ŠOLSKI CENTER VELENJE

ELEKTRO IN RAČUNALNIŠKA ŠOLA

Trg mladosti 3, 3320 Velenje

MADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

ZAPISOVANJE PODATKOV NA DNK

Tematsko področje: RAČUNALNIŠTVO

Avtor:

Samo Pungaršek Pritržnik, 4. letnik

Mentor:

Islam Mušić, prof.

Velenje, 2020

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniški šoli.

Mentor: Islam Mušić

Datum predstavitve: marec 2020

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, Elektro in računalniška šola, šolsko leto 2019/2020

KG računalništvo / shranjevanje podatkov / biokemija / DNK

AV PUNGARŠEK PRITRŽNIK, Samo

SA MUŠIĆ, Islam

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2020

LI 2020

IN ZAPISOVANJE PODATKOV NA DNK

TD Raziskovalna naloga

OP VII, 35 str., 24 sl., 14 vir.

IJ SL

JI sl/en

AI Pred petimi leti smo ljudje ustvarili 4,4 zettabajtov podatkov. Do leta 2025 strokovnjaki predvidevajo, da se bo ta številka zvišala na 160 zettabajtov. Trenutna infrastruktura ne more prenesti takšnega povečanja količine podatkov, saj če ne bomo spremenili tehnologije, bomo porabili ves potrebovani silicij za ustvarjanje mikročipov do leta 2040. Preobremenjenost podatkov je torej težava 21. stoletja, shranjevanje podatkov na DNK pa je odgovor na to težavo. V raziskovalni nalogi sem si zadal cilj izdelave pretvorbe med binarnim številom ter med nukleotidi, ki so sestavni del DNK. Opisal bom tudi osnove DNK, ter kako natančno je zapisovanje in branje z DNK-ja.

KEY WORDS DOCUMENTATION

ND ŠC Velenje, Elektro in računalniška šola, šolsko leto 2019/2020

CX computer science / information storage / biochemistry / DNA

AU PUNGARŠEK PRITRŽNIK, Samo

AA MUŠIĆ, Islam

PP 3320 Velenje, SLO, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2020

PY 2020

TI DNA DATA STORAGE

DT RESEARCH WORK

NO VI, 32 p., 22 fig., 6 ref.

LA SL

AL sl/en

AB Five years ago we had produced 4,4 zettabytes of data. That is set to increase to 160 zetttabytes by 2025. Current infrastructure cannot handle the coming data deluge, which is expected to consume all microchip-grade sillicon by 2040. DNA data storage could be the answer to this 21st-century problem, which is information overload. In the research project, I set myself the goal to create the translation between the binary number composition and the nucleotids, which are the integral part od DNA. Furthermore I will explain the basics of DNA and the accuracy of writing and reading information from it.

KAZALO KRATIC

DNK – Deoksiribonukleinska kislina (angl. DNA)

CPE – Centralna Procesna Enota (angl. CPU)

HDD – Trdi disk (angl. Hard Drive Disk)

SSD – angl. Solid State Drive

CD – Zgoščenka (Compact Disc)

DVD – angl. Digital Versitale Disc

USB – (angl. Universal Serial Bus)

RAM – Pomnilnik (angl. Random Access Memory)

DRAM – Dinamični pomnilnik (angl. Dynamic Random Access Memory)

MOS – angl. Metal oxide semiconductor field-effect transistor (MOSFET)

RNK – Ribonukleinska kislina (angl. RNA)

ATCG – Adenin, Timin, Citozin, Gvanin

ATP - Adenozin trifosfat

PCR - Verižna reakcija s polimerazo (angl. Polymerase Chain Reaction)

ASCII – angl. American Standard Code for Information Interexchange

KAZALO VSEBINE

KAZALO DEFINICJ

KAZALO ENAČB

KAZALO SLIK

1 UVOD

Vsako minuto v letu 2018 je Google opravil 3,88 milijonov iskanj, ljudje pa so si ogledali 4,33 milijonov videoposnetkov, poslali 156.362.760 e-poštnih sporočil in objavili 49.000 fotografij. Do leta 2025 bo na svetovni ravni ustvarjenih približno 1,7 megabajtov podatkov na osebo na sekundo. Magnetni ali optični sistemi za shranjevanje podatkov, ki trenutno uporabljajo binarni sistem, ne bodo sposobni shranjevanja takšne količine podatkov. Poleg tega vodenje podatkovnih centrov zahteva ogromno energije.

Nadomestna možnost trdih diskov je shranjevanje podatkov na osnovi DNK. DNK je sestavljen iz dolgih verig nukleotidov A, T, C in G. Podatki se lahko shranijo v zaporedju teh črk in DNK pretvorijo v novo obliko informacijske tehnologije.

