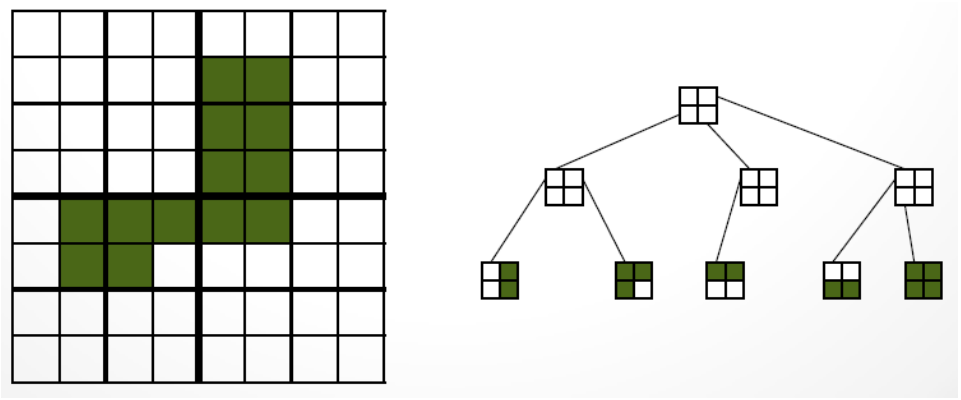
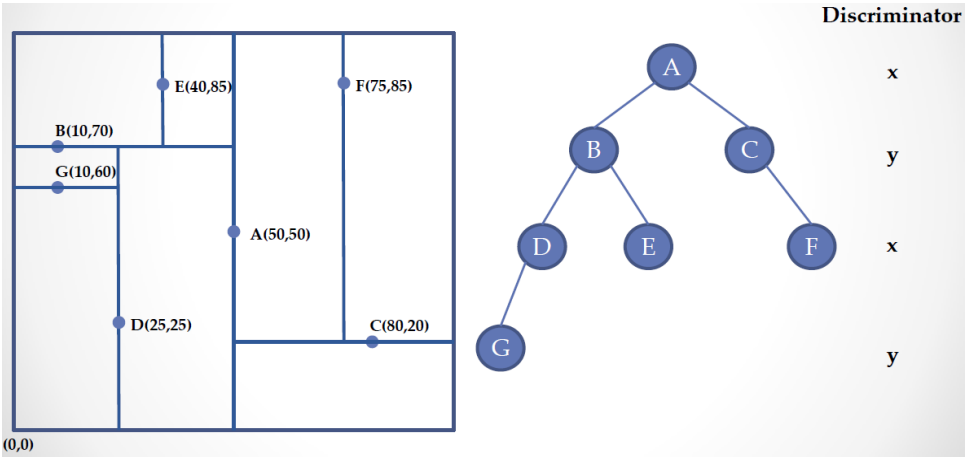
1_prostorové_db

teorie

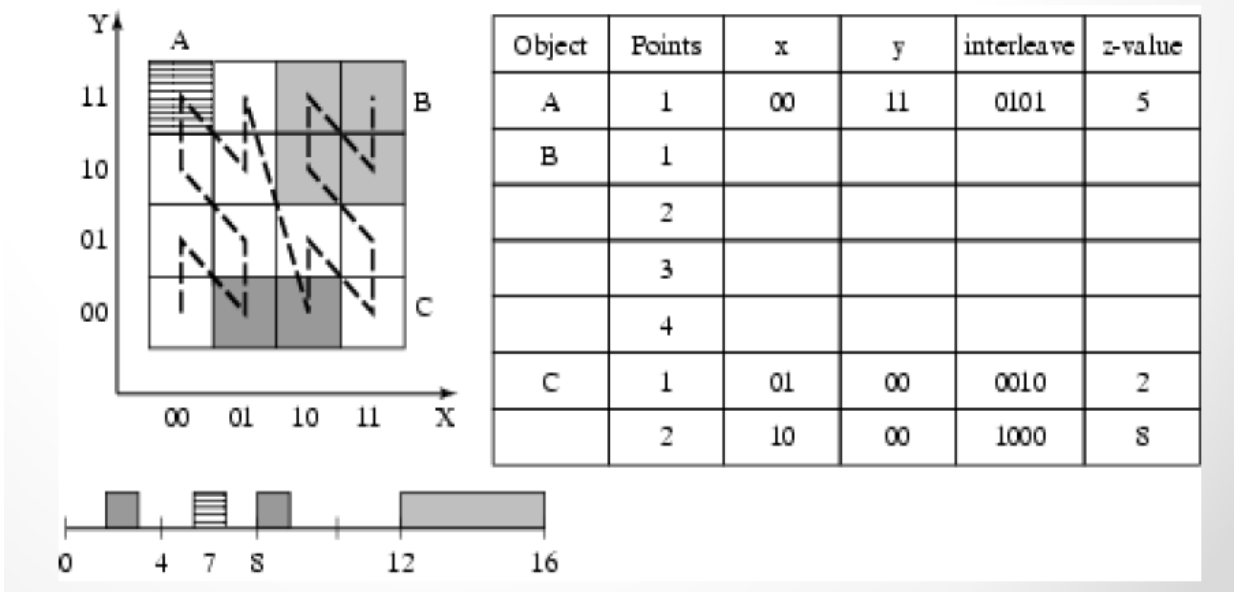
- jak se vyhledává v R-stromech
  - rekurzivně projdeme strom a vrátíme seznam listů (objektů) překrývajících vstupní objekt
- co je quad-tree
  - rekurzivně dělíme prostor do kvadrantů, každý uzel má 1-4 uzly



- k-d-Tree
  - binární strom (uzel: bod, osa podle níž dělíme, 2 synové)



- Buddy Tree
- co je MBR (česky MOO)
  - minimum bounding rectangle
- co je z-usporadani
  - z-hodnota vznikne prokladáním bitů x a y souřadnic



Object	Points	x	y	interleave	z-value
A	1	00	11	0101	5
B	1				
	2				
	3				
	4				
C	1	01	00	0010	2
	2	10	00	1000	8

