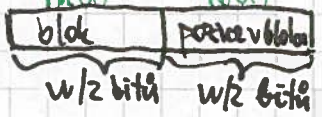
Celočíselné datové struktury ... vše na RAMu

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

- 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 $\sqrt{U}$ — předpokládáme $w = 2^k$, tedy $U = 2^{2^k}$
  - binární pohled: 

... rozklad v $O(1)$ na RAMu

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

- min S (zvlášť!) ... $S' := S \setminus \{\min S\}$ } pokud je strom prázdný, min = max = ∅
- max S (kopie)
- přihrádky $P_0, \dots, P_{\sqrt{U}-1}$ ... do nich rozstříháme S'
  - uvnitř P_i je vEB($\sqrt{U}$) s dolními polovinami prvků $P_i := \{lo(x) \mid x \in S' \wedge hi(x) = i\}$
- 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 $j \geq P_i.\max$:
   $i' \leftarrow Sum.Succ(i)$
   Vrátíme $P_{i'}.\min + i'\sqrt{U}$
5. Jinak:
   Vrátíme $P_i.Succ(j) + i\sqrt{U}$

$P_i.\min = \emptyset$ ← jelikož $x < \max$, i' určitě existuje

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

Insert(x):
1. Pokud $\min = \emptyset$, $\min \leftarrow x$, $\max \leftarrow x$ a skončíme.
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)

1'. Pokud $x = \min$, skončíme
[hodnota už ve stromu je]

Pokud proběhne 5. krok (založíme 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 $min \leftarrow P_{Sum.min}.min$, $x \leftarrow min$ } + $Sum.min\sqrt{U}$
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
⇓
$O(\log\log U)$

- 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 negarantuje 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.
  $I[N] \leftarrow i$
  $X[i] \leftarrow N$
  $N{+}{+}$

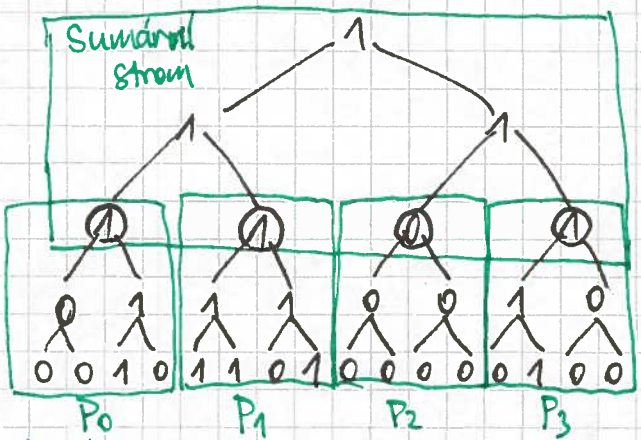
  } $O(1)$

Věta: Kterýkoli 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] → vstup je potřeba na začátku konvertovat, výstup na konci taktéž → proto ⊛

⊛ $+n$ velikost výstupu

Stromová interpretace vEB (à la intervalový strom)



$O(\log U)$ hladin

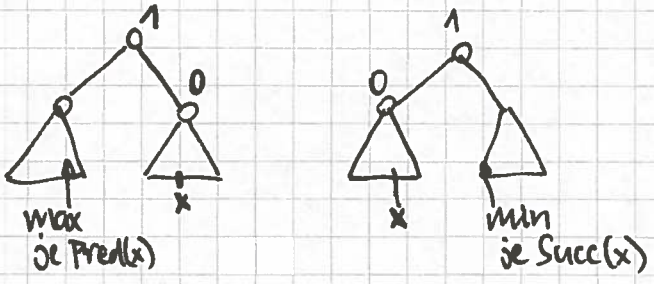
vnitřní vrcholy obsahují OR listů v podstromu

cesta kořen-list je monotónní (1...10...0)

← v listech indikátorový vektor množiny S

Min/Max: Jdu z kořene, pokud jsou pode mnou 2 jedničky, preferuji levou/pravou.