Možnost shranjevanja podatkov DNK ni zgolj teoretična. Leta 2017 je skupina na Harvardu ustvarila tehnologijo urejanja DNK CRISPR za snemanje slik človeške roke, ki so bile odčitane z več kot 90-odstotno natančnostjo. Poleg tega so v marcu, leta 2019, raziskovalci z univerze v Washingtonu in ekipa Microsoft Research razvili popolnoma avtomatiziran sistem za pisanje, shranjevanje in branje podatkov, kodiranih v DNK.

Medtem DNK že uporabljajo za upravljanje podatkov, in sicer kjer se raziskovalci spopadajo z ogromnimi količinami podatkov. Nedavni napredek tehnik sekvenciranja omogoča, da se več milijard DNK zaporedij prebere enostavno in hkrati.

Da bi bilo shranjevanje podatkov na DNK vsakdanje, vplivajo tudi izzivi. To so stroški in hitrost branja in pisanja podatkov na DNK, ki morajo pasti, če bo pristop konkurenčen elektronskemu shranjevanju.

Namen raziskovalne naloge je ustvariti lasten algoritem pretvarjanja med binarnim številskim sestavom ter nukleotidi. Raziskoval bom tudi o najbolj optimalnih rešitvah ter jih predstavil v tej nalogi.

1.1 Hipoteze

Hipoteze, ki sem si jih zastavil, so naslednje:

1. Shranjevanje na DNK je veliko počasneje, kot pa shranjevanje na trde diske.
2. Na DNK lahko shranimo neskončno število podatkov.
3. Shranjevanje na DNK lahko avtomatiziramo oziroma shranjevanje lahko naredimo samodejno.
4. Z uporabo računalniških algoritmov lahko pretvorimo podatke iz binarnega formata v ATCG-format.

2 PREGLED OBJAV

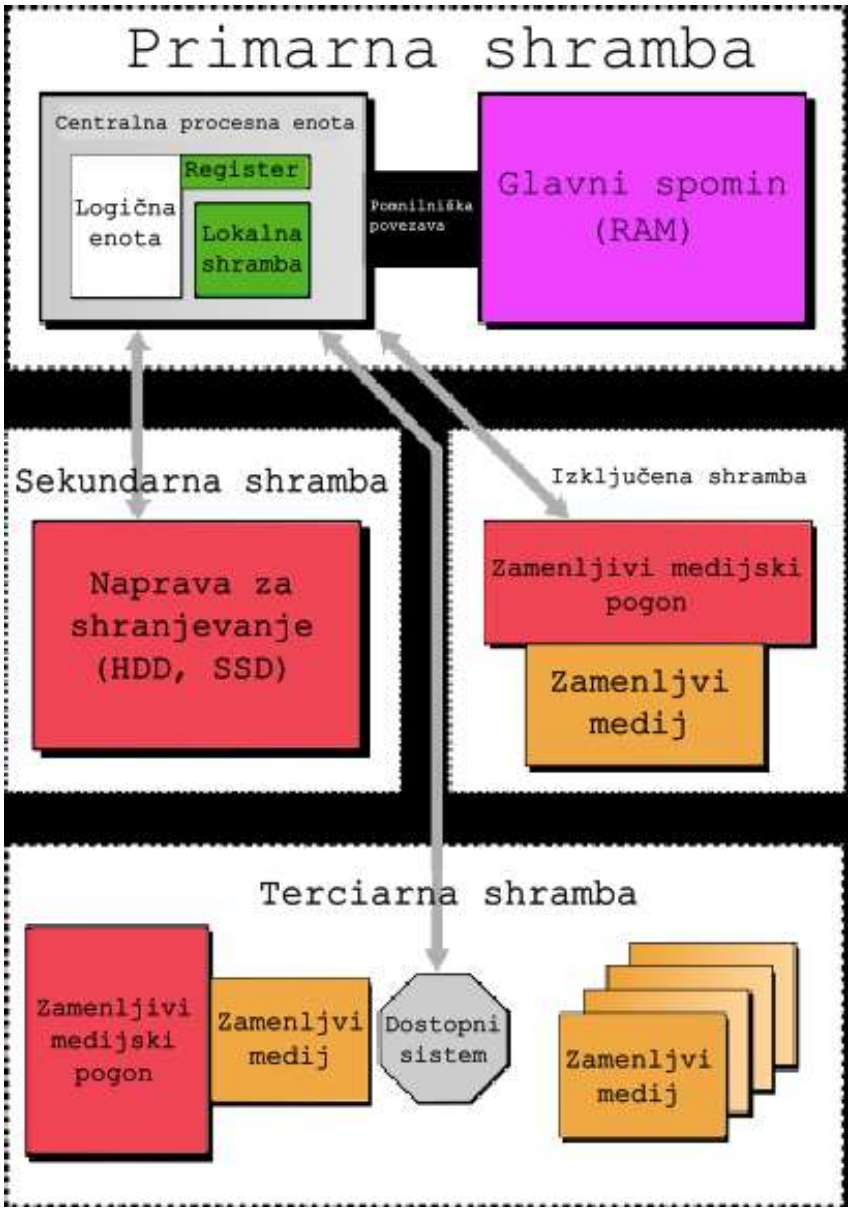
2.1 Računalniško shranjevanje podatkov

Računalniško shranjevanje podatkov, ki ga pogosto imenujemo shramba, je tehnologija, sestavljena iz računalniških komponent in snemalnih medijev, ki se uporablja za shranjevanje digitalnih podatkov.