- R+-Tree
  - jeden objekt se může vyskytovat ve více listech (je tedy duplikován)
  - vnitřní uzly R+-stromu obsahují MOO, které jsou disjunktní.
- jak se liší R*-strom od R-stromu
    - algoritmem na rozdělení přetečené stránky, bere v úvahu více trideni podle os, při přetečení provádí reinsertování, taky algoritmem na hledání listu pro insert (overlap místo nejmenšího rozšíření)

praxe

- štěpení uzlu v R-stromu dle **Guttmana** m=3, M=8

A	A	A			B	B	
	C	C			D	D	D
	C	C			D	D	D
F			E	E			
F		G			I		
	H						
	H						

  - Pick Seeds
    - největší nevyužitá plocha B,H=44
  - ∀Pick Next

	B	H	|Δ|		B	FH	|Δ|		BD	FH	|Δ|		BD	GFH	|Δ|		BD	FGHC	|Δ|
A	5	22	17		5	16	11		20	16	4		20	12	8		20	6	14
C	22	10	12		22	10	12		16	10	6		16	6	10		-	-	-
D	10	40	30		10	40	30		-	-	-		-	-	-		-	-	-
E	18	14	4		18	12	6		13	12	1		13	8	5		13	10	3
F	40	6	34		-	-	-		-	-	-		-	-	-		-	-	-
G	28	4	24		28	4	14		24	4	20		-	-	-		-	-	-
I	10	13	3		10	16	6		6	16	10		6	12	6		6	15	12

	BD	FGHCA	|Δ|		BDI	FGHCA	|Δ|
E	13	16	3		12	16	4
I	6	24	18		-	-	-

- **=> BDIE, FGHCA**

- štěpení uzlu v R-stromu dle **Greenové** m=3, M=8

A	A	A			B	B	
	C	C			D	D	D
	C	C			D	D	D
F			E	E			
F		G			I		
	H						
	H						

    - Pick Axis
        - PickSeeds(z Guttmana):
            - největší nevyužitá plocha B,H=44
        - spočteme normované vnitřní vzdálenosti:
            - y: 5/8
            - x: ⅜
        - vybereme tu s větší normovanou vzdáleností, tedy **y**
    - Half
        - setřídíme objekty podle y souřadnice a rozdělíme (je jedno kam zařadíme E)

A	B	C	D	E	F	G	I	H
7	7	4	4	3	2	2	2	0

        - **=> ABCDE, FGIH**

- štěpení uzlu v **R*-stromu** m=3, M=8

A	A	A			B	B	
	C	C			D	D	D
	C	C			D	D	D
F			E	E			
F		G			I		
	H						
	H						

  - Pick Axis
    - seradime podle osy **y** a počítáme h-okraje

A	B	C	D	E	F	G	I	H

h-okraje:

(7+4)*2=	22	24	26	28
	28	20	20	18
Celkem	**50**	**44**	**46**	**46**

- Celkem: 184

    - seradime podle osy **x** a počítáme h-okraje

F	A	H	C	G	E	I	B	D
0	0	1	1	2	3	5	5	5

h-okraje:

		22	22	22	26
		24	24	20	18
∑		**46**	**46**	**42**	**44**

- Celkem: 182

    - vybereme tu s menší hodnotou, tedy **x**

- Split
  - počítáme h-překryv když jsou 2 nejmenší stejné tak i h-objem

F	A	H	C	G	E	I	B	D

h-překryv:

		12	6	0	0

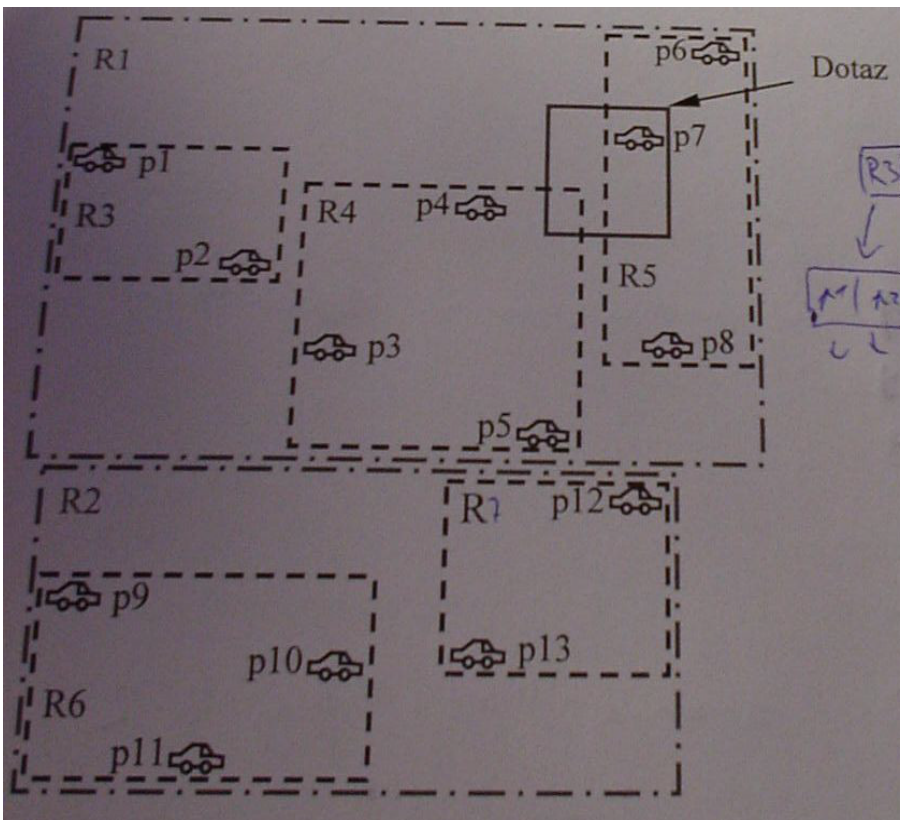
h-objem:

				24	40
				30	18
				54	**58**

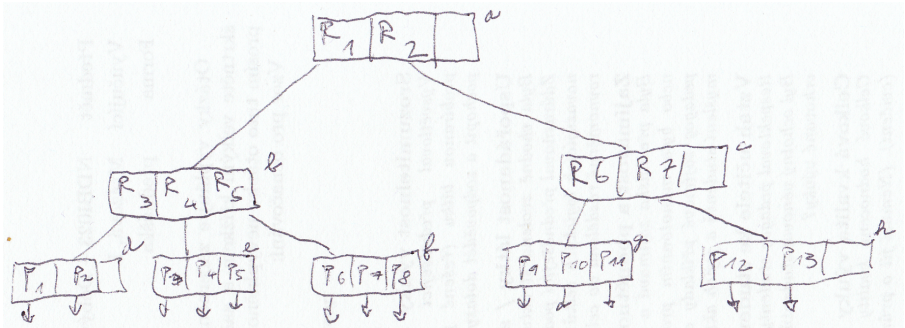
- zase vybereme nejmenší a podle nich rozdělíme
- **=> FAHCG, EIBD**

