Celočíselné datové struktury ... vše na RAMu

Máme celočíselné universum $\mathcal{U} = \{0 \ldots U\}$, $w = \log U$

- obyčejné množiny → hešování (třeba kukačka) – dotaz $O(1)$ w.c., update $O(1)$ průměrně amort.
- množiny s předchůdcem (Pred) a následníkem (Succ) → tato přednáška

van Emde-Boasovy stromy [1975] – zde v pozdější reformulaci

- universum rozdělíme na $\sqrt{U}$ bloků velikosti $\sqrt{U}$
  - binární pohled: $\underbrace{\boxed{\text{blok}}}_{w/2 \text{ bitů}}\underbrace{\boxed{\text{pozice v bloku}}}_{w/2 \text{ bitů}}$ ... rozklad v $O(1)$ na RAMu

Df: vEB(U) si pamatuje pro množinu $S \subseteq \mathcal{U}$:

- min S (zvlášť!) ... $S' := S \setminus \{\min S\}$
- max S (kopie)

} pokud je strom prázdný, min = max = ∅

- přihrádky $P_0, \ldots, P_{\sqrt{U}-1}$ ... do nich roztřídíme S'
  - uvnitř P_i je vEB($\sqrt{U}$) s dolními polovinami prvků
- sumární strom: vEB($\sqrt{U}$) pro $\{i \mid P_i \neq \emptyset\}$

Hloubka vnoření je $O(\log \log U)$

Find(x): triviální – zanořuji se do přihrádek, stojí $O(\log\log U)$

Succ(x):
1. Rozdělíme x na (i, j).
2. Pokud $x < \min$, vrátíme min.
3. Pokud $x \geq \max$, vrátíme ∅.
4. Pokud ($P_i = \emptyset$) nebo $j \geq P_i.\max$: ← jelikož $x < \max$, i' určitě existuje
   $i' \leftarrow$ Sum.Succ(i)
   Vrátíme $P_{i'}.\min + i'\sqrt{U}$
5. Jinak:
   Vrátíme P_i.Succ(j) $+ i\sqrt{U}$

$P_i.\min = \emptyset$

na každé úrovni $O(1)$ práce ⇒ celk. čas $O(\log\log U)$

Insert(x):
1. Pokud min = ∅, max ← x, min ← x a skončíme.
   1'. Pokud x = min, skončíme [hodnota už ve stromu je]
2. Pokud $x < \min$: $x \leftrightarrow \min$
3. Pokud $x > \max$: $\max \leftarrow x$
4. Rozdělíme x na (i, j).
~~5. Pokud $P_i \neq \emptyset$:~~
~~P_i.Insert(j)~~
5. Pokud $P_i = \emptyset$:
   [Založíme prázdnou P_i.]
   Sum.Insert(i)
6. P_i.Insert(j)

Pokud proběhne 5. krok (zakládáme P_i), 6. krok je triviální
⇒ jen 1 netriv. rekurzivní volání
⇒ celkově $O(\log\log U)$

Delete(x):
1. Pokud $min = \emptyset$, skončíme
2. Pokud $x = min$:
   Je-li $Sum.min = \emptyset$: $min, max \leftarrow \emptyset$ a skončíme
   Jinak $min \leftarrow P_{Sum.min}.min + Sum.min\sqrt{U}$, $x \leftarrow min$
3. Rozdělíme x na (i, j).
4. Pokud $P_i = \emptyset$, skončíme.
5. $P_i.Delete(j)$
6. Pokud $P_i = \emptyset$: $Sum.Delete(i)$
7. Pokud $Sum.max \neq \emptyset$: $max \leftarrow P_{Sum.max}.max + Sum.max \cdot \sqrt{U}$
   Jinak $max \leftarrow min$.

na $\forall$ úrovni rekurze čas $O(1)$,
opět je ≤ 1 větev rekurze netriviální
$\Downarrow$
$O(\log \log U)$

- Dobré zprávy: $O(\log \log U)$ na operaci
- Špatné: struktura zabere paměť $O(U)$ a tu je potřeba na začátku vynulovat!

