Teorie Informace 2. test

1. Detekční a korekční kódy (bezpečnostní kódy)

=kódy, které jsou schopny zjišťovat/detekovat popřípadě opravovat/korekční chyby způsobené přenosem informace sdělovacím kanálem

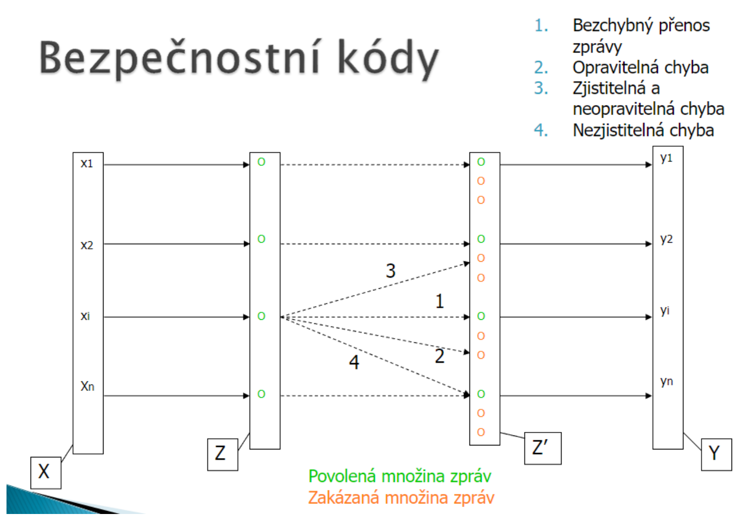
-využíváme k tomu **rozdělení množiny zpráv** na dvě skupiny:

- Povolená množina zpráv
- Zakázaná množina zpráv

-korekční kódy pomyslně obětovávají část své detekční schopnosti za schopnost přímo určit konkrétní chybu a napravit ji
-dále je u některých kódů využívaná samotná vlastnost chybovosti přenosu a to, že chyby vznikají v clusterech (chumel) vedle sebe, takže se optimalizují pro detekci/korekci chyb, jenž následují za sebou

2. Typy přenosu

- Bezchybný přenos zpráv (př. Impuls 5V-dopředu domluvený binární kód +-5V)
- Opravitelná chyba (př. Impulz v rozmezí 4,5-5,5V)
- Zjistitelná a neopravitelná chyba (př. Impulz o hodnotě 1V)
- Nezjistitelná chyba (př. Vyšleme Impuls +5V a dojde -5V)



3. Váha kódového slova W(Zi)

-u rovnoměrného binárního kódu je to počet 1 v kodovém slově
-je to ukazatel, pomocí kterého vyjadřujeme Hammingovu vzdálenost

4. Hammingova vzdálenost

-je to počet bitů/symbolů v nichž se 2 slova liší
-používá sčítání XORu 2 slov
d(zi,zj)=d(1100,1111)=2
-vlastnosti:

- d(zi,zj)>=0
- zj = zi právě když d(zj, zi)=0
- d(zi, zj)+ d(zj, zk)>=d(zi, zk)
    -trojúhelníková nerovnost a+b>=c
-d(zi, zj)=d(zj,zi)

5. Minimální Hammingova vzdálenost a její význam

- Je to nejmenší počet znaků, ve kterém se liší množina kódových slov->porovnám Každé s každým a nejmenší hammingova vzdálenost je minimální Hammingova Vzdálenost
-dmin(z)=min (zi, zj) přes všechna (i!=j)
-význam:

-s->zjistitelná chyba; t->opravitelná chyba
-dmin(z)>s pak je kód schopen zjistit libovolnou s-násobnou chybu
-dmin(z)>2t pak je kód schopen opravit libovolnou t násobnou chybu
-dmin(z)>t+s, t<=s pak kód je schopen opravit libovolnou **t** násobnou chybu nebo Menší a současně detekovat libovolnou **s** násobnou chybu

6. lineární kódy

-systematický kód:

- n->počet symbolů v kódovem slově
- k->počet informačních symbolů
- n-k->počet zabezpečovacích symbolů
-jestliže prvních k symbolů je informačních a zbývajících n-k je zabezpečujících jedná se o **systematický kód(n,k)**

-**nesystematický** = informační a zabezpečující bity jsou proházené

-vnitřní vztahy mezi symboly uvnitř kodoveho slova jsou u linearnich kodu dany Soustavou n-k lineárních nezávislých rovnic

7. generátorová matice

=slouží ke generování kódových slov

-v případě kódu (7,4) musíme pro vytvoření generátorové matice vyjádřit pomocí prvních 4 neznámých zbylé 3 proměnné

-generátorovu matici vytvoříme doplněním **jednotkové matice k x k** (4x4) vyjádřenými proměnnými zapsanými do sloupců, kde jednotlivé řádky odpovídají známým proměnným