- další příklady:
  - http://www.ksi.mff.cuni.cz/~zemlicka/pdf/StepeniPreplnenehoUzlu.pdf

- uvažujte objekty p1-p13 umístěné do MOO na obrázku



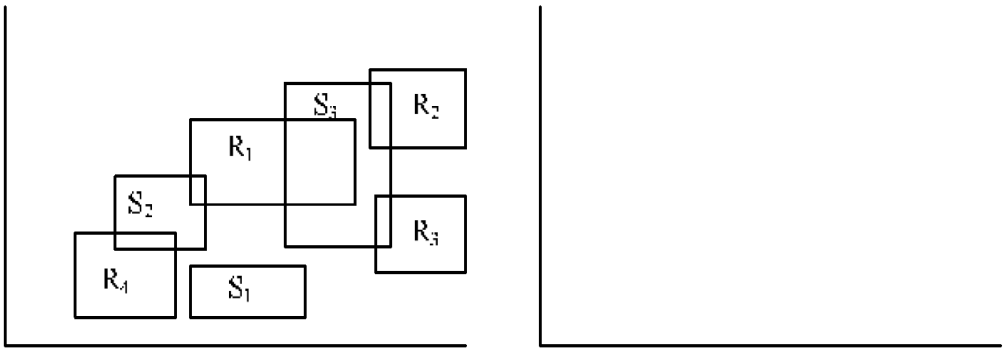
  - zkonstruujte vedle obrázku odpovídající 3-ární R-strom tak, že v jeho listové úrovni jsou ukazatele na n-tice obsahující informace o daných objektech



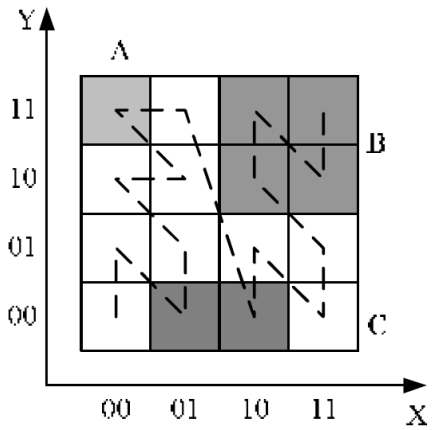
  - kolik stránek R-Stromu musíme přečíst, abychom se dozvěděli vše o objektech v okně dotazu?
    - stránky a,b,e,f a data p7 pokud v listových uzlech jsou MOO objektů, pokud ne, tak i data p3, p4, p5, p6, p8

- co se musí udělat v R-Stromu, chceme-li umístit 1cm pod objekt p13 další objekt p14
  - najdeme vhodný listový uzel, kam vložíme p14
    - asi to bude R7, do něj se vejde (nemusí se štěpit)
    - pak opravíme R7, R2 asi zvětšovat nebude třeba

- Na obrázku jsou prostorové objekty (reprezentované MOO) ze dvou souborů R a S. Každý objekt je reprezentován levým dolním a pravým horním rohem (xL, yL, xR, yR). Zametací algoritmus realizuje prostorové spojení souborů R a S založené na neprázdném ∩ prvků z R a S. Jeho předpokladem je uspořádání prvků R∪S podle souřadnice xL. Nechť Ri je prvek z R∪S s nejmenší souřadnicí xL.
  - Formulujte podmínku, která umožňuje testovat pouze omezené množství prvků z S na spojení, následné odstranění Ri z R∪S a pokračování algoritmu s menším počtem prvků.
    - alg. sweep
  - Naznačte rozmístění prvků z R∪S vpravo od obrázku tak, aby nastal nejhorší případ, tj. testovalo se |R * |S| prvků.
    - zarovnat nad sebe (aby začínali stejně)



- Na obrázku vlevo jsou 3 prostorové objekty A, B a C jejichž body jsou uspořádány pomocí Peanovy křivky. V tabulce vpravo jsou odpovídající z-hodnoty pro A a C.
  - S počítejte z-hodnoty pro objekt B.

