Úsporné datové struktury

- Chceme uložit prvek množiny C (třeba jeden ze stromů na n vrcholech) a použít téměř optimální prostor ... $OPT = \lceil \log |C| \rceil$ bitů } entropie při rovnoměrném rozdělení psti
- Přitom chceme rychlé operace
- Varianty:
  - implicitní DS – jen data sama ve vhodném pořadí (setříděné pole, halda)] $OPT + O(1)$ ← rounding
  - úsporné (succinct) DS – $OPT + o(OPT)$
  - kompaktní DS – $O(OPT)$ ... ale pozor, třeba BVS nemusí být (nebo trie) kompaktní kvůli pointerům – nepočítáme slova, ale bity → lze obejít pakováním do nafukovacího pole
    (někdy dovolujeme $O(1)$ slov – vejde se n a pointer)

Problém: Reprezentace řetězce nad obecnou abecedou

Příklad: $[10] \to 4$ bity ($\log 10 \doteq 3.322$)
$[10]^2$ ... mohu uložit jako $2 \times 4b$ nebo jako $[100] \to 7$ bitů
$[10]^k$ ... uložím jako $[10^k] \to k \cdot \log 10 + 1$ namísto $k \cdot (\log 10 + 1)$
↳ limitně se blížím $k \log 10$ b/znak (redundance = délka kódu − entropie jde k 0)
... ale ztrácím lokální dekódovatelnost

Praktické řešení: dělím na bloky o 9 znacích ... délka kódu = 30
entropie $9 \cdot \log 10 \doteq 29.897$
→ redundance $\doteq 0.103$

Blok se vejde do 1 slova → počítám s ním v $O(1)$ času i prostoru

↳ pro string délky n: $30 \cdot \lceil n/9 \rceil \leq 30/9\, n + 30$
↳ redundance $\leq 0.103 \cdot \frac{n}{9} + 30$ ($\doteq 0.0114n$)

↳ $(a,b) \in X \times Y$ kóduji po složkách ⇒ délky kódů i entropie se sčítají → redundance také

Příklad #2: Posíláme proud bitů, ale dopředu nevíme, jak bude dlouhý. → instantní (prefixový) kód
[potřebujeme umět zakódovat značku pro konec, aniž by neustálé „ještě to pokračuje" zabralo dost místa]

Triviální řešení: rozdělíme na bloky o b bitech, za $\forall$ blok připíšeme 1 bitovou značku, do posledního bloku padding 10^k
↳ redundance $\frac{n}{b} + b + 1$

SOLE Encoding [Dodis, Pătraşcu, Thorup 2010]
(Short-Odd-Long-Even)

- vstup rozdělíme na bloky o b bitech → prvky abecedy $[B]$ pro $B = 2^b$
- poslední blok dopadujeme ($10 \ldots 0$ na konec), ale potřebujeme přidat spec. blok označující EOF
  ↳ převod $[B+1]^* \to [B]^*$
- nastavíme $b \geq 2\log n + 2 \Rightarrow B \geq 4n^2$] bloky se vejdou do $O(1)$ slov RAMu
  (n = horní mez na délku zprávy)

číslo bloku	1	2	3	4	5	6	...	k
vstupní abeceda	B	B	B	B	B	B	...	EOF
+ EOF	B+1	B+1	B+1	B+1	B+1	B+1		B+1
1. průchod	B	B+3	B-3	B+6	B-6	B+9	...	0
2. průchod	B	B	B	B	B	B	...	

