ELEKTRO IN RAČUNALNIŠKA ŠOLA

POROČILO IZVEDENE VAJE PRI PREDMETU:

IKS

Številka in naslov vaje: 2. vaja

Datum: _____ **23.9.2024** **Šolsko leto: 2024/2025**

Ime in priimek dijaka: **Kevin Šertl** _____

Razred: **4.trb** _____

Učitelj: **Roman Herlah** _____

Ocena in podpis učitelja: _____

Naloga 1

Razložite postopek kodiranja in dekodiranja podatkov. Kako se podatki kodirajo v digitalni obliki, in kako se dekodirajo nazaj v izvorno obliko? Raziskujte različne metode kodiranja, kot so kodiranje v binarni obliki, Huffmanovo kodiranje…

Kodiranje podatkov je postopek pretvarjanja izvornih informacij v neko drugo obliko, običajno z namenom, da jih lahko računalnik razume in obdela. Najpogosteje se podatki kodirajo v binarni obliki, kjer se informacije predstavijo z uporabo ničel in enic. To omogoča računalnikom, da podatke shranijo in obdelajo.

Ena od metod je Huffmanovo kodiranje, ki zmanjšuje velikost podatkov s pomočjo krajšanja kod za bolj pogoste simbole in podaljševanja kod za manj pogoste simbole. Tako se zmanjša količina prostora, ki je potreben za shranjevanje informacij.

Dekodiranje je obratni postopek, kjer se kodirani podatki spet pretvorijo v svojo prvotno obliko, da jih lahko človek razume ali uporabi za različne namene.

Naloga 2

Raziskujte, kako se kodiranje uporablja v modernih komunikacijskih sistemih, kot so brezžični telefoni, satelitske komunikacije, in internet. Kakšne izzive prinaša kodiranje v teh sistemih in kako se rešujejo?

Analizirajte primer sodobne tehnologije, kot je 4G/5G ali Wi-Fi, in preučite vlogo kodiranja v teh sistemih.

V modernih komunikacijskih sistemih, kot so 4G, 5G in Wi-Fi, se kodiranje uporablja za zanesljiv in hiter prenos podatkov kljub šumom in motnjam. Pri 4G/5G se uporabljajo tehnike, kot so turbo kodiranje in LDPC, ki zmanjšajo napake brez ponovnega pošiljanja podatkov. Wi-Fi uporablja OFDM, ki signal razdeli na manjše dele, kar zmanjšuje vpliv motenj. Izzivi kodiranja vključujejo iskanje ravnovesja med hitrostjo, zanesljivostjo in učinkovito rabo pasovne širine.

Naloga 3

Razložite različne metode kodiranja signala, kot so Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), in Quadrature Amplitude Modulation (QAM). Kako te metode vplivajo na kakovost signala in prenos podatkov?

Amplitude Shift Keying (ASK) spreminja amplitudo signala glede na digitalne podatke. Ko je amplituda višja, predstavlja "1", ko je nižja, pa "0". Slabost ASK je občutljivost na šum, kar lahko povzroči poslabšanje kakovosti signala.

Frequency Shift Keying (FSK) spreminja frekvenco signala glede na podatke. Ena frekvenca predstavlja "1", druga pa "0". FSK je bolj odporen na šum kot ASK, a zahteva več pasovne širine.

Phase Shift Keying (PSK) spreminja fazo signala za prenos podatkov. Različne faze predstavljajo različne bite. PSK je bolj zanesljiv kot ASK in FSK, še posebej pri višjih hitrostih prenosa, vendar lahko pride do težav pri sinhronizaciji faz.

Quadrature Amplitude Modulation (QAM) kombinira spreminjanje amplitude in faze. To omogoča večji prenos podatkov, saj lahko prenaša več bitov hkrati. QAM ponuja visoko učinkovitost, a je občutljiv na šum in zahteva boljšo kakovost signala.

Vsaka metoda vpliva na kakovost in hitrost prenosa, odvisno od odpornosti na šum in razpoložljive pasovne širine.

Naloga 4

Podatku **10001101**, določi paritetni bit tako da bo veljala liha partiteta. V kakšnem primeru paritetni bit ne bo uspešno zaznal napake pri prenosu?

Za liho pariteto mora biti število enic v podatku liho. Podatek 10001101 ima trenutno 5 enic, kar je že liho število. Zato je paritetni bit 0, da ohrani število enic liho. Tako bo končni podatek z dodanim paritetnim bitom: 100011010.

Paritetni bit ne bo zaznal napake, če pride do parnega števila napak (npr. dveh ali več napak). V tem primeru bo število enic ostalo liho, paritetni bit pa bo "mislil", da ni napake, čeprav se je dejansko zgodila.

Naloga 5

Podan je naslednji podatek: **10011010**

- Izračunajte paritetni bit za **sodo in liho pariteto**. Zapišite končni podatek z dodanim paritetnim bitom.
- Pošljite podatke, ki ste jih izračunali v prejšnjem koraku. Uporabite oba niza (za sodo in liho pariteto) in preverite, če paritetni bit po prejemu podatkov še vedno velja.
- Spremenite en bit v podatkovnem nizu (naključno spremenite en **1** v **0** ali obratno) in simulirajte, da je prišlo do napake med prenosom. Razložite kako ste zaznali napako.
- Spremenite dva različna bita (dve napaki) v podatkovnem nizu. Preverite, ali bo paritetni bit zaznal napako. Razložite, zakaj v tem primeru paritetni bit ne zazna napake.