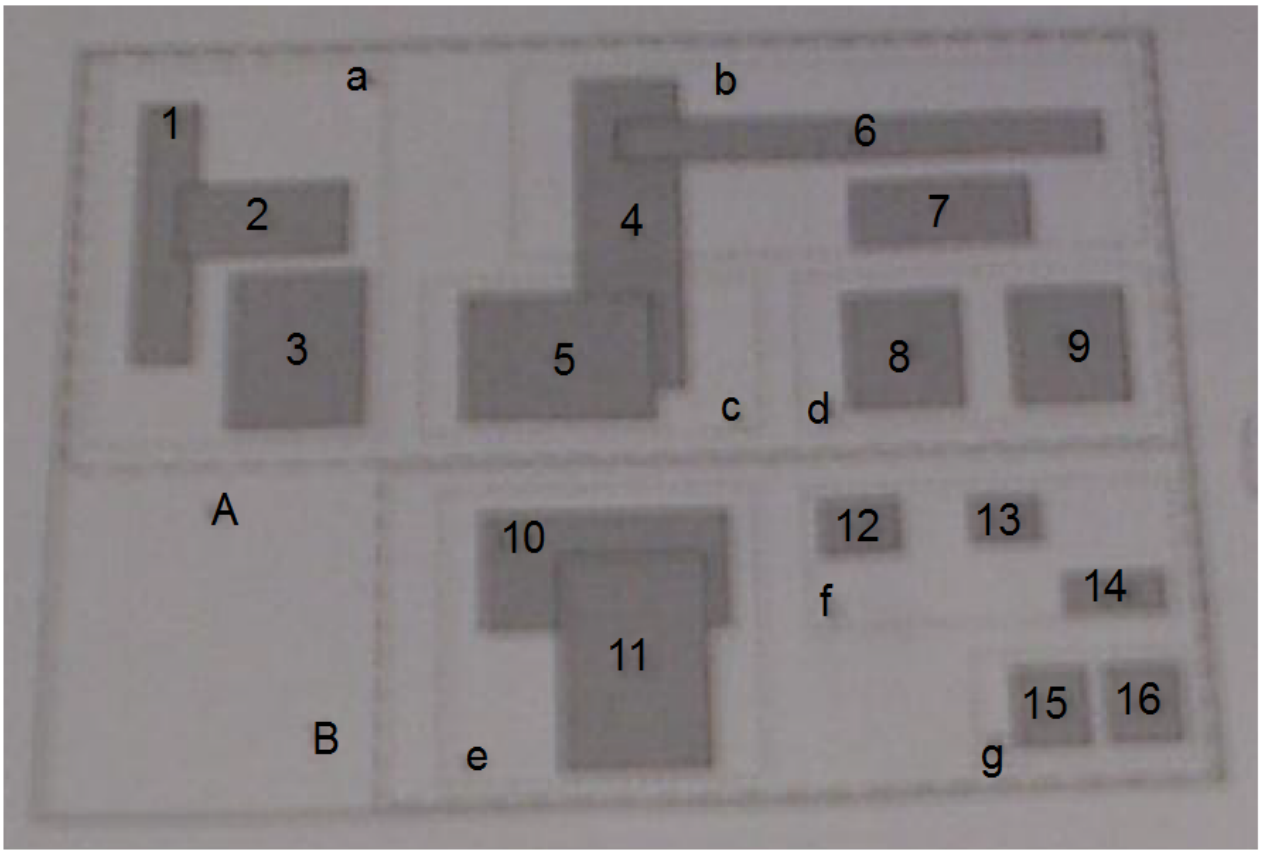


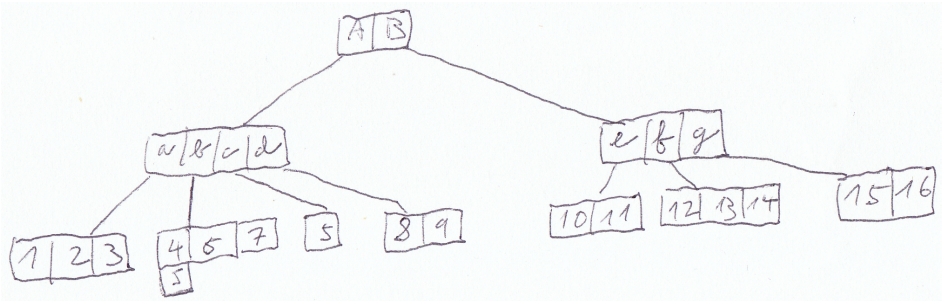
objekt	x	y	kod	z-value
A	00	11	0101	5
B	**10**	**10**	**1100**	**12**
	10	**11**	**1101**	**13**
	11	**10**	**1110**	**14**
	11	**11**	**1111**	**15**
C	01	00	0010	2
	10	00	1000	8

  - Kde vidíte nevýhodu v uspořádání bodů reprezentujících objekty A, B, C?

        - C dostalo adresy které nejsou blízko (narozdíl od B)

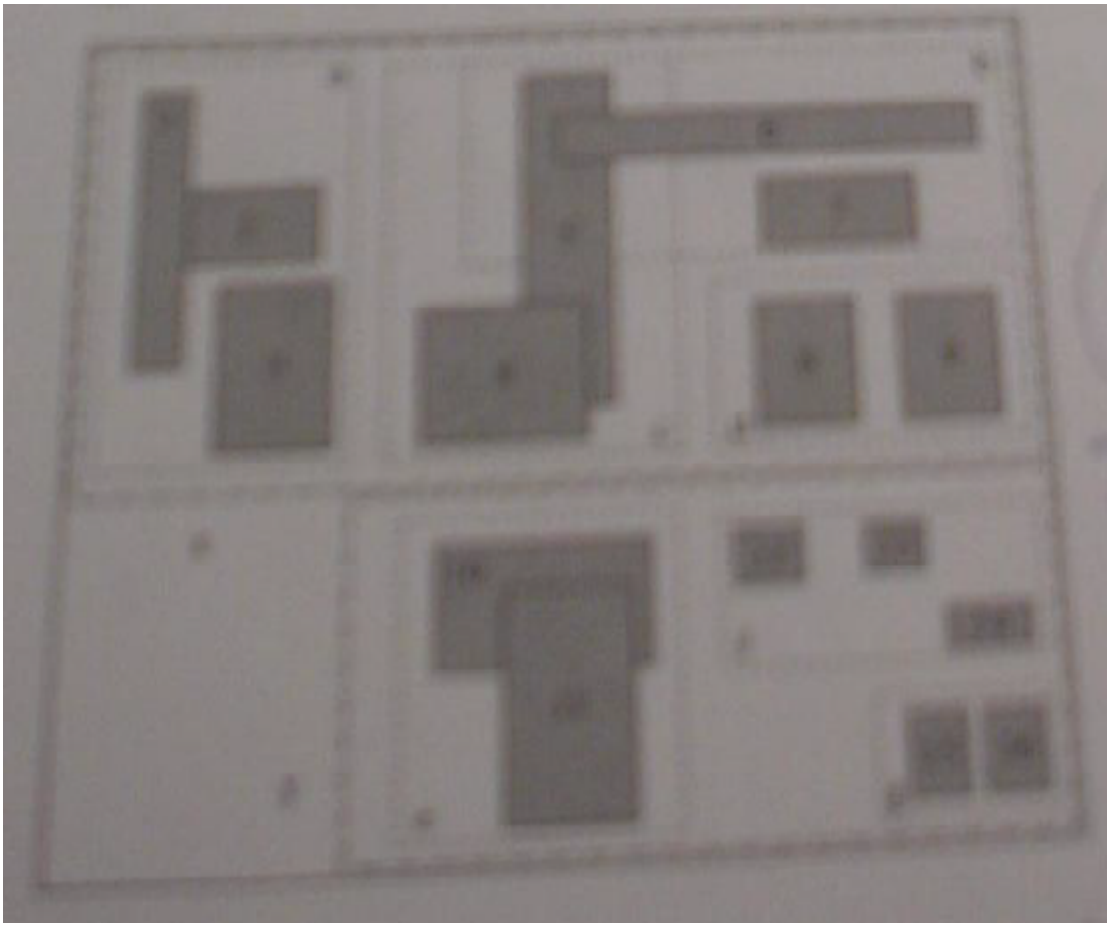
- R+-strom se liší od R-stromů tím, že
  - jeden se může vyskytovat ve více listech (je tedy duplikován)
  - vnitřní uzly R+-stromu obsahují MOO, které jsou disjunktní.
    - Uvažujte množinu 16 objektů na obrázku. Tyto objekty jsou rozděleny do skupin a pokryty MMO.
      - Zkonstruujte odpovídající R+-strom (označuj MOO identifikátory z obrázku).
        - **(POZOR v obrazku je chyba, prohoďte čísla 4 a 5)**