← sem si domyslíme nulový blok

1. průchod: $(B+1)(B+1) \leq (B-3i)(B+3i+3)$

$$\cancel{B^2} + 2B + 1 \leq \cancel{B^2} + \cancel{3Bi} + 3B - \cancel{3Bi} - 9i^2 - 9i$$

$$-B + 1 \leq -9i^2 - 9i$$

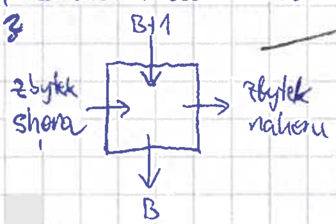
$$B - 1 \geq 9i^2 + 9i$$

$$B \geq 9i^2 + 9i + 1, \quad \text{neboť } i \leq \frac{n+1}{2} \text{ a } B \geq 4n^2$$

2. průchod: $(B+3i)(B-3i) = B^2 - 9i^2 \leq B^2$

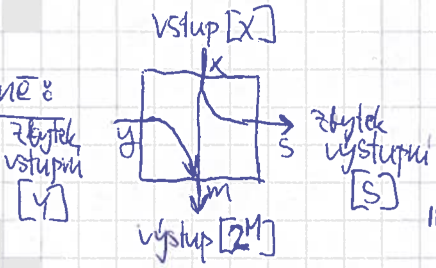
- redundance $\leq 3b$ (1 blok zaokrouhlení + 1 EOF + 1 extra na každý # bloků)
- proudové kódování + dekódování
- lokální dekódování i změny v $O(1)$ na RAMu [potřebuji $\Theta(\log n)$ bitů]
  ↳ $O(1)$ slov

Postupně z toho odvodíme obecnou konverzi abeced...
- proč to vlastně fungovalo?



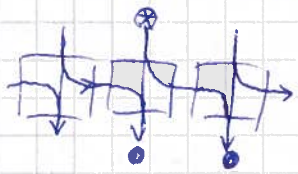
nechám zbýt malé množství informace a to v příštím kroku přičítám k většímu, čímž snížím redundanci
zaokrouhlení na celé bity je příliš redundantní
... zbytky se posílají jen do omezené vzdálenosti
→ lokálně dekódovatelné

Obecně:



"MIXÉR"

... při řetězení stále lok. dekódovatelné



pro dekódování ⊛ stačí znát oba ●

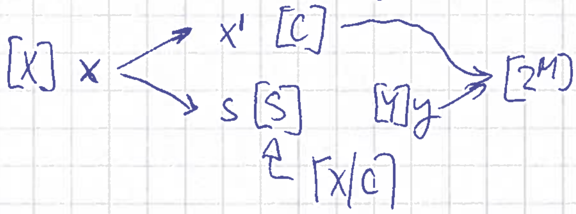
Lemma: Nechť $X, Y \leq 2^w$. Pak $\exists M, S$ a zobrazení $f: [X] \times [Y] \to [2^M] \times [S]$ t. ž.:

1. $S \in O(\sqrt{X})$, $2^M \in O(Y \cdot \sqrt{X})$ a S, M závisí pouze na X, Y
2. f lze vyhodnotit (na RAMu) v čase $O(1)$
3. x lze dekódovat z m, s v $O(1)$
4. y lze dekódovat ze samotného m v $O(1)$
5. Redundance je $O(1/\sqrt{X})$ bitů

↳ $(M + \log S) - (\log X + \log Y)$

entropie výstupu — entropie vstupu

Dk: Zvolíme M (později)
↳ $[2^M]$ musí obsahovat celé y a nějakou část x' čísla x … jakou? $C := \lfloor 2^M / Y \rfloor$ možností



dokud každý mezivýsledek používáme jen 1x, redundance můžeme sčítat (steleskopují se)

- redundance kódování $x', y \to m$:

$$R_1 = M - \log(Y \cdot C) \le M - \log(2^M - Y) = \log \frac{2^M}{2^M - Y} = \log \frac{1}{1 - \frac{Y}{2^M}} = O\left(\frac{Y}{2^M}\right) = O\left(\frac{1}{C}\right)$$

$\log(Y \cdot C)$ ↳ $\log\left(Y \cdot \left\lfloor \frac{2^M}{Y} \right\rfloor\right)$, kde $Y \cdot \lfloor 2^M / Y \rfloor \ge 2^M - Y$