Trik: Pokud náš RAM dovoluje číst neinicializovanou buňku (byť nezaručuje nic o hodnotě), lze inicializaci simulovat.

- Myšlenka: Udržujeme si seznam buněk, do nichž jsme už zapsali.
  Při čtení se podíváme, je-li buňka na seznamu, jinak vrátíme 0.

- 1. pokus: $M[0\ldots]$ - paměť původního RAMu
  $I[0\ldots N]$ - seznam inicializovaných buněk
  N - # inic. buněk

  poskládáno v paměti proloženě

  Read i Write stojí $O(N)$. ... pomalé.

- Zrychlení: přidáme $X[\ldots]$ - index k poli I
  - pokud je i-tá buňka inicializovaná, pak $I[X[i]] = i$.
  - jinak $X[i]$ neinicializované

  Read(i): $x \leftarrow X[i]$
  Pokud $x < N$ a $I[x] = i$, vrátíme $M[i]$.
  Jinak vrátíme 0.

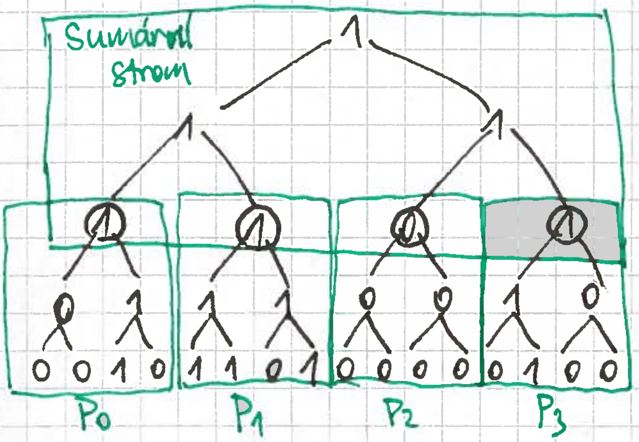
  Write(i,y): $M[i] = y$
  $x \leftarrow X[i]$
  Pokud $x < N$ a $I[x] = i$, skončíme
  $I[N] \leftarrow i$,
  $X[i] \leftarrow N$
  $N{+}{+}$

  $O(1)$

Věta: Kterýkoli ~~algoritmus~~ program se složitostí časovou $T(n)$ a prostorovou $S(n)$, který předpokládá paměť inicializovanou nulami, lze transformovat na program se složitostí $O(T(n))$, $O(S(n))$, který to nepředpokládá.

[potenciální chyták: výstup v neinicializované buňce]

Stromová interpretace vEB (à la intervalový strom)



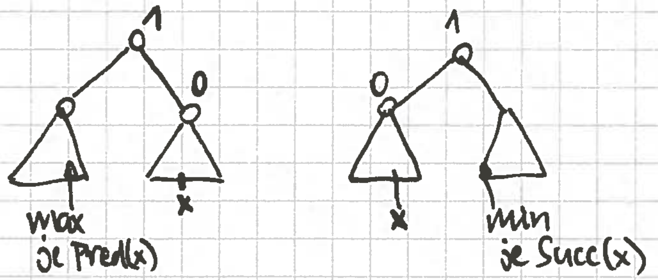
} O(log U) hladin

vnitřní vrcholy obsahují OR listů v podstromu

← v listech indikátorový vektor množiny S

💡 cesta kořen-list je monotónní (1…10…0)

Pred/Succ: Na cestě kořen-list hledáme přechod 1-0 → binárně v $O(\log\log U)$



pro $x \notin S$ dostaneme snadno Pred xor Succ

↓

stačí si prvky S udržovat v seznamu

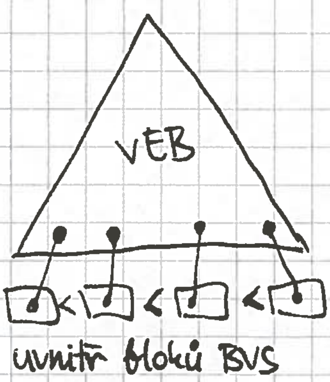
Update trvá $O(\log U)$ … klasický vEB z toho vybrousil ukládáním minima zvlášť