Sodobni digitalni računalnik predstavlja podatke s pomočjo sistema binarnih številk. Besedilo, številke, slike, zvok in skoraj katero koli drugo obliko informacij je mogoče pretvoriti v niz bitov ali dvojiških števk, od katerih ima vsaka vrednost 1 ali 0. Najpogostejša enota shranjevanja je bajt, ki je enaka 8 bitom. Z informacijo lahko ravna kateri koli računalnik ali naprava, katere prostor za shranjevanje je dovolj velik, da lahko vsebuje dvojiško predstavitev informacij oziroma podatkov.

2.1.1 Hierarhija shranjevanja

Hierarhija shranjevanja se deli na tri razrede. To so primarna, sekundarna in terciarna shramba. Na naslednji sliki vidimo delovanje med shrambami (slika 1).



Slika 1: Prikaz delovanja hierarhije [1]

2.1.2 Primarna shramba

Primarni pomnilnik (znan tudi kot glavni pomnilnik, notranji pomnilnik ali glavni pomnilnik), ki ga pogosto imenujemo tudi pomnilnik, je edini, ki je neposredno dostopen CPE-ju. CPE neprestano bere tam shranjena navodila in jih po potrebi izvaja. Vsi podatki, ki se aktivno uporabljajo, se tudi tam hranijo enotno.

Glavni pomnilnik je neposredno ali posredno povezan s centralno procesno enoto preko pomnilniškega vodila. Gre pravzaprav za dva vodila: naslovno vodilo in podatkovno vodilo. CPE najprej pošlje številko skozi naslovno številko, številko, imenovano spominski naslov, ki označuje želeno lokacijo podatkov. Nato prebere ali zapiše podatke v pomnilniške celice s pomočjo vodila podatkov. Poleg tega je enota za upravljanje pomnilnika majhna naprava med CPE in RAM, ki preračuna dejanski pomnilniški naslov, na primer za odvzem virtualnega pomnilnika ali druge naloge.

2.1.3 Sekundarna shramba

Sekundarni pomnilnik (znan tudi kot zunanji pomnilnik ali pomožni pomnilnik) se od primarnega pomnilnika razlikuje po tem, da CPE ni neposredno dostopen. Računalnik običajno uporablja svoje vhodno/izhodne kanale za dostop do sekundarnega pomnilnika in prenos želenih podatkov v primarni pomnilnik. Sekundarno shranjevanje je nehlapno shranjevanje podatkov, ko je napajanje izključeno. Sodobni računalniški sistemi imajo običajno za dva razreda več sekundarnih pomnilnikov, kot primarni pomnilnik, ker je sekundarno shranjevanje manj drago.

V sodobnih računalnikih se trdi diski (HDD ali SSD) običajno uporabljajo kot sekundarni pomnilnik. Čas dostopa na bajt za trde diske se običajno meri v milisekundah, medtem ko se čas dostopa na bajt za primarno shranjevanje meri v nanosekundah. Tako je sekundarno shranjevanje bistveno počasnejše od primarnega. Vrteče se optične naprave za shranjevanje, kot so pogoni CD in DVD, imajo še daljši čas dostopa. Drugi primeri tehnologij sekundarnega shranjevanja vključujejo USB bliskovne pogone, diskete, magnetni trak, luknjane kartice in diski RAM.

Ko glava za branje ali zapis diska na trde diske doseže pravilno namestitev in podatke, lahko do naslednjih podatkov na posnetku zelo hitro dostopa. Za zmanjšanje časa iskanja in rotacijske zamude se podatki v velikih sosednjih blokih prenašajo na diske in z njih. Zaporedni dostop na diskih je na velikosti hitrejši od naključnega dostopa, zato so bile razvite številne prefinjene paradigme za načrtovanje učinkovitih algoritmov, ki temeljijo na zaporednem in blokovskem dostopu. Drug način zmanjšanja ozkega grla zapisa ali branja je uporaba več diskov vzporedno, da se poveča pasovna širina med primarnim in sekundarnim pomnilnikom.

Sekundarni pomnilnik je pogosto oblikovan v skladu z obliko datotečnega sistema, ki zagotavlja abstrakcijo, potrebno za organiziranje podatkov v datoteke in mape, hkrati pa vsebuje tudi podatke, ki opisujejo lastnika določene datoteke, čas dostopa, dovoljenja za dostop in druge informacije.