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

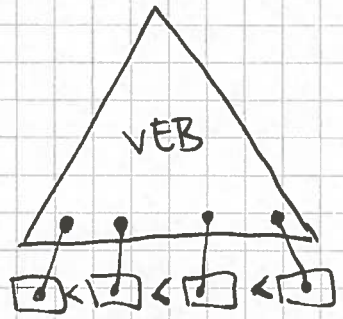
→ použijeme indirekci

Indirekce [Willard 1983]

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

(či spíše setříděnou pož. prvků)

B nastavíme na $\Theta(\log U)$



uvnitř bloků BVS ... operace v $O(\log B) = O(\log \log U)$

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 hustotu $[\frac{1}{4}, 1]$ — tedy # prvků leží mezi $\frac{1}{4}B$ a B

Výjimka: ∃ 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

Delete: 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 nemáme reprezentanta? ... jen ho škrtneme a zůstane v bloku

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

x-fast stromy [Willard 1984]

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

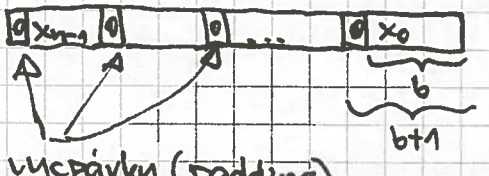
↓ indirekce

y-fast stromy

- 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]

RAM jako vektorový počítač:

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



nebo zavedeme slabé nevážený RAM: $O(1)$ konstant závislých jenom na w

výplňky (padding)

konstanty, o kterých umíme spočítat v $O(1)$, ale i kdyby ne, umíráme dost času na předvýpočet

$\text{Read}(x,i) = (x \gg 2^{(b+1)i}) \,\&\, 1^b$ (b bin. číslo)

$\text{Write}(x,i,\alpha) = (x \,\&\, 1^{(b+1)(n-i-1)} 0^{b+1} 1^{(b+1)i}) + (\alpha \ll (b+1)i)$ (s b jedničkami)

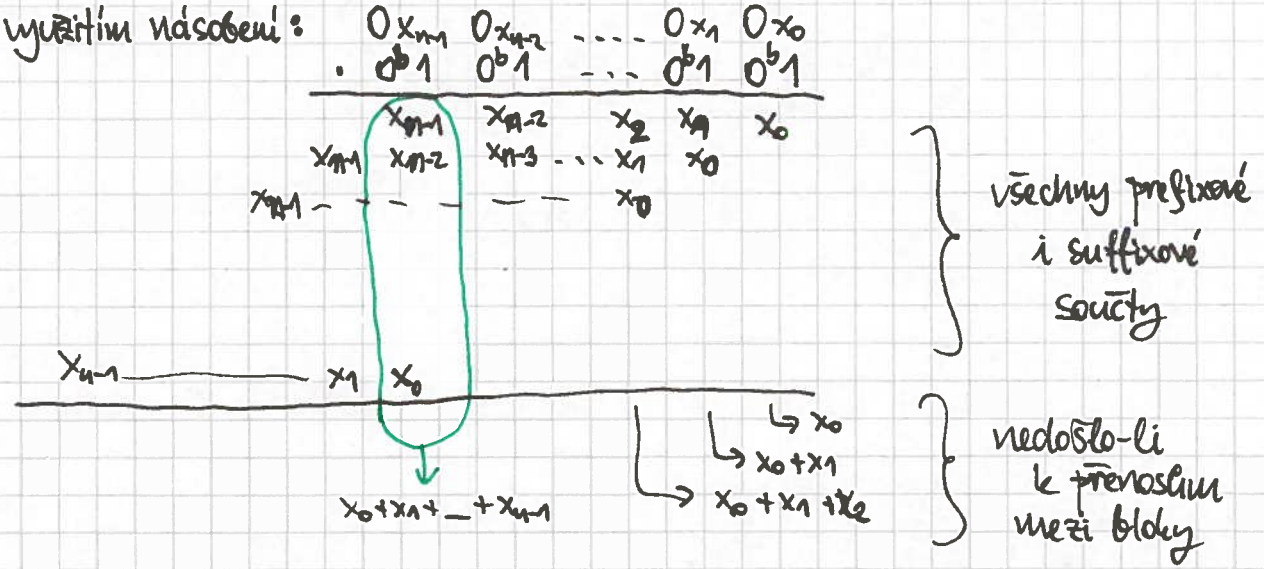
Replicate(α) ... vytvoří vektor $(\alpha_1, \_, \alpha)$ – stačí $\alpha \cdot (0^b 1)^n$