→ použijeme indirekci

Indirekce [Willard 1983]

Myšlenka: množinu rozdělíme na bloky velikosti $\sim B$, $\forall$ blok má reprezentanta v glob. stromu

či spíše setříděnou posl. prvků



Find/Pred/Succ:

- globální strom mi řekne 2 bloky, kde se x může vyskytovat } $O(\log\log U)$

  ↓

- ≤ 2 dotazy na BVS } $O(\log\log U)$

- Update je založený na dělení/slučování bloků.

Invariant: Blok má hustoty $[\frac{1}{4}, 1]$ … tedy # prvků leží mezi $\frac{1}{4}B$ a $\frac{3}{4}B$

Výjimka: $\exists$ jediný blok

Insert: pokud blok přeplním, rozdělím ho na 2 bloky o hustotě $\frac{1}{2} \pm \varepsilon$ … čas $\Theta(B)$

změny reprezentantů → $O(1)$ updatů globálního stromu

<u>Delete</u>: Pokud hustota klesne pod 1/4, podíváme se na souseda:

① soused má hustotu $[\frac{5}{8}, 1]$: půjčíme si od něj $\frac{1}{8}$ → náš blok má $\frac{3}{8}$
soused $[\frac{4}{8}, \frac{7}{8}]$

② soused má $[\frac{2}{8}, \frac{5}{8}]$: sloučíme se s ním → $[\frac{4}{8}, \frac{7}{8}]$

Opět $\Theta(B)$ času + $O(1)$ updatů glob. struktury.

! co když mažeme reprezentanta? ... jen ho škrtneme a zůstane v bloku

<u>Amortizace</u>: Po rozdělení/slití je hustota alespoň o 1/8 vzdálena od mezí
⇒ děje se to nejvýše 1× za B/8 operací ⇒ amortizovaně $O(1/B)$ krát

- tedy amort. $\Theta(1)$ času na op. + $O(1/B)$ updatů glob. struktury.
- pro vEB volíme $B = \log U$ → čas $O(\log\log U)$ amort. na update i čtení
  ... ale prostor stále $\Theta(U)$!

<u>x-fast stromy</u> [Willard 1984]

- vyjdeme z "intervalového" vEB
- uložíme do hešovací tabulky (kukačka či dyn. perf. hešování) pozice všech jedniček [to jsou prefixy bit. zápisů prvků S]
- prostor $O(n \cdot \log U)$ w.c.
- čtení místo stromu čte heš. tabulku → $O(\log\log U)$ w.c.
- zápisy updatují $\Theta(\log U)$ položek heše → $O(\log U)$ průměrně amort.

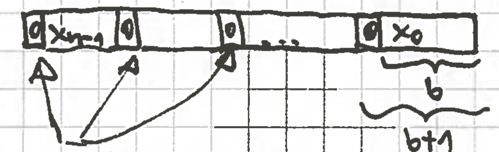
↓ indirekce

<u>y-fast stromy</u>

- čtení $O(\log\log U)$ w.c.
- zápisy $O(\log\log U)$ průměrně amort.
- prostor $O(n)$ w.c. [glob. strom obsahuje $\Theta(n/\log U)$ položek, každá stojí $\Theta(\log U)$ buněk]

<u>RAM jako vektorový počítač:</u>

Vektor $x_0 \ldots x_{n-1} \in [2^b]$, kde $n \cdot b = O(w)$, můžeme reprezentovat číslem v $O(1)$ slovech:
skaláry (značíme řeckými písmeny)



vycpávky (padding)