~~$e^x \ge 1+x$, $x \ge \log(1+x)$, $-x \le \log\frac{1}{1+x}$, $x \ge \log\frac{1}{1-x}$~~

z toho vyjde opačná nerovnost – správná je v ⊛ na straně (4)

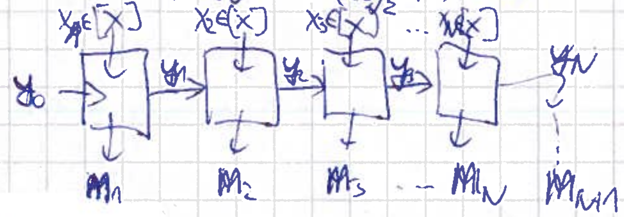
- redundance rozkladu $x \to x' s$:

$$R_2 = (\log C + \log S) - \log X = \log C + \log \lceil X/C \rceil - \log X = \log \frac{C \cdot \lceil X/C \rceil}{X} \le \log \frac{X+C}{X} =$$
$$= \log\left(1 + \frac{C}{X}\right) = O\left(\frac{C}{X}\right).$$

- celkem tedy minimalizujeme $O\left(\frac{1}{C} + \frac{C}{X}\right)$ … $C \approx \sqrt{X} \to S = O(\sqrt{X})$, $2^M = O(Y \cdot \sqrt{X})$
→ redundance $O(1/\sqrt{X})$, jak jsme slíbili. [nezávisí na Y]

První pokus o reprezentaci stringu

$A \in [\Sigma]^n$ … rozdělíme na bloky o velikosti $\Theta(\log_\Sigma n)$ tak, aby $X \approx n^2$
$\Rightarrow N \in O\left(\frac{n}{\log_\Sigma n}\right)$ bloků



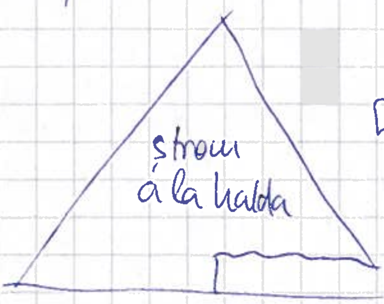
$y_i \in [O(\sqrt{X})] = [O(n)]$
$m_i \in [O(\sqrt{X} \cdot \sqrt{X})] = [O(n^2)]$

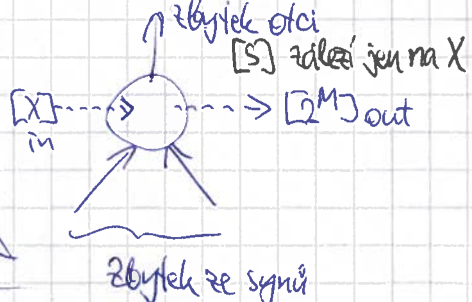
… redundance $O\left(\frac{1}{n}\right)$ v každém bloku
↳ celkově $O(1)$, což je milé
… lokální kódování/dekódování
OOPS! Každý blok má jiné parametry!
→ potřebujeme tabulku $O\left(\frac{n}{\log n}\right)$ konstant!
↳ tohle v SOLE nevadilo, protože $X = 2^b + 1$, takže jsme jednotlivá Y_i uměli aproximovat aritmetickou posloupností … ale obecně to nebude fungovat

↳ zaokrouhlení na bloky netriviální, víme délku, takže pod. blok můžeme udělat atypický

Záchrana: stromové kódování

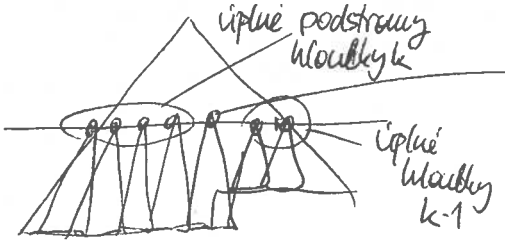
Opět volíme $X := \Theta(n^2)$, $N = O\left(\frac{n}{\log_\Sigma n}\right)$