1. Sodo število je že, saj je sodo število ničl, da nastane liho pa je potrebno dodati še 1 ničlo.

2. Za sodo pariteto (100110100):

Niz ima 4 enice (10011010) + paritetni bit 0.

Število enic ostaja sodo, zato sodi paritetni bit velja.

Za liho pariteto (100110101):

Niz ima 4 enice (10011010) + paritetni bit 1.

Število enic je zdaj 5, kar je liho, zato liha pariteta velja.

3.Spremenimo en bit v podatkovnem nizu.

Recimo, da pri sodi pariteti spremenimo prvi bit iz 1 v 0, tako da niz postane: 000110100.

Ta niz ima zdaj 3 enice, kar je liho število.

Ker sodi paritetni bit zahteva sodo število enic, to pomeni, da smo zaznali napako, saj se število enic in pariteta ne ujemata.

4. Spremenimo dva bita v podatkovnem nizu.

Recimo, da spremenimo prvi bit in peti bit: 100110100 → 000100100.

Ta niz ima zdaj 2 enici, kar je še vedno sodo število.

Čeprav sta bili narejeni dve napaki, paritetni bit ne bo zaznal napake, ker se število enic in sodi paritetni bit še vedno ujemata. To je zato, ker paritetni bit zazna samo napake z lih številom sprememb bitov (ena, tri, pet, ...), ne pa parnih (dve, štiri, ...).

Naloga 6

Za spodaj podane podatkovne nize izračunajte kontrolno vsoto ter preverite ali je prišlo do napake pri prenosu.

a. 01001010 11110000 00001111
b. 10011001 11100010 00100100 10000100
c. 10100101 11010010 00101001

a. Primer:

   Seštejemo prva dva podatkovna bita in pride: 00111010. Potem prištejemo še zadnji bit in pride : **01001010**. In ni prišlo do napake

b. Primer:

   Ponovimo isto kot na prvem primeru in na koncu pride 00100100, zato ni prišlo do napake

c. Primer:
   Ponovimo isto kot na drugih primerih in na koncu pride 10011110, zato ni prišlo do napake

Naloga 7

Raziskujte in opišite zgodovinski razvoj metode CRC. V nalogi odgovorite na naslednja vprašanja:

- Kdo je izumil CRC in kdaj?
- Kakšne so glavne izboljšave CRC skozi čas in zakaj so bile potrebne?
- Opišite, kako se CRC uporablja danes v omrežnih protokolih, kot so Ethernet in USB. Raziščite tudi kako CRC pomaga zagotoviti zanesljivost prenosa podatkov.

Kdo je izumil CRC in kdaj?

CRC je bila prvič opisana leta **1961** s strani **W. Wesley Peterson**, ki je raziskoval telekomunikacijske metode.

Glavne izboljšave CRC skozi čas

Izboljšani algoritmi: Sčasoma so se razvili hitrejši algoritmi za izračun CRC, kar je bilo nujno za hitrejše omrežne tehnologije.

Različne dolžine CRC: Uvedene so bile različne dolžine (npr. CRC-16, CRC-32) za prilagoditev različnim potrebam po zanesljivosti.

Prilagojeni polinomi: Razvili so se polinomi, ki zagotavljajo boljšo zaščito pred specifičnimi napakami.

Kako se CRC uporablja danes

Ethernet: Uporablja CRC-32 za preverjanje celovitosti podatkovnih paketov. Zazna napake in zavrže napačne pakete.

USB: Uporablja CRC-16 in CRC-32 za zanesljivo komunikacijo med napravami.

Kako CRC pomaga zagotoviti zanesljivost prenosa podatkov

Zaznavanje napak: CRC učinkovito zaznava različne vrste napak pri prenosu.

Zmanjšanje ponovitev: Hitro zaznavanje napak zmanjšuje potrebo po ponovnem prenosu podatkov.

Robustnost: Uporaba CRC v kombinaciji z drugimi zaščitnimi metodami povečuje zanesljivost komunikacijskih sistemov.

Naloga 8

Preveri pravilnost prenosa podatkov s CRC naslednjim podatkom:

Ni me bilo v šoli zato upam da je prav(Pomagu z internetu)

a. SPOROČILO: 101101101
VZOREC: 101

101	10110110100
	−101
	001001
	−000
	0010010
	−000
	00100100
	−000
	001001000
	−101
	0000100

b. SPOROČILO: 101110101
VZOREC: 10110

10110	1011101010000
	−10110
	000001
	−00000
	0000010
	−00000
	00000100
	−00000
	000001000
	−10110
	000000100

c. SPOROČILO: 1100101100
VZOREC: 1101

1101	110010110000
	−1101
	000010
	−00000
	0000100
	−00000
	00001000
	−1101
	000001100

d. SPOROČILO: 10111010111
VZOREC: 1001

1001	10111010111000
	−1001
	001010
	−00000
	0010100
	−00000
	00101000
	−1001
	000010100

e. SPOROČILO: 1100110
VZOREC: 111

111	110011000
	−111
	00110
	−00000
	001100
	−111
	0001000