$\underline{Read(x,i)} = (x >> 2^{(b+1)i})\ \&\ \underbrace{1^b}_{\text{bin. číslo s b jedničkami}}$

$\underline{Write(x,i,\alpha)} = (x\ \&\ 1^{(b+1)(n-i-1)}\, 0^{b+1}\, 1^{(b+1)i}) + (\alpha << (b+1)i)$

konstanty obvykle umíme spočítat v $O(1)$, ale i kdyby ne, míváme dost času na předvýpočet

nebo zavedeme slabě neuniformní RAM: $O(1)$ konstant závislých jenom na w

Replicate(α) … vytvoří vektor $(\alpha, \ldots, \alpha)$ – stačí $\alpha \cdot (0^b 1)^n$

Sum(x) … sečte $x_0 + \ldots + x_{n-1}$, součet se musí vejít do skaláru

① "magické" řešení: $x \bmod 2^{b+1}$ tedy $2^{b+1}-1$ … $\sum_i 2^{(b+1)i} \cdot x_i \overset{\bmod 2^{b+1}-1}{\equiv} \sum_i 1^i x_i \equiv \sum_i x_i$.

② využitím násobení:

$$\begin{array}{ccccccccc}
 & & 0x_{n-1} & 0x_{n-2} & \ldots & 0x_1 & 0x_0 \\
 & \cdot & 0^b1 & 0^b1 & \ldots & 0^b1 & 0^b1 \\
\hline
 & & x_{n-1} & x_{n-2} & \ldots\; x_2 & x_1 & x_0 \\
 & x_{n-1} & x_{n-2} & x_{n-3} & \ldots\; x_1 & x_0 & \\
x_{n-1} & \ldots & & & x_0 & & \\
x_{n-1} \ldots x_1 & x_0 & & & & & \\
\hline
\end{array}$$

$x_0 + x_1 + \ldots + x_{n-1}$

všechny prefixové i suffixové součty

x_0

$x_0 + x_1$

$x_0 + x_1 + x_2$

nedošlo-li k přenosům mezi bloky

Cmp(x,y) $= z$, $z_i = \begin{cases} 1 & \text{pokud } x_i < y_i \\ 0 & \text{jinak} \end{cases}$

odčítáme

$$\begin{array}{ccccc}
 & 1\,x_{n-1} & 1\,x_{n-2} & \ldots & 1\,x_1 \quad 1\,x_0 \\
- & 0\,y_{n-1} & 0\,y_{n-2} & \ldots & 0\,y_1 \quad 0\,y_0 \\
\hline
\end{array}$$

← x OR maska

pokud $x_i \geq y_i$, 1 zůstane

$x_i < y_i$, 1 pohltí přenos a změní se v 0

tedy: $\left(\left(x \mid (10^b)^n\right) - y\right) \gg b \;\&\; (0^b 1)^n$

Min(x,y) $= z$, $z_i = \min(x_i, y_i)$

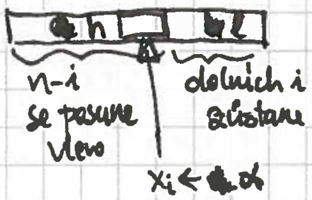
$m \leftarrow \mathrm{Cmp}(x,y) \cdot 1^b$ … $m_i = \begin{cases} 0^b & \text{pokud } x_i \geq y_i \\ 1^b & \text{pokud } x_i < y_i \end{cases}$

$z \leftarrow (x \,\&\, m) \mid (y \,\&\, \sim m)$

Rank(x,α) $= \#i : x_i < \alpha$ … $\mathrm{Sum}(\mathrm{Cmp}(x, \mathrm{Replicate}(\alpha)))$

skalár

Insert(x,α) … vkládání do setříděného vektoru



$i \leftarrow \mathrm{Rank}(x, \alpha)$

$h \leftarrow (x \,\&\, 1^{(b+1)(n-i)} 0^{(b+1)i})$

$l \leftarrow (x \,\&\, 1^{(b+1)i})$

Vrátíme $(h \ll (b+1)) \mid (\alpha \ll (b+1)i) \mid l$.