$[Y_1] \times [Y_2] \cong [Y_1 \cdot Y_2]$

- dekódování je stále lokální
(z otce stačí out, ten určuje zbytek)



1 spec. podstrom

na každé hladině 3 typy vrcholů
pro ně si pamatujeme:
- # vrcholů tohoto typu
- adresu dat 1. vrcholu
- parametry mixéru

celkem $O(\log n)$ konstant (slov)

↓ + potřebujeme tabulku $O(\log n)$ mocnin $|\Sigma|$ na extrakci symbolů z bloku

Dekódování: pozice ve streamu → číslo bloku = pozice ve stromu

log i (v čase $O(1)$)

hladina → typ → adresa v paměti

vše $O(1)$

Lokální změna též v $O(1)$.

Celková redundance $O(1)$ [jako u předch. postupů] + $O(1)$
- $O(1)$ za kódování bloků
- $O(1)$ zaokrouhlení posledního (neúplného) bloku (ten kódujeme atypicky)

Věta: Na Word-RAMu lze reprezentovat prvek $[\Sigma]^n$
v prostoru $\lceil n \log \Sigma \rceil + O(1)$ bitů
se čtením/zápisem prvku v čase $O(1)$
s použitím $O(\log n)$ konstant závislých na n a Σ.

⊛ Chceme $\log \frac{1}{1-x} \leq cx$ pro nějaké $c>0$ a $\forall x \in [0, 1/3]$.

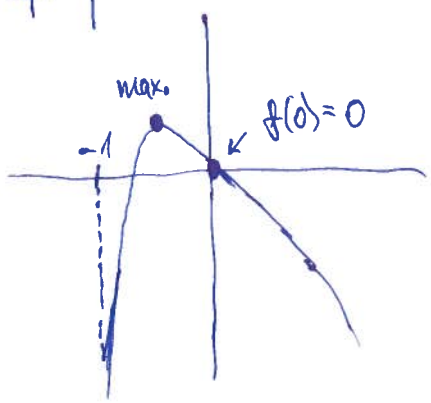
Tedy: $-\log(1-x) \leq cx$

$\log(1-x) \geq -cx$

$\log(1+x) \geq cx$ pro $x \in (-1/2, 0]$

Uvažme $f(x) := \log(1+x) - cx$ ... chceme $f(x) \geq 0$ pro $x \in (-1/2, 0]$

Např. pro $c=2$



Ukážeme, že pro dost velké $c>1$ má f maximum vlevo od $x=-1/2$.

$f'(x) = \frac{1}{1+x} - c$ ... $f'(0) < 0$

Hledáme $f'(x)=0$:

$\frac{1}{1+x} - c = 0$

$\frac{1}{1+x} = c$

$1+x = \frac{1}{c}$

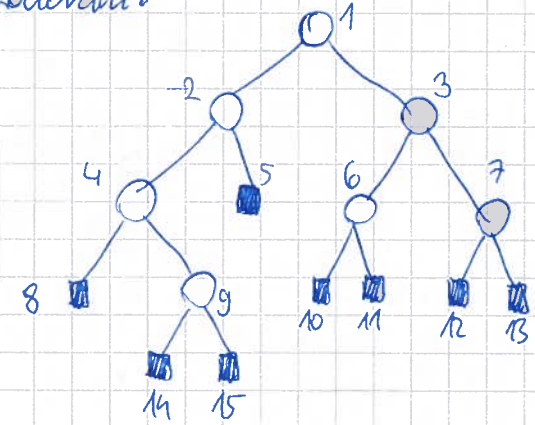
$x = \frac{1}{c} - 1$

a to dokážeme dostat volbou c libovolně blízko k -1

... stačí $c>2$ pro max. před $-1/2$

Úsporná reprezentace binárních stromů

- chceme uložit strukturu stromu, umět hledat syny, otce atd.
- bin. stromů s n vrcholy existuje $C_n = \frac{1}{n+1}\binom{2n}{n} \approx 4^n / poly(n)$
  ↳ $2n + o(n)$ bitů
- kódování:



