Fusion Trees

Marta Anna Ziobro

Wrocław, 24 stycznia 2011

1 Cel

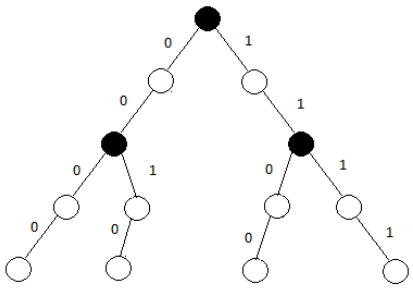
Wyobraźmy sobie świat, w którym liczby należą do dużego $Uniwersum$, a ich postać binarna ma długość ω. W świecie tym istnieje $Maszyna$, która potrafi wykonywać na tych liczbach podstawowe operacje matematyczne tj. $dojaj$, $odejmij$, $pomnoz$, XOR, $ZnajdzPierwszyNiezerowyBit$ itp w czasie stałym. My natomiast nie marzymy o niczym innym tylko o posiadaniu statycznej struktury, która zawiera zbiór Z n liczb z $Uniwersum$ i portafi odpowiadać na pytanie, co jest następnikiem a co poprzednikiem danej liczby q w Z. Wymagamy, żeby odpowiedź na takie pytanie była w $O(log_\omega(n))$, a struktura miała rozmiar $O(n)$.

2 Fusion Tree

Nasz problem rozwiązuje użycie struktury B-drzewa o liczności wierzchołków $\Theta(\omega^{1/5})$. Drzewo takie ma wysokość w $\Theta(log_{\omega^{1/5}}(n)) = \Theta(log_\omega(n)) = \Theta(log(\omega)/log(n))$. Musimy więc tylko zadbać o to, żeby algorytm przeszukiwania takiego drzewa odwiadzał każdy wierzchołek w czasie stałym. Ponieważ nie mamy czasu na przeczytanie wszystkich $\omega^{1/5}$ słów o długości ω, spróbujemy je skompresować do jednego słowa i przy sprawdzaniu, w którym poddrzewie wierzchołka należy szukać sąsiadów liczby k, skorzystać z mocy naszej $Maszyny$ i wykonać stałą liczbę operacji matematycznych.

2.1 Interesujące bity i zarys liczby

Wyobraźmy sobie binarne $Trie$ zbudowane dla reprezentacji binarnych kluczy w wierzchołku. Takie drzewo będzie miało co najwyżej $\omega^{1/5}$ liści i co najwyżej $\omega^{1/5} - 1$ wierzchołków dzielących. Zauważmy, że poziomy drzewa, na których występują wierzchołki dzielące utożsamiają się z pozycjami w reprezentacjach binarnych, na których któreś z kluczy się różnią. Będziemy nazywać te pozycje $b_0, \ldots, b_{r-1}$ (r to liczba wierzchołków dzielących), a bity interesującymi bitami.



Rysunek 1: $Trie$ dla kluczy 0000, 0010, 1100, 1111 z zaznaczonymi wierzchołkami dzielącymi. Interesującymi bitami są bit 0-owy i 2-gi

Definicja 2.1. $IdealnyZarys(x)$ to r-bitowy wektor, w którym na i-tej pozycji mamy b_i-ty bit x. Jest to podciąg bitów x zbudowany z interesujących bitów.

Dla kluczy z obrazka 2.1 $IdealneZarysy$ to kolejno 00, 01, 10, 11.

Zauważmy, że jeśli $x < y$ to $IdealnyZarys(x) < IdealnyZarys(y)$

Niestety $IdealnyZarys(x)$ trudno jest obliczyć, dlatego ograniczymy się do obliczenia $Zarys(x)$, który jest podciągiem bitów x zbudowanym z interesujących bitów z powstawianymi gdzie niegdzie dodatkowymi zerami.

Dążymy do tego aby móc zapisać $\omega^{1/5} - 1$ skonkatenowanych $Zarys$-ów w ω bitach. Dlatego każdy $Zarys$ nie powinien być dłuższy niż $\omega^{4/5}$.