Unpack$(\alpha) = \mathbf{x} : x_i = \underbrace{i\text{-tý bit čísla } \alpha}_{\alpha[i]}$: $X \leftarrow \text{Replicate}(\alpha)$
$y \leftarrow (2^{b-1}, \dots, 2^0)$
$t \leftarrow x \& y$ .... $t_i = \begin{cases} 0 \\ y_i & \text{pokud } \alpha[i]=1 \end{cases}$
$z \leftarrow \text{Cmp}(t, y) \oplus (0^b1)^n$

... Unpack$_\pi(\alpha)$ ... bity permutuje podle π: pouze se změní maska y: $y_i = 2^{\pi(i)}$

Pack(x) ... inverzní k Unpack

Špinavý trik: "přeformátujeme" vektor: $|0\,0\,0\,0\,x_3|0\,0\,0\,0\,x_2|0\,0\,0\,0\,x_1|0\,0\,0\,0\,x_0|$
$|0\,0\,0\,0|x_3\,0\,0\,0|0\,x_2\,0\,0|0\,0\,x_1\,0|0\,0\,0\,x_0|$
$\downarrow$ Sum (pozor, nekorektně kódovaný vektor $\Rightarrow$ Trik s mod nefunguje, násobení ano.)
$x_3x_2x_1x_0$

Operace s čísly v kvadratické šířce slova $(w = \Omega(b^2))$

Weight(α) ... Hammingova váha, tedy $\sum_i \alpha[i]$ ... stačí Sum(Unpack(α))

Permute$_\pi(\alpha)$ = Pack(Unpack$_\pi(\alpha)$)

LSB(α) ... $\min\{i \mid \alpha[i]=1\}$ ...
α 10000
$\alpha-1$ 01111
$\alpha \oplus (\alpha-1)$ 000011111
} stačí tedy Weight$(\alpha \oplus (\alpha-1)) - 1$

MSB(α) ... $\max\{i \mid \alpha[i]=1\}$ ... třeba $b-1-$LSB(Permute$_{\text{zrcadlení}}(\alpha)$)

Jiná možnost: $\#(\#i: 2^i < \alpha)$ ... tedy Sum(Cmp($(2^{b-1}, \dots, 2^0)$, Replicate(α)))

MSB v prostoru $O(w)$ [podobně LSB] [Brodnik 1993] $\leftarrow$ ale asi to bylo známé i dříve

1. $b \leftarrow \lfloor\sqrt{w}\rfloor$, $\ell \leftarrow b$ ... ℓ bloků po b bitech + padding mezi bloky
2. $x \leftarrow (\alpha \& (01^b)^\ell) \mid ((\alpha \& (10^b)^\ell) \gg b)$ ... $x_i \neq 0 \iff i$-tý blok byl neprázdný (proč tak složitě? Musíme dodržet padding)
3. $y \leftarrow \text{Cmp}(0, x)$ ... $y_i = \begin{cases} 0 \\ 1 & \text{pokud } i\text{-tý blok} \neq 0 \end{cases}$
4. $\beta \leftarrow \text{Pack}(y)$, $p \leftarrow \text{MSB}(\beta)$ ... p = číslo nejvyššího bloku s 1
5. $\gamma \leftarrow (\alpha \gg (b+1)p) \& 1^b$ ... obsah bloku
   $q \leftarrow \text{MSB}(\gamma)$ ... MSB uvnitř bloku
6. Vrátíme $(b+1)p + q$.