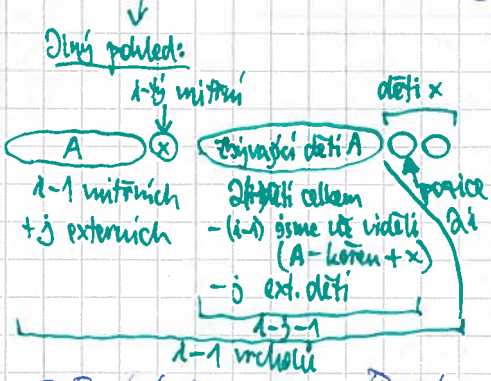
  - přidáme externí vrcholy
  - všechny vrcholy očíslujeme po hladinách à la halda
  - za ∀ vrchol zapíšeme 1 bit $\begin{cases} 0 & \text{pro externí} \\ 1 & \text{pro interní} \end{cases}$
  ↓
  $2n+1$ bitů

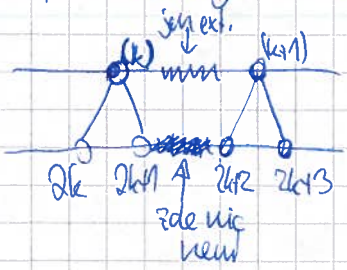
- adresace:

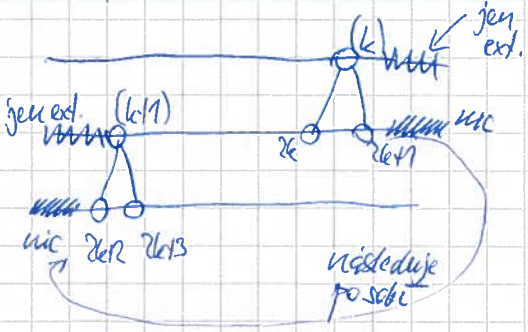
Lemma: Synové k-tého vnitřního vrcholu leží na pozicích $2k$ a $2k+1$
(počítáno od 1 po hladinách; opět od 1)

Dk: Indukcí podle k: Pro $k=1$ triviálně platí.

$k \to k+1$ ... ① $k, k+1$ na stejné hladině ② přes hladinu







- Redukujeme na $Rank_1$ a $Select_1$ v posloupnosti $N = 2n+1$ bitů

$Rank_1(i)$ = počet jedniček na pozicích $1 \dots i$

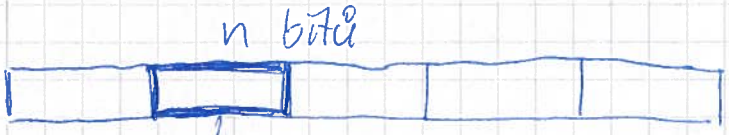
$Select_1(i)$ = pozice i-té jedničky

- Z toho: $Left(i) = 2 \cdot Rank_1(i)$ ——— (vstup i výstup jsou pozice v posl. vrcholů)

$Right(i) = Left(i)+1$

$Parent(i) = Select_1(\lfloor i/2 \rfloor)$

Rank



bloky o $L^2 = \log^2 n$ bitech

podbloky o $\frac{1}{2}L$ bitech

delenpozice...

předpočítáme odpovědi na všechny dotazy

$$\underbrace{2^{\frac{1}{2}L}}_{\text{možných obsahů bloku}} \cdot \underbrace{\frac{1}{2}L}_{\text{možných dotazů}} \cdot \underbrace{LL}_{\text{velikost odpovědi}} \approx \sqrt{n} \cdot L \cdot LL \in o(n)$$