Spróbujemy zatem przesunąć każdy interesujący bit x tak, ażeby zachował swoją kolejność w porządku bitów i żeby bit b_0 był odległy od bitu b_{r-1} o conajwyżej $\omega^{4/5}$ miejsc.

2.2 Obliczanie zarysu

Niech $x = \sum_{i=0}^{\omega-1} x_i 2^i$. Najpierw wyzerujemy wszystkie bity, które nie są interesujące:

$$xAND \sum_{i=0}^{r-1} 2^{b_i} = \sum_{i=0}^{r-1} x_{b_i} 2^{b_i} \tag{1}$$

Następnie przemnożymy wynik przez liczbę $m = \sum_{i=0}^{r-1} 2^{m_i}$ taką, że:

1. $\forall_{i,j} b_i + m_j$ są unikatowe.
2. $(b_{r-1} + m_{r-1}) - (b_0 + m_0) \leqslant \omega^{4/5}$
3. $b_0 + m_0 < b_1 + m_1 < \cdots < b_{r-2} + m_{r-2} < b_{r-1} + m_{r-1}$

$$\sum_{i=0}^{r-1} x_{b_i} 2^{b_i} \sum_{j=0}^{r-1} 2^{m_j} = \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} x_{b_i} 2^{b_i+m_j} \tag{2}$$

Po tej operacji każdy b_i-ty interesujący bit wpadnie w swoich prywatnych r miejsc. Nam potrzebne jest tylko jedno powtórzenie każdego bitu, ciesząc się zatem, że m spełnia warunki 2 i 3, wykonujemy operację:

$$\sum_{i=0}^{r-1} \sum_{j=0}^{r-1} x_{b_i} 2^{b_i+m_j} AND \sum_{i=0}^{r-1} 2^{b_i+m_i} = \sum_{i=0}^{r-1} x_{b_i} 2^{b_i+m_i}. \tag{3}$$

Biorąc pod uwagę bity od pierwszego do ostatniego niezerowego otrzymujemy słowo, którego długość jest mniejsza od $\omega^{4/5}$, a wszystkie interesujące bity, z których jest złorzone, są w odpowiedniej kolejności. Pozostałe bity są zerami. Podsumowując:

$$Zarys(x) = ((xAND \sum_{i=0}^{r-1} (2^{b_i}))m) AND \sum_{i=0}^{r-1} (2^{b_i+m_i}) >> \min_i (b_i + m_i). \tag{4}$$

A oto jak wyznaczamy m:

Krok 1: Poczynając od $i = 0$, aż do $r - 1$ wybieramy kolejno $m'_i < r^3$ takie, że $\forall_{0 \leqslant i,j<r} b_i + m_j$ są unikatowe modulo r^3.
Załóżmy, że wybraliśmy już spełniające warunki $m'_0 < m'_1 < \cdots < m'_{t-1}$. Zauważmy, że żeby wybrać m'_t musimy wystrzegać się liczb postaci $m_i + b_j + b_k \forall_{0 \leqslant i,j,k<r}$.
Możemy m'_t wybrać, ponieważ mamy r^3 liczb do wyboru, a tych, których musimy się wystrzegać jest najwyżej tr^2.
Krok 2: Powyższe wartości nie spełniają warunku zachowania kolejności. Poprawimy je zatem:

$$m_i = m'_i + ir^3 + ((\omega - b_i)\text{zaokrąglone w dół do wielokrotności } r^3\ ) \tag{5}$$

Teraz m spełnia warunki:

1. $\forall_i m_i \equiv_{r^3} m'_i$ więc $\forall_{0 \leqslant i,j<r} b_i + m_j$ są unikatowe modulo r^3, a zatem są kompletnie unikatowe.
2. $b_0 + m_0$ wynosi w przybliżeniu ω, a $b_{r-1} + m_{r-1}$ w przybliżeniu $\omega + r^4$, więc ich różnica jest $O(r^4) = O(\omega^{4/5})$
3. $\forall_{0 \leqslant i<r} b_i + m_i$ jest w przedziale rozmiaru r^3 za $(\lfloor \omega / r^3 \rfloor + i)r^3$, więc zachowujemy porządek

Zauważmy, że nadal jeśli $x < y$ to $Zarys(x) < Zarys(y)$

2.3 Zarys wierzchołka, a zapytanie

Definicja 2.2. $Zarys(w)$, gdzie w to wierzchołek $FusionTree$, jest konkatenacją $Zarys$-ów wszystkich kolejnych kluczy danego wierzchołka poprzedzielanych 1 tj.
$Zarys(w) = 1Zarys(x_0)1Zarys(x_1) \ldots 1Zarys(x_{k-1})$, gdzie k- liczba kluczy.