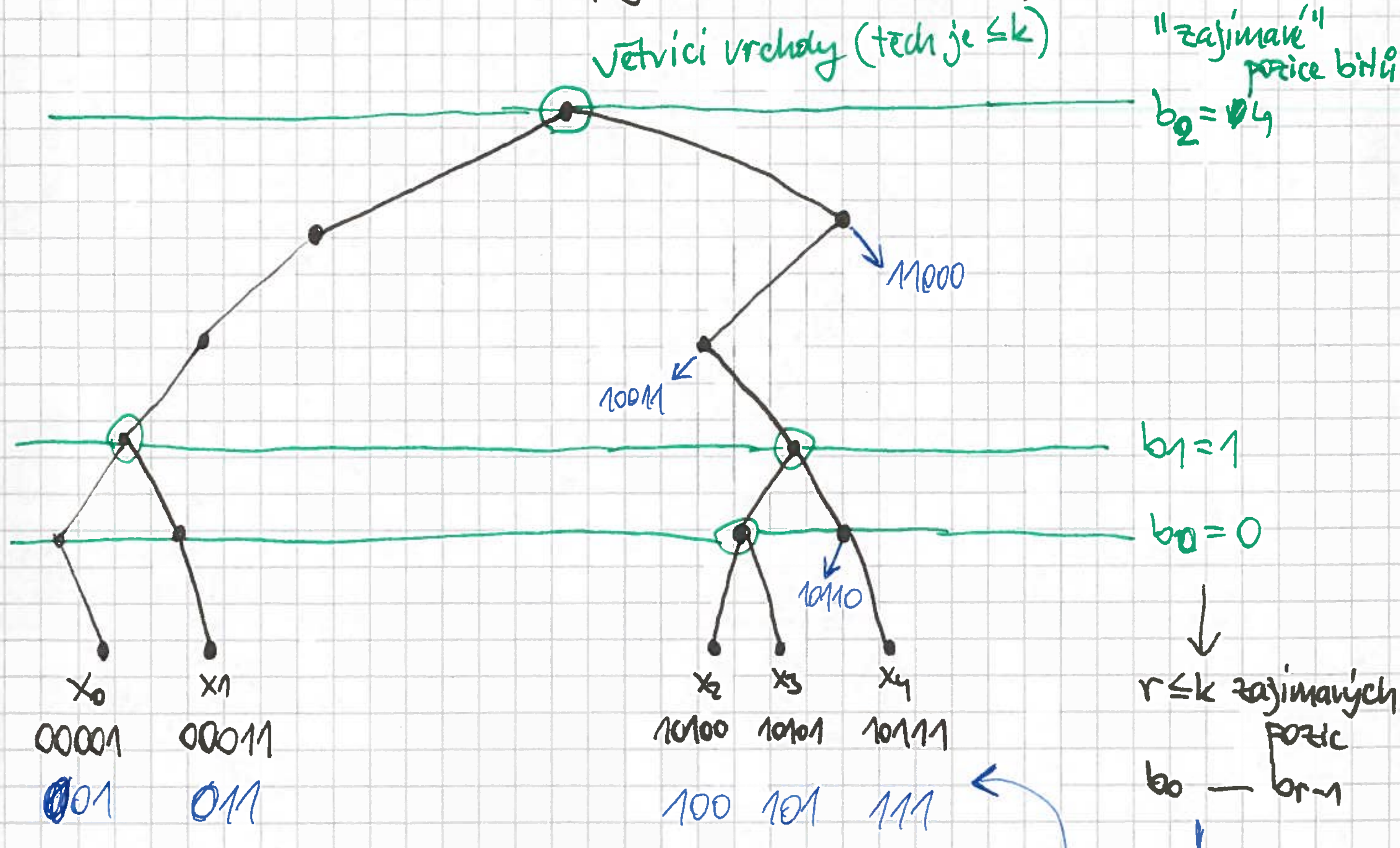
} čas $O(1)$, šířka slova $O(w)$

Fusion trees [Fredman & Willard 1990]