        - Kolik operací READ bude vyžadovat nalezení objektu 5?
          - 5+2 (5)

- Uvažujte množinu 16 objektů na obrázku. Tyto objekty jsou rozděleny do skupin a pokryty MMO.
  - Zkonstruujte odpovídající R-strom (označuj MOO identifikátory z obrázku).
    - (čísla objektů jako nahoře)



- Kolik operací READ bude vyžadovat nalezení objektu 5?
  - 3+1 (5)

2_hledani_v_textech

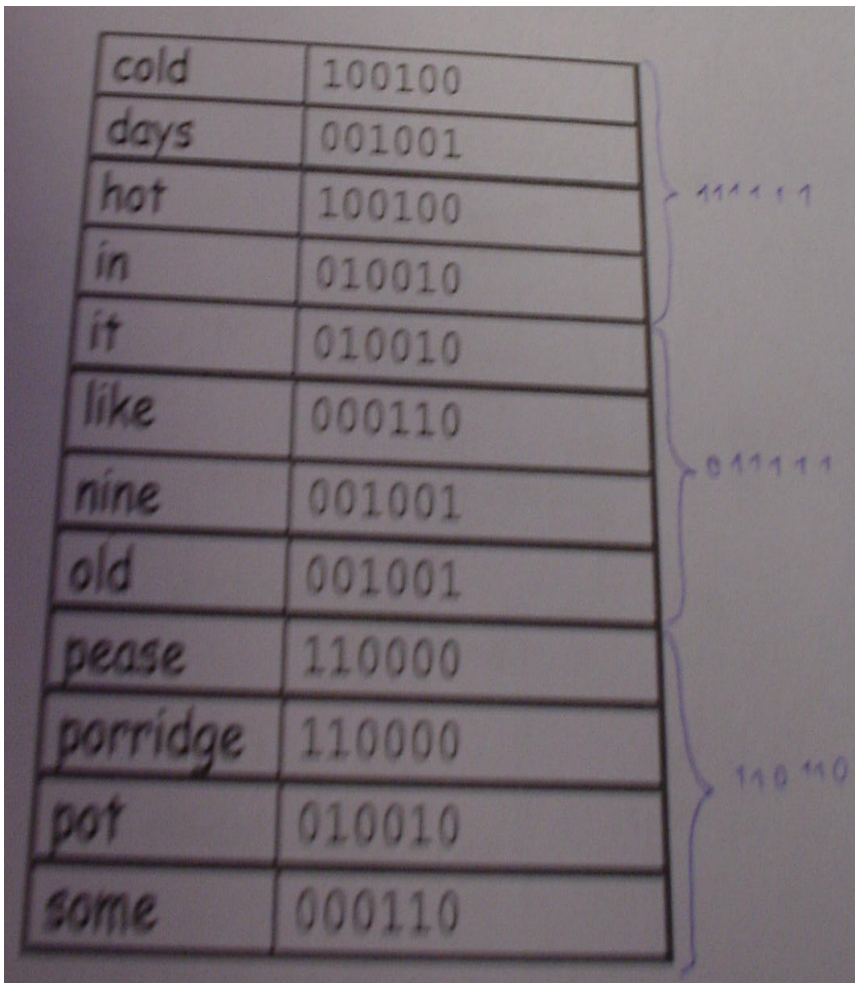
teorie

- co je to lemmatizace
    - převedení slova do základního tvaru (např. pro hledání fulltextem)

praxe

- Počet 1 v signatuře termu: $k = \log_2 1/Pf = -\log_2 Pf$
- Doporučený počet bitů signatury: $d = 1 / \ln 2 \; n \log_2 1/Pf$
    - Pf je pst chybného výběru a n je počet termů v dokumentu

- Uvažujte termy získané analýzou krátkých textových zpráv čtených sekvenčně za sebou. Pro každou zprávu jsou tyto termy čtyři. Vygenerované bitové řetězce slouží jako zdroj pro vytvoření blokových signatur dvouvrstvým vrstveným kódováním.
    - Zkonstruujte vedle obrázku tyto signatury. Přestavte si u nich ukazatel, který určí odpovídající textovou zprávu umístěnou v nějaké stránce na disku.
        - zORujeme je po 4řech, obrazek je podle me dobre



    - Uvažujte další term *father* kódovaný jako 010001 a *new* kódovaný jako 1000...(zbytek textu se z písemky nedochoval).
    - Kolik bude přečteno zpráv pro dotazy a kolik bude chybných výběrů? Zapište do tabulky:
        - zpráva = bloková signatura:

term	**hledací signatura (OR)**	**# přečtených zpráv**	**# chybných zpráv**
father	010001	2blokové signatury	2blokové signatury
old	001001	2blokové	1blokové
days **and** in	days or in = 011011	2blokové	1blokové

some **and** days	some or days=001111	2blokové	2blokové

- odkaz co jsem nasel :)
    - http://books.google.cz/books?id=2F74jyPl48EC&pg=PA140&lpg=PA140&dq=cold+%22100100%22+days+%22001001%22&source=bl&ots=5QfTHr9X3c&sig=1b8_N_bttmsT6y3dHjpG17lKYmI&hl=cs&sa=X&ei=k5qPUfnNLtKQ4AS_ioCQDQ&ved=0CDAQ6AEwAA#v=onepage&q=cold%20%22100100%22%20days%20%22001001%22&f=false

3_komprese

teorie

- jak je definována redundance kódu
- jaký je vztah mezi očekávanou délkou BSTW a Huffmanova kódování
    - Huffmann by častěji měl být lepší

praxe

Fibonacci kódování

- zakodujte F2(100), F3(10), F2(112), F3(112), F4(112)

F2	1	1	2	3	5	8	13	21	31	55	89	oddělovač
F2(100)	X											1
F2(112)	X											1

F3	1	1	1	3	5	9	17	31	57	105		oddělovač
F3(10)	X											110
F3(112)	X											110

F4	1	1	1	1	4	7	17	31	57	105		oddělovač
F4(112)	X											1110

    - prvni 1 se nepise, oddelovac je (1^(k-1) 0) mimo F2
- precist posloupnost bitu jako Fibbonacciho kod radu 2 a take jako kod gamma.
- zadaný binární číslo a mělo se převíst pomocí Fib3 a delta nazpět (tu deltu nazpět jsme asi nedělali, ale dalo se to zpětně vymyslet).