-musí platit **$H \times G^T = 0$** (pak můžeme vytvářet kodér a dekodér, který bude fungovat)

8. kontrolní matice

-u LK musí platit **$H \times \alpha^T = 0^T$** kde:

-H = kontrolní matice z níž dostáváme soustavu rovnic

-α^T = transponovaný vektor kódového slova

9. Syndrom

-výsledek rovnice **$H \times \alpha^T = s^T$** označujeme jako syndrom kódového slova α

-**nulový** syndrom = kódové slovo α **patří** do množiny povolených slov

-**nenulový** syndrom = kódové slovo α **nepatří** do množiny povolených slov, syndrom odpovídá sloupci kontrolní matice H, který odpovídá bitu, ve kterémnastala při přenosu **chyba**

10. Hammingův kód $(n, k) = (2^{(n-k)}-1\ ,\ 2^{(n-k)} - (n-k) - 1)$

-příklady Hammingova kódu (3,1) ; (7,4) ; (15,11) ; (31,26) ; (63,57) ; (127,120)

-informační poměr se rychle blíží 1 (4/7 = 0,571   57/63 = 0,905), ale jelikož dokážeme ale jelikož dokážeme kvůli d_{min} =3 detekovat pouze dvojnásobnou chybu, jsou výhodnější kratší kódy

-d_{min}=3 => detekuje dvojnásobnou nebo opravuje jednonásobnou chybu (mají nejmenší možnou redundanci)

11. rozšířený Hammingův kód

-kontrolní matici rozšířeného HK (n+1,k) získáme **přidáním řádku** jedniček nahoru + doplněním ostatních řádků na **sudou paritu** jedniček ke kontrolní matici HK (n,k)

1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 0
0 1 1 0 0 1 1 0
1 0 1 0 1 0 1 0

-d_{min}=4 => detekuje trojnásobnou chybu nebo opravuje jednonásobnou a detekuje dvojnásobnou chybu

12. lineární cyklické kódy

- jsou podtřídou lineárních kódů
-všechny vlastnosti stejné jako lineární kódy
-z jednoho povoleného kodoveho slova vytvoříme nové cyklických posunem + samé nuly!
-pro zápis CK se používá polynom stupně n-1

13. generující polynom

-slouží ke kódování pomocí **vynásobení s informačními bity** (**a(x) = g(x) * i(x)**)

-dekódování funguje na principu **vydělení zprávy generujícím polynomem** (**b(x) / g(x) = m(x) a zbytek r(x)**), kde zbytek r(x) určuje, zda došlo při přenosu k chybě, nebo ne, popřípadě k jaké

Vlastnosti
-reprezentuje povolené kódové slovo polynomu nejnižšího stupně
-bezezbytku dělí polynom **x^n+1**

-pro CRC16 g(x) = 0x1021 = 10001000000100001b = x16 + x12 + x5 + 1

14. systematický cyklický kód $i(x) * x^{n-k} + r(x) = g(x) * m(x)$

Postup tvorby kódového slova
1. informační bity vynásobíme x^{n-k} (stačí přidat tolik nul, jaký je nejvyšší stupeň v g(x))
2. informační bity podělíme g(x), pro získání zbytku r(x)
3. výsledek kroku 1 převedeme do formy polynomu a přičteme zbytek r(x), čímž získáme polynomiální formu povoleného kódové slovo a(x), kde **prvních k bitů** je **informační část** a **zbytek zabezpečující část**

15. detekční schopnosti CK

-každá jednoduchá chyba má svůj zbytek po dělení => lze **opravit**
-každý shluk dvojnásobných chyb má svůj zbytek po dělení => lze **opravit**

-cyklický kód (n,k) detekuje každý shluk chyb délky n-k
-závisí to tedy na poměru informačních a všech bitů, přičemž to opět ovlivní to, zda chceme chyby pouze detekovat nebo i opravit
-lepší vlastnosti pro detekcí shuku chyb(přirozených) než náhodných

16. CRC

-díky své ohromné detekční schopnosti se CK používají jako CRC pro opravování **neúmyslných** chyb vzniklých při přenosu

-vůči **úmyslným** chybám CRC **nelze** použít z důvodu možnosti ošálení úpravou nepodstatných bitů

Možnosti počítání
1. **Dělení polynomů** (hodnota CRC se určuje podle nejvyššího stupně v polynomu)
2. **If, XOR, Shift** (pro CRC4 můžeme XOR-ovat po čtveřicích, CRC16 po šestnácticích apod.; využívá se k tomu lookup table)