- rozhraní jako vEB stromy, fungují lépe pro širší slova (větší universum)
- předvedeme statickou verzi, později použijeme obecnou dynamizační techniku. (exponenciální stromy)
- Fusion node – pamatuje si k klíčů $x_0 < x_1 < \_\_ < x_{k-1}$, $k = O(w^{1/5})$ (w: šířka slova)
  - umí v konst. čase spočítat $\mathrm{Rank}(q) := \#i : x_i < q$
    → z toho Pred, Succ v $O(1)$
  - konstrukce v čase $\mathrm{Poly}(k)$
- Fusion tree – B-strom pro $B = \Theta(w^{1/5})$, ve vrcholech jsou Fusion nodes
  - hloubka $O(\log_w n) = O\left(\frac{\log n}{\log w}\right)$ … čím delší slovo, tím lepší
  - Rank počítá v $O\left(\frac{\log n}{\log w}\right)$

Konstrukce Fusion nodes

- uvažujme trii nad binárními zápisy klíčů (má hloubku w)



↓ $r \le k$ zajímavých pozic b_0 — b_{r-1}

↓ náčrtek (sketch) čísla $s(x)$ – zajímavé bity "sklepány" k sobě

$s(x_0) < s(x_1) < \_\_ < s(x_{k-1})$

… pokud hledáme $q \in X$, stačí hledat $s(q)$ mezi $\{s(x_i)\}$

- všechna $s(x_i)$ mají $r \cdot k \le k^2 = O(w^{2/5})$ bitů
  → vejdou se do $O(1)$ slov → stačí paralelní Cmp. "fusion"

Co když hledáme $q \notin X$? $s(q)$ nás na 1. pohled zavede na scestí...

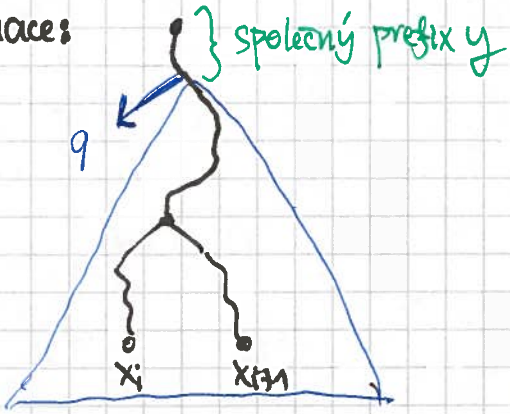
Např. pro $q = 11000$ je $s(q) = 100$, což nás zavede do x_2, ačkoli $Rank(q) = 5$

Přesto z toho něco odvodíme: Nechť $s(x_i) \le s(q) < s(x_{i+1})$ ...

Spočítáme $MSB(q \oplus x_i)$, $MSB(q \oplus x_{i+1})$ a $MSB(x_i \oplus x_{i+1})$

místo, kde cesta do q odbočí od cesty do x_i

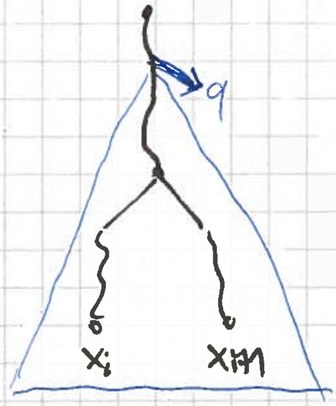
Možné situace:



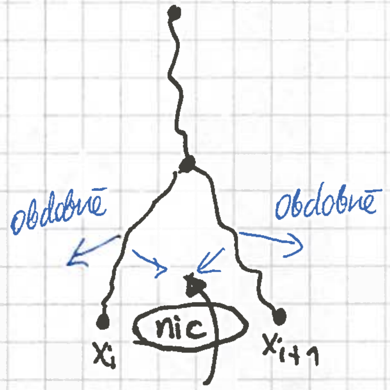
Succ(q) je Min modrého podstromu

$\Downarrow$

necháme se vést $s(y10\ldots0)$



K Pred(q) nás vede $s(y01\ldots1)$



zde je q mezi x_i a x_{i+1}

Takže stačí umět počítat v $O(1)$ $s(x)$ a MSB. Z toho Rank v $O(1)$.