Eliášovy: alfa, beta(‘), gamma(‘), delta, omega kódování

- zakóduj: gamma’(18), gamma(18), gamma’(112), gamma(112), delta(112), omega(12), omega(112), omega(170), omega(100)
    - gamma’(18) = alfa(beta(18)) beta**’**(18)

      16 8 4 2 1
        - beta(18) = “binární 18” = 1 0 0 1 0
        - beta’(18) = “binární 18 bez první 1” = 0010
        - alfa(10010) = “unární 10010” = 00001
        - gamma’(18) = 000010010
    - gamma(18) = “prokládání alfa(beta(18)) a beta**’**(18)” = 000001001
    - gamma’(112), gamma(112)

      64 32 16 8 4 2 1
        - beta(112) = 1 1 1 0 0 0 0
        - alfa(1110000) = 0000001
        - gamma’(112) = 0000001110000
        - gamma(112) = 0101000000001
    - delta(112) = gamma(|beta(112)|) beta’(112)
        - gamma(|1110000|) = gamma(7)
            - beta(7) = 111
            - gamma(7) = 01011
        - delta(112) = 01011110000

- omega(18) = { k = 0; while($\lfloor \log_2 N \rfloor$ > 0) { k = beta(n)k; N = $\lfloor \log_2 N \rfloor$ } }
  - 0 => 10010 (N=4) =>100 (N=2) => 10 (N=1)
  - omega(18) =10 100 10010 0
- omega(112)
  - 0 => 1110000 (N=|beta(112)|-1=6) =>110 (N=|beta(6)|-1=2) => 10 (N=1)
  - omega(18) =10 110 1110000 0
- omega(170)

  128 64 32 16 8 4 2 1
  - beta(170) = 1 0 1 0 1 0 1 0
  - 0 => 10101010 (N=7) =>111 (N=2) => 10 (N=1)
  - omega(18) =10 111 10101010 0
- omega(100)
  - 0 => 1100100 (N=6) =>110 (N=2) => 10 (N=1)
  - omega(100) =10 110 1100100 0

Shannon-Fano kódování

- SF(EMA MELE MASO A MELE U TOHO A MELE A MELE A MELE)
  - seřadíme podle vyskytu
  - pulime seznam (přibližně stejně rozdělíme výskyty) a prirazujeme bity dokud nejsou jednoznacne urceny vsechny symboly

1	1	1	1	3	4	6	7	11	12
H	S	T	U	O	L	A	M	E	_
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1		
0	0	1	1	0	1				
0	1	0	1						

- vysledek:

" "	E	M	A	L	O	U	T	S	H
11	10	001	010	0011	0010	00011	00010	00001	00000

- SF(EMA MELE MASO)

1	1	1	2	2	3	3
L	O	S	_	A	E	M
0	0	0	0	0	1	1
0	0	0	1	1	0	1
0	0	1	0	1		
0	1					

  - výsledek má EMA(2+2+3b)+mezera(3)+MELE(10b)+mezera(3)+MASO(12) = 35bitů

- SF(prší prší)

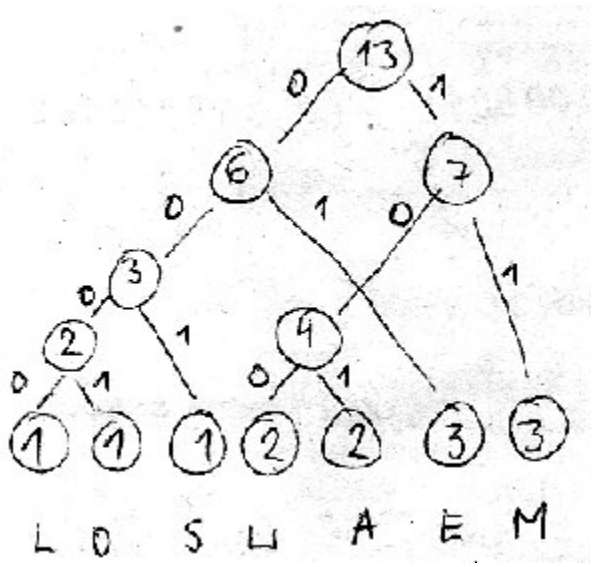
1	2	2	2
_	š	r	p
0	0	1	1
0	1	0	1

Huffmanův kód

- Huffman(EMA MELE MASO)
  - seřadíme podle vyskytu

1	1	1	2	2	3	3
L	O	S	_	A	E	M

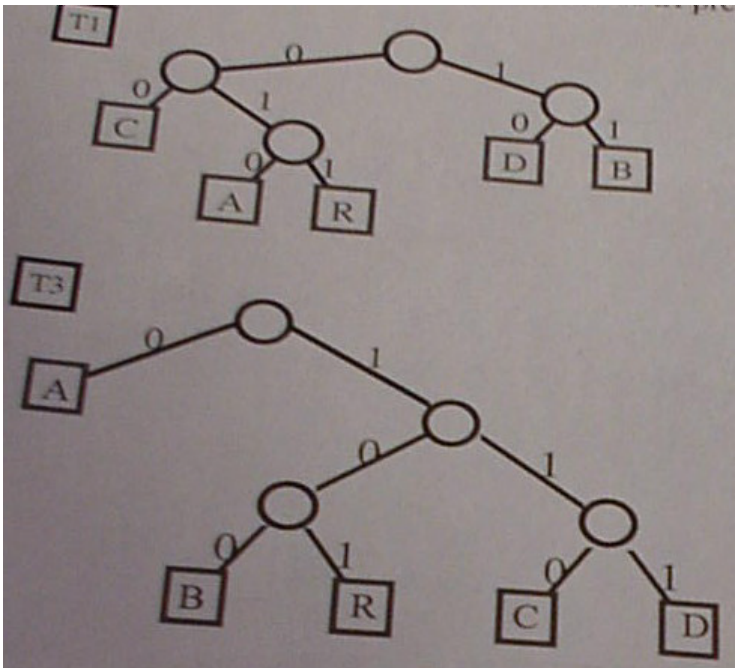
- spojováním dvojic zleva vytváříme strom. Nové vrcholy zařazujeme do řady (co nejvíce doprava)