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

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

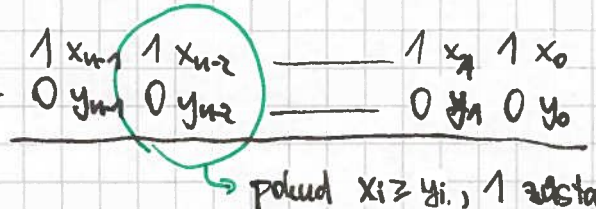
tedy $2^{b+1}-1$

② využitím násobení:



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

odečítáme:



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

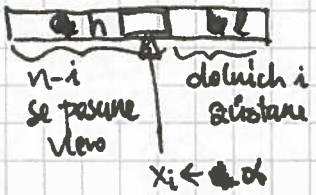
Min(x,y) = z, $z_i = \min(x_i, y_i)$ $\quad m \leftarrow \text{Cmp}(x,y) \cdot 1^b \ \ldots\ m_i = \begin{cases} 0 & \text{pokud } x_i \geq y_i \\ 1^b & \text{pokud } x_i < y_i \end{cases}$

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

Rank(x, α) = i ... $\#i : x_i < \alpha$ ... Sum(Cmp(x, Replicate(α)))

α: skalár

Insert(x, α) ... vkládání do setříděného vektoru



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

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

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

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

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

... $\underline{Unpack_\Pi(\alpha)}$ ... bity permutuje podle α: pouze se změní maska y: $y_i = 2^{\Pi(i)}$

$\underline{Pack(x)}$ ... inverzní k Unpack

Špinavý trik: "přeformátujeme" vektor: $\begin{array}{|c|c|c|c|} \hline 0\,0\,0\,0 & x_3 & 0\,0\,0\,0 & x_2 & 0\,0\,0\,0 & x_1 & 0\,0\,0\,0 & x_0 \\ \hline 0\,0\,0\,0 & x_3\,0\,0\,0 & 0\,0\,0\,0 & x_2\,0\,0\,0 & 0\,0\,0\,0 & x_1\,0\,0\,0 & 0\,0\,0\,0 & x_0 \\ \hline \end{array}$

$x_3 x_2 x_1 x_0$ ↓ Sum (pozor, nekorektně kódovaný vektor & Trik s mod nefunguje, násobení ano.)

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

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

$\underline{Permute_\Pi(\alpha)} = Pack(Unpack_\Pi(\alpha))$

$\underline{LSB(\alpha)}$ ... $\min\{i \mid \alpha[i]=1\}$ ... α 10000
$\alpha-1$ 01111
$\alpha \oplus (\alpha-1)$ 0000 1111

} stačí tedy $Weight(\alpha \oplus (\alpha-1)) - 1$

$\underline{MSB(\alpha)}$ ... $\max\{i \mid \alpha[i]=1\}$ ... třeba $b-1-LSB(Permute_{zrcadlení}(\alpha))$

Jiná možnost: $\#(i : 2^i \leq \alpha)$ ... tedy $Sum(Cmp((2^{b-1}, \dots, 2^0), Replicate(\alpha)))$
→ tohle je Rank vzhledem k $(2^{b-1}, \dots, 2^0)$