tohle bohužel neumíme (pro $s(x)$); viz vektorové triky (pro MSB)

Budeme počítat ~~přibližné~~ přibližné sketche $a(x)$ ... délky $O(r^4) \le O(w^{4/5})$ } vektory stále mají $O(1)$ slov

... $s(x)$ "prostrkané nulami"

→ stále platí $a(x_i) < a(x_{i+1})$

Princip: $x' = x \,\&\, \sum_i 2^{b_i}$ ... vynulujeme nezajímavé bity

$$x' \cdot M = \left(\sum_{i=0}^{r-1} x[b_i]\cdot 2^{b_i}\right)\cdot\left(\sum_{j=0}^{r-1} 2^{m_j}\right) = \sum_{i=0}^{r-1}\sum_{j=0}^{r-1} x[b_i]\cdot 2^{b_i+m_j}$$

(M … pomocná maska)

$$a(x) := \left((x'\cdot M) \,\&\, \left(\sum_i 2^{b_i+m_i}\right)\right) \gg (b_0+m_0)$$

Lemma: Pro $b_0 < \ldots < b_{r-1}$ existují $m_0, \ldots, m_{r-1}$ taková, že:

1. všechna $b_i + m_j$ jsou navzájem různá (nevzniknou kolize)
2. $b_0 + m_0 < \ldots < b_{r-1} + m_{r-1}$ (zachováváme pořadí)
3. $(b_{r-1} + m_{r-1}) - (b_0 + m_0) = O(r^4)$ (malé rozpětí)

Dk: (A) Najdeme $m_0' - m_{n-1}' < r^3$ tak, aby $b_i + m_j'$ byly různé modulo r^3.

Indukcí: Máme-li $m_0' - m_{t-1}'$ a hledáme m_t'

Chceme se vyhnout $m_i' + b_j \equiv m_t' + b_k$

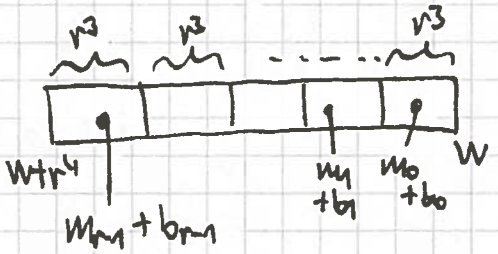
$$m_t' \equiv \underbrace{m_i'}_{t \text{ možností}} + \underbrace{b_j}_{r} - \underbrace{b_k}_{r} \Big\} \; tr^2 < r^3 \text{ možností}$$

$\Downarrow$ aspoň 1 volná $\rightarrow$ m_t' lze zvolit

(B) Posuneme m_i o násobky r^3, aby $\forall i$ $m_i + b_i \in [i \cdot r^3, (i+1) r^3)$

$\downarrow$ m_i

$\downarrow$ to by mi mohla vycházet záporná, tak ke všem přičteme w



... $m_i + b_j$ jsou stále mod r^3 různá, takže bez mod také.

Shrnutí:

vEB odpovídá v čase $O(\log w)$ ... s w roste

FT v čase $O\left(\frac{\log n}{\log w}\right)$ ... s w klesá

} vyrovná se pro $\log n \approx \log^2 w$ $\Downarrow$ $O(\sqrt{\log n})$

Dolní odhad [Sen & Venkatesh 2006] pro Cell Probe

Princip: studujeme interaktivní protokol:

Alice (zná dotaz) $\xrightarrow{\text{a bitů adresy}}$ Bob (zná strukturu)

$\xleftarrow{\text{b bitů dat}}$

Věta: # cell probes $= \Omega(\min(\log_a w, \log_b n))$.

[bez důkazu]

Důsledek: pro $a = \Theta(\log n)$ [Poly prostor], $b = w$ $\rightarrow$ $\frac{\log w}{\log \log n}$ téměř vEB, $\frac{\log n}{\log w}$ Fusion Tree