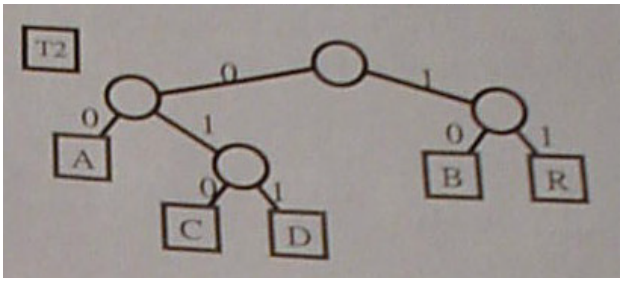


  - výsledek má EMA (2+2+3) + (3) + MELE (2+2+4+2) + (3) + MASO (2+3+3+4) = 35bitů

M	E	A	_	S	O	L
11	01	101	100	001	0001	0000

- uvažujte text ABRACADABRA a tři prefixové kódy pro abecedu {A,B,C,D,R}





  - který z kódů je Hufmanův a proč
    - T3 - sourozenecká vlastnost - jsou serazeni podle vyskytu

1	1	2	2	5
D	C	R	B	A

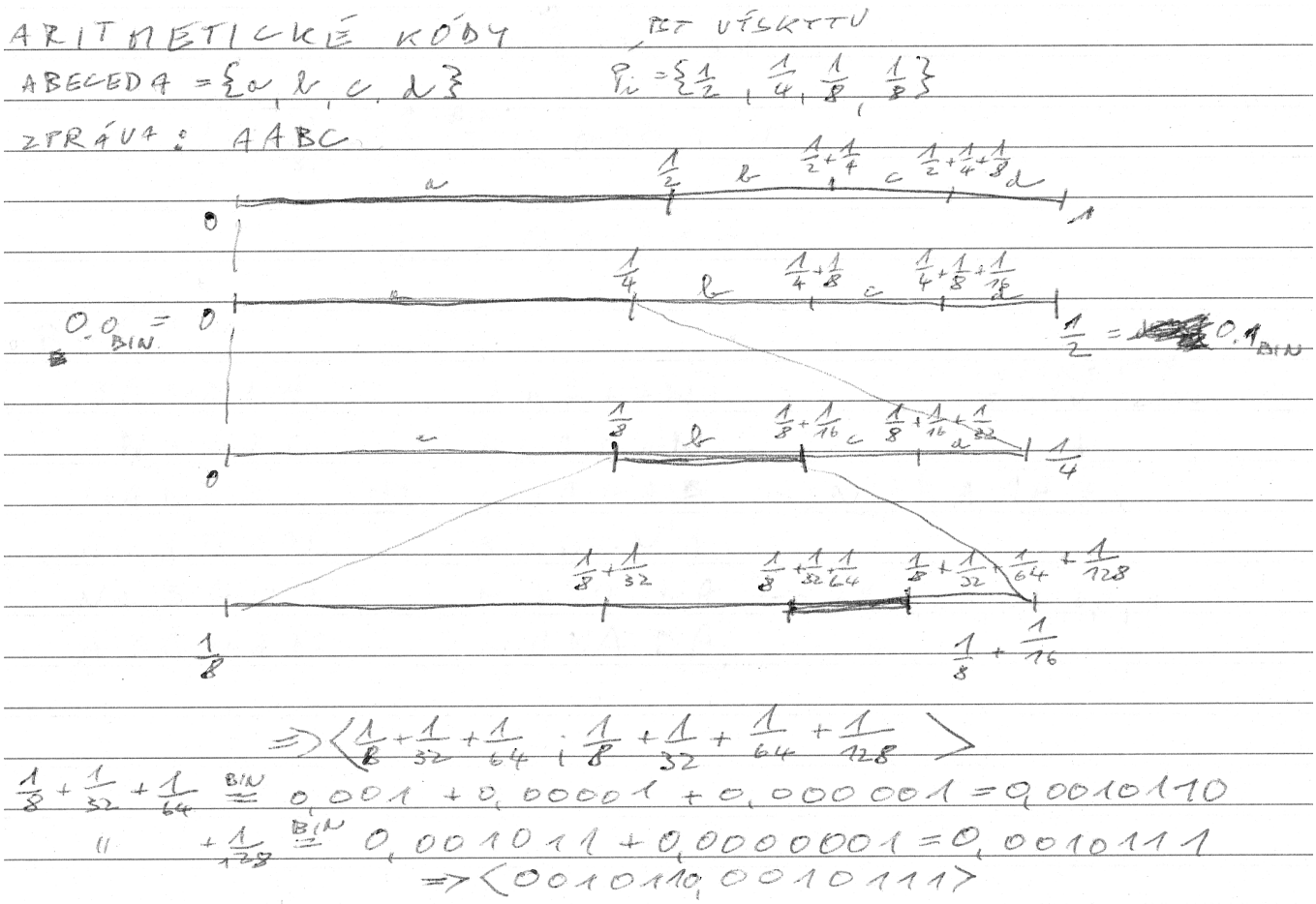
  - spočtěte pro něj očekávanou délku AL
    - AL(K) = ∑(delka_kodu_znaku * pr-st vyskytu znaku) = 3*1/11 + 3*1/11 + 3*2/11 + 3*2/11 + 1*5/11 = #bitů/#znaků = 23/11 = 2,09
  - který z daných kódů je pro kódování daného textu nejhorší a proč
    - Jde o to spočítat délku (# bitů) pro zakódování slova "ABRACADABRA", což se počítá následovně:
      - $\sum_{PresZnaky}$(#vyskytuZnaku * #bituProZnak)

znaky	D	C	R	B	A	**#bitů celkem**
#výskytů	1	1	2	2	5	---
#bitů T1	2b	2b	3b	2b	3b	2+2+6+4+15=29b
#bitů T2	3b	3b	2b	2b	2b	3+3+4+4+10=24b
#bitů T3	3b	3b	3b	3b	1b	3+3+6+6+5=23b

    - tedy T1 je nejhorší

Aritmetické kódování

- aritmeticky zakodovat "JUJ!"
- Aritmetický kodování - naštěstí poměr písmen 1:1 (myslím OJOJ).
- příklad ze cvičení:



BW (Burrows-Wheelerova transformace)