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

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

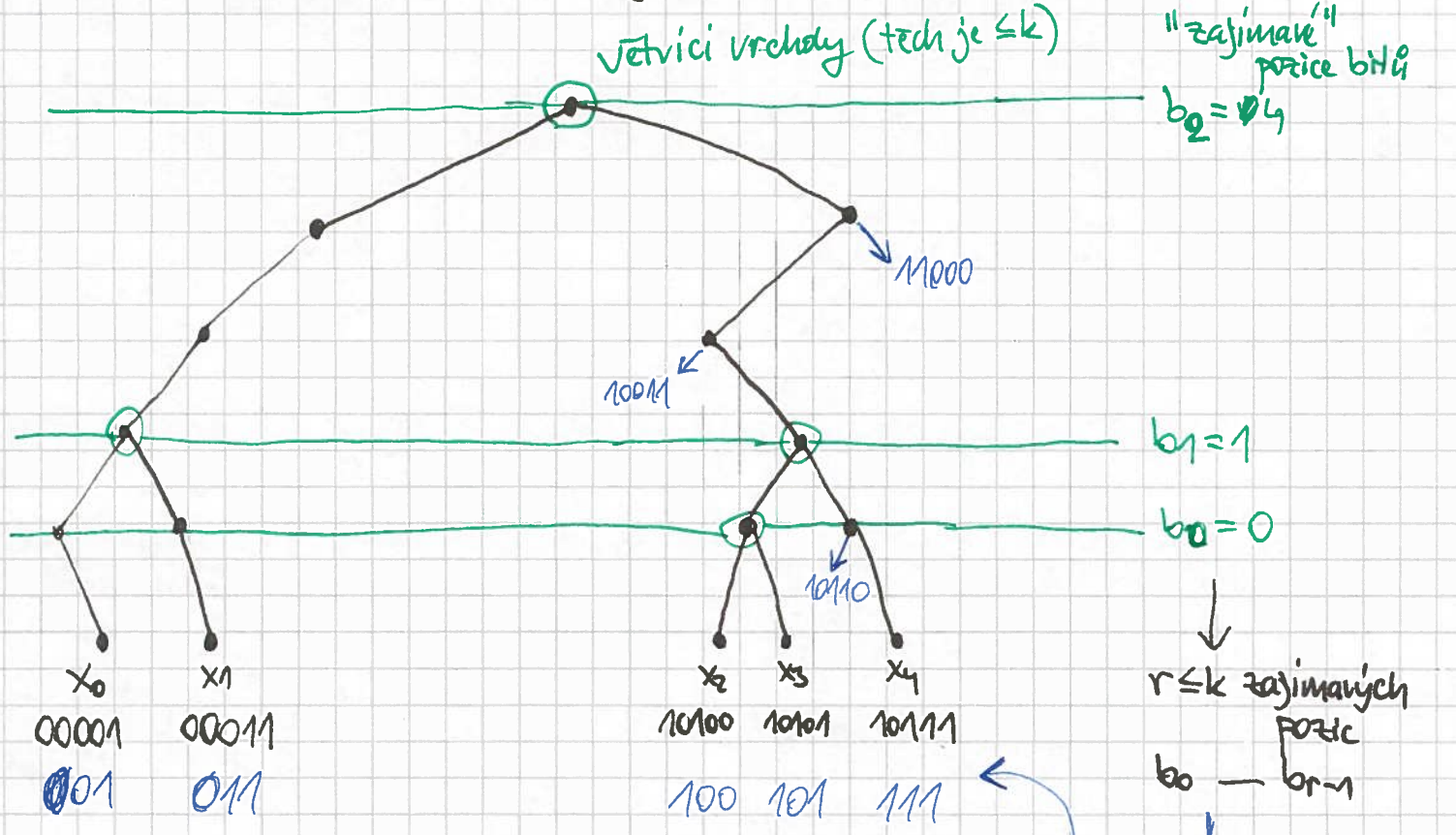
} čas $O(1)$
šířka slova $O(w)$

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 $x_0 < x_1 < \dots < x_{k-1}$, $k = O(w^{1/5})$ (šířka slova)
  - umí v konst. čase spočítat $\text{Rank}(q) = \#i : x_i < q$
    → z toho Pred, Succ v $O(1)$
  - konstrukce v čase $\text{Poly}(k)$
- **Fusion tree** - B-strom pro $B = \Theta(w^{1/5})$, ve vrcholech jsou Fusion nodes
  - hloubka $O(\log_w n) = O(\frac{\log n}{\log w})$ ... čím delší slovo, tím lepší
  - Rank počítá v $O(\frac{\log n}{\log w})$

Konstrukce Fusion nodes

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



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

... 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...