Ponieważ $k \leqslant \omega^{1/5}$, długość $Zarysu(x) \leqslant r^4$ zatem długość $Zarysu(w) \leqslant (r^4 + 1)k$. Ponieważ $r \leqslant k - 1$ to $r^4 + 1 < k^4$, zatem $Zarys(w) < k^5 \leqslant \omega$. Zatem zarys wierzchołka mieści się w jednym słowie.

Definicja 2.3. $PowielonyZarys(q)$, gdzie q to liczba z zapytania, to skonkatenowany k razy $Zarys(q)$ przedzielony 0-ami.

$PowielonyZarys(q)$ dla wierzchołka w ma długość równą $Zarys(w)$ i 0-ra przedzielające na tych samych miejscach, co 1-ki przedzielające.

Zauważmy, że:

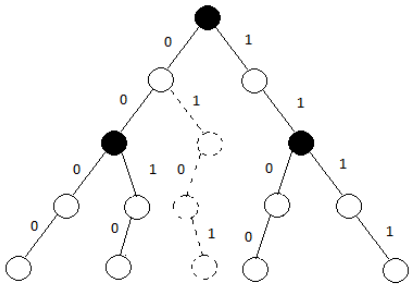
$$(Zarys(w) - PowielonyZarys(q))AND \sum_{i=0}^{k-1} 2^{i(r^4+1)+r^4} = \sum_{i=0}^{k-1} c_i 2^{i(r^4+1)+r^4} \tag{6}$$

gdzie

$$c_i = \begin{cases} 0, & Zarys(x_i) < Zarys(q) \\ 1, & Zarys(x_i) \geqslant Zarys(q) \end{cases} \tag{7}$$

Wystarczy więc odpowiedzieć na pytanie gdzie jest najbardziej znaczący bit (6), żeby znaleść $Zarys$ klucza, który jest poprzednikiem $Zarysu$ zapytania.

Niestety klucz, którego $Zarys$ jest poprzednikiem zarysu q, nie jest koniecznie poprzednikiem q. Rozważmy poprzedni przykład zestawu kluczy i zapytanie z $q = 0101$. $Zarys(q) = 00$, ale q wpada tak naprawdę do przedziału 0010 - 1100



Rysunek 2: $Trie$ dla kluczy 0000, 0010, 1100, 1111 i zapytania 0101

Ale zidentyfikowanie klucza 0000 jest po prawdzie bardzo przydatne. Możemy znaleść najdłuższy wspólny prefiks q i poprzednika/następnika jego $Zarys$-u. Nazwiemy go y (Możemy to zrobić czasie stałym dzięki operacji XOR). Po y nie może występować bit interesujący, bo jeśli by było inaczej, wtedy istniał by klucz, który ma dłuższy prefisk z q.
Następnie musimy się uporać z jedną z dwóch możliwych sytuacji: q należy do nieistniejącego poddrzewam za y, które jest lewe lub prawe. W naszym przykładzie należy do prawego. Poprzednikiem naszego q jest zatem największy element w lewym poddrzewie y. Aby go znaleść wyszukujemy $Zarys(e)$, gdze $e = y0111..11$ (w przypadku prawego poddrzewa uruchomilibyśmy zapytanie dla $e = y100..00$). Ponieważ e zaczyna się od $y0$, to zapytanie wpadnie w odpowiednie poddrzewo, a ponieważ zawiera poza tym same 1-ki, to jego $Zarys$ będzędzie większy lub równy od każdego $Zarysu$ w tym poddrzewie. A ponieważ porządek kluczy jest taki sam jak ich $Zarys$-ów, to zostanie znaleziony największy klucz naszego poddrzewa. Klucz ten będzie poprzednikiem q.