-pro CRC16   g(x) = 0x1021 = 10001000000100001b = $x^{16} + x^{12} + x^5 + 1$

17. konvoluční kód

-konvoluční kódy zpracovávají data kontinuálně na rozdíl od předchozích blokových kódů
-jedná se o třídu rekurentních kódů(rekurze), jsou lineární a stále v čase
-parametry:
- -rychlost rek. Kodu: R=k0/n0
- -délka kodového omezení: v=m*k0 (m=počet předchozích skupin)
- -informační délka slova: k=(m+1)*k0
- -kodová delka bloku: n=(m+1)*n0
- -délka zřetězení: K=m+1

18. rovnice

-způsob zápisu vnitřních vazeb konvolučního kódu (jak vznikají výstupy)
-lze odvodit ze schématu, nebo z nich jde vytvořit schéma

1. Diferenční (např. $y_{3i} = x_i + x_{i-1} + x_{i-2}$)
2. Polynomální (např. $g_{13} = 1 + x + x^2$) (lze zapsat G(x) = rov1   rov2   rov3)

19. schéma

-způsob zápisu vnitřních vazeb konvolučního kódu (jak vznikají výstupy)
-lze odvodit z rovnic, nebo z něj jde vytvořit rovnice

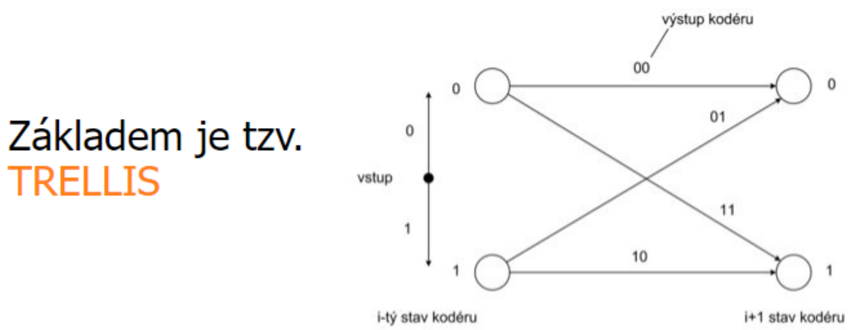
20. stavový diagram

=určuje jak zakódujeme hodnotu na vstupu při daných variantách naplnění pamětí

-vytvoříme 2^n uzlů, kde n = počet pamětí
-pro každý uzel určíme co se stane, když dojde 1 a 0
-hodnoty na cestách diagramu určíme buď pomocí rovnic nebo schématu
-pokud není určeno jinak vycházíme ze stavu, kde jsou v pamětech samé 0

21. Viterbiho dekodér

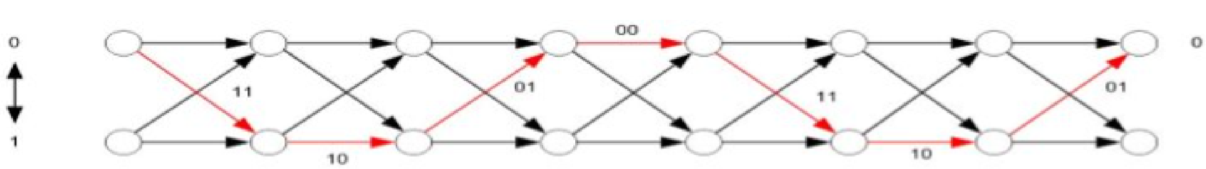
-vychází z kódového stromu (rozvinutého diagramu kodéru)
-zobrazuje se pomocí trellisu



-princip dekodovaní:
- -vychází z 0
- -čte postupně vstup
- -uvažuje možné kombinace a počítá jejich váhy (o kolik bitů se liší cesta od dané dvojice bitů na vstupu + nížší varianta z předchozích)
- -na výstupu zpětně volíme cestu s nejnižší váhou (pokud některé mají stejnou, pak náhodně)

-nejmenší číslo na výstupu říká, kolik ASI vzniklo chyb při přenosu

Př.: i = 1 1 0 0 1 1 0 (vždy se začíná ve stavu 0)



-využití konvolučních kodů např. GSM

Soft/Hard rozhodování = dekodér potřebuje z demodulátoru získat informaci o tom jaká je hodnota je pravděpodobnější (slabá/silná nula/jednička), na základě čehož se rozhodne

-hard->rovnou0/1

-soft->dle pravděpodobnosti je přisuzována váha 0/1

Děrování (punchering) = chceme snížit požadavky na přenosovou kapacitu kanálu za cenu toho že některé bity zahodíme. Bity, které vyřadíme určíme pomocí matice 3x2. Dekodér dokáže tyto chyby s 50% šancí opravit.

Prokládání (interleaving) = pokud vzniknou chyby ve shluku, je třeba je rozdělit, aby si s nimi další dekodér dokázal poradit

-pokud vytvoříme ze slov matici, tak jeden koduje řádky a druhý sloupce

22. trellis

-je to stavový vyhodnocovač, pomocí kterého realizujeme Viterbiho dekodér (viz.21)