- pozor oblibena otazka (!)
- Dostali jste slovník a kod, pak pomocí BW sestavit slovo (byl to nějakej nesmysl). Z toho nesmyslu se měla sestavit celá tabulka pro BW (bloky). Z tabulky pak vypadlo slovo KOLOTOC (to sem si ale jen domyslel, úkolem to nebylo).
- moje zapisky ze cviceni za moc nestoji, takže příklad z wikipedie:
    - http://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform

BSTW

- BSTW(EMA MELE MASO A MELE A MELE) - ignorujte mezery
  - střídá se kodovani slov a neslov, použité slovo se přemístí na začátek slovníku, odkazujeme se do slovníku
  - kodujeme:
    - 1 EMA
    - 2 MELE
    - 3 MASO
    - 4 A
    - 3
    - 2
    - 2
- zakódujte pomocí BSTW kodovanim zpravu: “a chalupu a chalupu dostane od táty od táty” (mezery ignorujte)
  - kodujeme:
    - 1 a
    - 2 chalupu
    - 2
    - 2
    - 3 dostane
    - 4 od
    - 5 táty
    - 2
    - 2

LZ77, (LZSS)

- LZ77(EMA MELE MASO) parametry dány 7 a 3 bity, kolik bitů má 1 triplet?
  - jak daleko vidíme zpět = 7bitů, kolik vidíme = 3bity
  - kódujeme tripety <jak daleko zpět, kolik vzít, vkládaný znak>
  - <0,0,E> <0,0,M> <0,0,A> <0,0,_> <3,1,E> <0,0,L> <2,1,_> <5,1,A> <0,0,S> <0,0,O>
  - 1triplet = $\lceil \log_2 7 \rceil + \lceil \log_2 3 \rceil$ + 8 bitů na znak = 3 + 2 + 8 = 13

- Posloupnost několika set 1 a 0 a tabulka znaků - z toho měl člověk pomocí LZ77 (parametry dány 7 a 3 bity) sestavit zprávu - nějaký prší prší. Na zkoušce zpráva začínala: 0000001010000000000010100100000...
  Znaky byly A = 01000001, B = A+1 atd. až do Z.
  - na indexování vyhledávací části dlouhé 7 bitů potřebuju 3 bity, na 3 bitový výhled pak 2 bity a 8 bitů na znak, takže triplety jsou vždy 3+2+8 bitů, ten první např. <000,00,01010000>.

- LZSS(ABLABLABLAM!) parametry dány 12 a 4 bity
  - variace na LZ77, posleme bud znak nebo dvojici
  - <A> <B> <L> <3,4> <3,3> <M> <!>
  - 8+1 bitů = znak a 4+2+1 bit = dvojice

- LZSS(EMA MELE MASO) (promyslet dobre, kdy se vyplati kodovat kolik znaku : dnes bylo okno 4b + 3b na delku shody, takze jeden znak se kodovat vyplatil)
  - <E> <M> <A> <_> <2,1> <E> <L> <2,1> <_> <M> <A> <S> <O>

LZ78, LZW

- LZ78(EMA MELE MASO)
  - pri kodovani si stavime slovnik postupne
  - <0,E> <0,M> <0,A> <0,_> <2,E> <0,L> <1,_> <2,A> <0,S> <0,O>
  - slovník:
    1. E
    2. M
    3. A
    4. _
    5. ME
    6. L
    7. E_
    8. MA
    9. S
    10. O
- Zakódujte pomocí LZ78 algoritmu zprávu “000000000000100000000000”. Pro názornost zapisujte pozice přirozenými čísly v dekadické soustavě.
  - 0 00 000 0000 001 00000 000000
  - <0,0> <1,0> <2,0> <3,0> <2,1> <4,0> <6,0>
  - slovník:
    1. 0
    2. 00
    3. 000
    4. 0000
    5. 001
    6. 00000
- LZW(EMA MELE MASO)
  - prvotní slovník, znaky (0..255) začíná 65:
    65. A
    66. B
    67. C
    68. D
    69. E
    70. F
    71. G
    72. H
    73. I
    74. J
    75. K
    76. L
    77. M
  - kódujeme:
    - E M A _ M E L E _M A S O EOD
    - 69 77 65 32 77 69 76 69 260 65 83 79 256
  - rozšiřujeme slovník (257...)
    257. EM
    258. MA
    259. A_
    260. _M
    261. ME
    262. EL
    263. LE
    264. E_
    265. _MA

266. AS
267. SO

Různé

- 6. Uvažujte kód C3={1, 101}
    - Je C3 prefixový?
        i. Moje odpověď (Patrik): není - Prefixový je, pokud jednotlivé kódy nejsou prefixem ostatních kódů. V našem případě je “1” prefixem “101”.
    - Je C3 jednoznačně dekódovatelný?
        i. Moje odpověď (Patrik): je - Pokud přečtu 1, za kterou následuje 0, jedná se o 2. kód. pokud přečtu jedničku, za kterou následuje 1, jedná se o první kód. Ať mám tedy za sebou libovolnou sekvenci 0 a 1 složenou z těchto 2 kódů, tak dokážu jednozncně dekompresovat.
- 7. Uvažujte kód C3={0, 10, 110, 111}
    - Je C3 prefixový?
        i. Moje odpověď (Patrik): je
- 8. Uvažujte kód C3={0, 01, 011, 111}
    - Je tento kód jednoznačně dekódovatelný? Zdůvodněte odpověď.

Ostatní (necvičily se)

Fázování

- potrebujem dobru dusu, ktora by mi vysvetlila Fazovanie. Na cvikach sa to pocitalo, ale vobec mi nesedia vysledky. Podla definicie som to skusal pre n=10, vyslo mi to:

  0 - 000
  1 - 001
  2 - 0100
  3 - 0101
  4 - 0110
  5 - 0111
  6 - 1000
  7 - 1001
  8 - 1010
  9 - 1011

  Takze ked chceme rozkodovat postupnost 0010100001, tak mi to vyjde 1,2,1...

  Pokud si dobre pamatuji tak maximalni mocnina 2 je 8 takze mi zbydou dva posledni kody, ktere rozdvojim - 110

  a 111.
  Pak tabulka vypada spis takhle

  0 - 000
  1 - 001
  2 - 010
  3 - 011
  4 - 100
  5 - 101
  6 - 1100
  7 - 1101
  8 - 1110
  9 - 1111

Intervalové kódování

- intervalove kodovani

Team řešitelů přeje štěstí u zkoušky a u státnic:)