добавяне** на елементи (анализ):

Фиксираме едно j. Когато B_j е новопостроения блок в $Insert$, времето за цялата функция $Insert$ е $O\left(\log n + P_s\left(|L_j|\right)\right)$, като това което знаем е, че $L_0, L_1, \ldots, L_{j-1} = \emptyset$ и от друга страна $2^{j-1} < |L_j| < 2^j$.

Да разгледаме момента от време $t_1 < t_2$, в който се строи блок B_j. $\left(t_1, t_2 \text{ - броят операции от типа } Insert \text{ и } Delete \text{ (но не броят операциите за заявки)}\right)$. В момента t_2 в блоковете $L_0 \cup L_1 \cup \ldots \cup L_{j-1}$ има повече от 2^{j-1} елемента.

Те могат да се появят от:
1) реални добавяния - $\mathbf{I_R}$
2) от изтривания от стъпка 2, приложена към блока B_j - $\mathbf{I_E}$

Това което твърдим е, че $t_2 - t_1 \geq 2^{j-2}$, $t_2 - t_1 \geq I_R + \underbrace{I_E}_{\substack{\text{стъпки 2 от} \\ \text{триенето} \\ \leq \frac{\#\text{стъпки 1}}{2^{j-3}}}} 2^{j-3}$.

От друга страна, $2^{j-1} < I_R + I_E 2^{j-2}$

1) ако $I_E = 0$, тогава $I_R > 2^{j-1} \Rightarrow t_2 - t_1 \geq I_R > 2^{j-1}$
2) ако $I_E = 1$, тогава $2^{j-1} < I_R + 2^{j-2} \Rightarrow I_R > 2^{j-2} \Rightarrow t_2 - t_1 \geq 2^{j-2} + 2^{j-3} > 2^{j-2}$
3) ако $I_E \geq 2$, тогава $t_2 - t_1 \geq 2 \times 2^{j-3} = 2^{j-2}$

Следователно, ако m е броя на всички операции (триене и добавяне) - общото време за добавяне е: $O\left(\sum_{j=0}^{k} \frac{m}{2^{j-2}}\left(\log n + P_s\left(|L_j|\right)\right)\right) =$

$$= O\left(m \log n + 4 \sum_{j=0}^{k} m \frac{P_S\left(|L_j|\right)}{|L_j|}\right) = O\left(m \log n + 4 \sum_{j=0}^{k} m \frac{P_s(2^k)}{2^k}\right) =$$

$$= O\left(m \log n + 4m \frac{P_s(8n)}{8n} \log n\right).$$

Отново, след като разделим на общия брой операции, които сме извършили, т.е. m - получаваме амортизираната сложност за добавянията.

$\Rightarrow \overline{\mathcal{T}}_{\mathcal{D}}(n) = O\left(\log n + \frac{P_s(8n)}{n}\log n\right) = O\left(\frac{P_s(8n)}{n}\log n\right)$, като n е максималния брой елементи, които реално някога сме имали в нашата структура.