$= \sqrt{n}$

... pro # bloků předpočítáme #1 v předchozích blocích

$$\frac{n}{L^2} \cdot L = \frac{n}{L} \in o(n)$$

(počet bloků · bitů na #1)

pro # podbloků předpočítáme #1 v předch. podblocích téhož bloku

$$\frac{n}{L} \cdot \underbrace{\log(L^2)}_{\approx LL = \log\log n} \in o(n)$$

Select

blok r bitů s $L \cdot LL$ jedničkami

$r \geq (L \cdot LL)^2$

$r < (L \cdot LL)^2$

rozdělíme po $L \cdot LL$ jedničkách a zaznamenáme jejich pozice + pointery na substruktury

$$\frac{n}{L \cdot LL} \cdot 2L = \frac{2n}{LL} \in o(n)$$

(# částí · pozice + pointer)

uložíme pozice všech 1

$$\frac{n}{(L \cdot LL)^2} \cdot L \cdot LL \cdot L = \frac{n}{LL} = o(n)$$

(max. # takových bloků · #1 v bloku · pozice)

podblok r' bitů s LL^2 jedničkami

ještě 1x totéž, tentokrát každá LL^2-tá 1

$r' \geq LL^4$

uložíme pozice všech 1 relativně k bloku

$$\frac{n}{LL^4} \cdot LL^2 \cdot O(LL) \in O\left(\frac{n}{LL}\right) \subseteq o(n)$$

(max. # bloků se (pod-) · #1 v podbloku · pozice)

$r' < LL^4$

$$\frac{n}{LL^2} \cdot O(LL) \in O\left(\frac{n}{LL}\right) \in o(n)$$

(# částí celkem · pozice uvnitř bloku + pointer na substrukturu)

→ pro dost velké n je $r' < \frac{1}{2}L$

předpočítáme pro všechny tvary podbloků

$$\underbrace{2^{\frac{1}{2}L}}_{\text{tvarů}} \cdot \underbrace{LL^2}_{\text{dotazů}} \cdot \underbrace{LL}_{\text{odpovědí}} \in O\left(\frac{n}{LL}\right) \subseteq o(n).$$

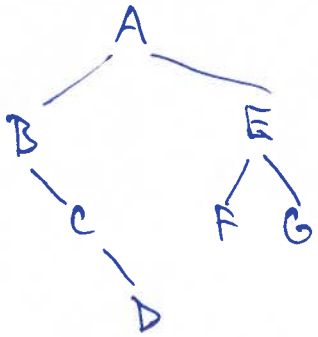
Celkově $o(n)$ bitů navíc (přidali jsme index)

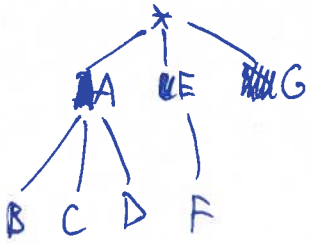
Čas: $\Theta(n)$ předvýpočet

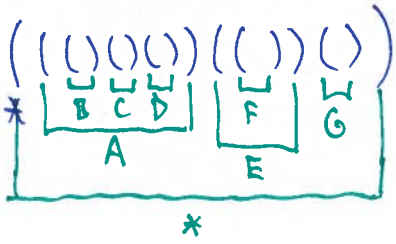
$\Theta(1)$ dotaz

Jiná reprezentace bin. stromů, která umí i počítat velikosti podstromů

bin. strom → pěstovaný strom (kořen, pořadí dětí) → závorkování







n vrcholů — n+1 vrcholů — 2n+2 závorek (bitů)

vrchol →	vrchol →	(....)
levé dítě	první dítě	další znak, je-li to (
pravé dítě	další sourozenec	znak po párové), je-li to (
rodič	předch. sourozenec nebo rodič	předch. znak je) → párová (jinak předch. (
velikost podstromu X	size(X) + size dalších sourozenců X	½ vzdálenosti do obalující)
je list?	je list bez dalších souroz.?	()
# listů v podstromu		rank()) obalující) − rank()) zde

Přidáme indexy pro Match ... párová závorka
a Enclose ... obalující (

Index pro Rank na pozicích bitů generované orákulem

Umí se v O(n) prostoru
O(n) čase na Build
O(1) čase na dotaz

Opět je to delamporice, ale výrazně složitější.