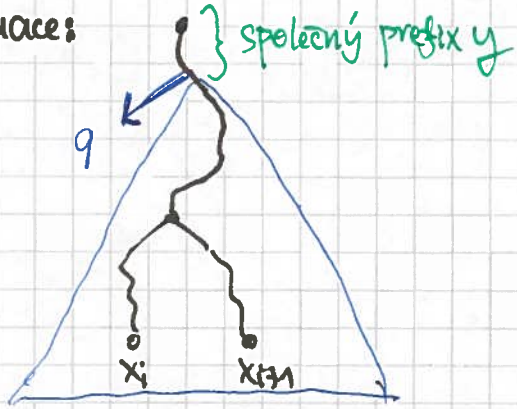
Např. pro $q = 11000$ je $s(q) = 100$, což nás zavede do x_2, ačkoli $Rank(q) = 5$

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

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

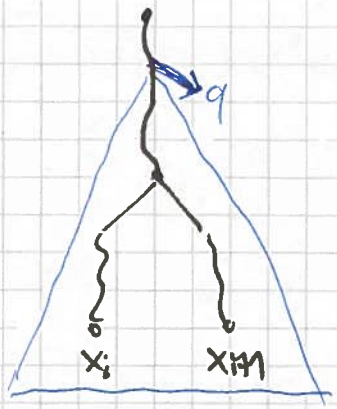
Možné situace:



Succ(q) je Min modrého podstromu

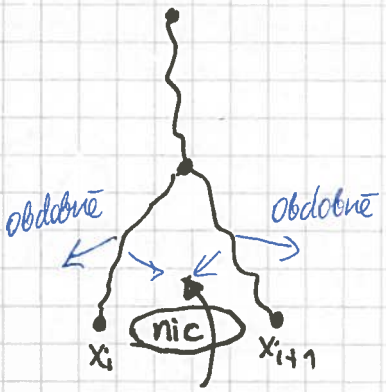
⇓

necháme se vést $s(y10\_\_0)$



K Pred(q) nás vede

$s(y01\_\_1)$



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 $O(1)$,

↓ ↓

vše velitorové triky

tohle bohužel neumíme

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

... $s(x)$ "prostrkané nulami"

→ stále platí $a(x_i) < a(x_{i+1})$

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

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

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

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

① všechna $b_i + m_j$ jsou navzájem různá (nevzniknou kolize)

② $b_0 + m_0 < \_\_\_ < b_{r-1} + m_{r-1}$ (zachováme pořadí)

③ $(b_{r-1} + m_{r-1}) - (b_0 + m_0) = O(r^4)$ (malé rozpětí)

Dk: (A) Najdeme $m_0 - m_{r-1} < r^3$ 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 $m_i + b_j \equiv m_t + b_\ell$ pro všechna $i<t$, $j,\ell<r$

$m_t \equiv m_i + b_j - b_\ell$

t možností, r, r } $tr^2 < r^3$ možností ⇓ aspoň 1 volná

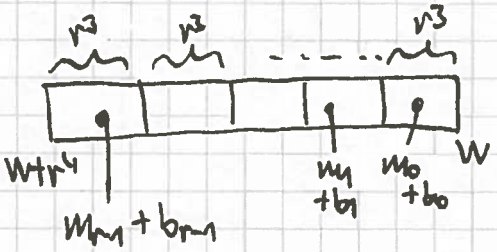
m_t lze zvolit ← aspoň 1 volná

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

m_i ↓

$m_i + b_i$ ↓

to by m_i mohla vyčnívat zdroma, tak je 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 $\log n \approx \log^2 w$ ⇓ $O(\sqrt{\log n})$

Dolní odhad [Sen & Venkatesh 2006] pro Cell Probe

Princip: studujeme interaktivní protokol:

Alice —(a bitů adresy)→ Bob

Alice: zná dotaz

Bob: zná strukturu

Bob —(b bitů dat)→ Alice

Statická DS pro hledání předchůdce má

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

[bez důkazu]

$\log_a w$: $a = w$, $\frac{\log w}{\log \log n}$, téměř vEB

$\log_b n$: $b = w$, $\frac{\log n}{\log w}$, Fusion Tree

Důsledek: pro $a = b = w$ je $\Omega(\log n)$ [Poly prostor]