Večina računalniških operacijskih sistemov uporablja koncept virtualnega pomnilnika, kar omogoča uporabo več primarne pomnilniške zmogljivosti, kot je fizično na voljo v sistemu. Ko se primarni pomnilnik napolni, sistem premakne najmanj uporabljene koščke (strani) v izmenjalno datoteko ali datoteko strani v sekundarnem pomnilniku in jih pozneje po potrebi pridobi. Če je veliko strani premaknjeno v počasnejši sekundarni pomnilnik, je delovanje sistema poslabšano.

2.1.4 Terciarno shranjevanje

Terciarno shranjevanje ali terciarni pomnilnik je raven pod sekundarnim pomnilnikom. Običajno vključuje robotski mehanizem, ki bo vgradil (vstavil) in demontiral odstranljive množične pomnilniške medije v napravo za shranjevanje v skladu z zahtevami sistema; takšni podatki se pred uporabo pogosto kopirajo v sekundarni pomnilnik. Uporablja se predvsem za arhiviranje redko dostopnih informacij, saj je veliko počasnejše od sekundarnega shranjevanja (npr. 5–60 sekund v primerjavi z 1–10 milisekund). To je koristno predvsem za izjemno veliko shranjevanje podatkov, do katerih dostopajo brez človeških operaterjev. Tipični primeri vključujejo knjižnice trakov in optične avtomate (slika 2).

Slika 2: Slika knjižnice trakov [2]

Ko mora računalnik prebrati informacije iz terciarne shrambe, se najprej posvetuje s kataloško bazo podatkov, da ugotovi, kateri trak ali disk vsebuje informacije. Nato bo računalnik ukazal robotizirani roki, da vzame medij in ga postavi v pogon. Ko računalnik konča branje informacij, bo robotska roka vrnila medij na svoje mesto v knjižnici.

2.1.5 Mediji za shranjevanje

Najpogosteje uporabljeni nosilci za shranjevanje podatkov so polprevodniški, magnetni in optični.

2.1.5.1 Polprevodniški pomnilniki

Polprevodniški pomnilnik uporablja čipe z integriranim vezjem na osnovi polprevodnikov za shranjevanje informacij. Podatki so običajno shranjeni v spominskih celicah kovin-oksid-polprevodnik (MOS). Polprevodniški pomnilniški čip lahko vsebuje milijone pomnilniških celic, sestavljenih iz drobnih MOS-tranzistorjev s poljskim učinkom ali MOS-kondenzatorjev. Obe, hlapne in nehlapne oblike polprevodniškega pomnilnika, obstajajo.

V sodobnih računalnikih primarno shranjevanje skoraj izključno sestavlja dinamični hlapni polprevodniški pomnilnik z naključnim dostopom (RAM), zlasti dinamični pomnilnik z naključnim dostopom (DRAM). Od preloma stoletja je vrsta nehlapnih polprevodniških pomnilnikov s plavajočimi vrati, ki jih poznamo kot bliskovni pomnilnik, vedno bolj pridobil kot shranjevanje za domače računalnike.

2.1.5.2 Magnetno shranjevanje podatkov

Magnetno shranjevanje za shranjevanje informacij uporablja različne vzorce namakanja na magnetno prevlečeni površini. Magnetno shranjevanje je nehlapno. Do informacij lahko dostopate z eno ali več glavami za branje/pisanje, ki lahko vsebujejo enega ali več snemalnih pretvornikov. Glava za branje/pisanje pokriva samo del površine, tako da se morata glava ali medij ali oboje premakniti glede na drugega, da se dostopa do podatkov.

V sodobnih računalnikih bo magnetno shranjevanje potekalo v naslednjih oblikah:

- Magnetni disk
    - Disketa, ki se uporablja za shranjevanje brez povezave.
    - Trdi disk, ki se uporablja za sekundarno shranjevanje.
- Magnetni trak, ki se uporablja za terciarno in zunanje shranjevanje.
- Pomnilnik vrtiljaka (magnetni zvitki).

2.1.5.3 Optično shranjevanje podatkov

Optična shramba, tipični optični disk, shranjuje informacije v deformacijah na površini krožnega diska in jih bere tako, da osvetli površino z lasersko diodo in opazuje odboj. Shranjevanje optičnih diskov je nehlapno. Deformacije so lahko trajne (samo za branje medijev), ki se tvorijo enkrat (pišejo enkrat mediji) ali so reverzibilne (zapisljive ali za branje /pisanje medijev).

2.1.5.4 Pomnilnik vakuumske cevi

Te primarne naprave za shranjevanje so bile na trgu kratkotrajne, saj je bila Williamsova cev nezanesljiva, cev Selectron pa draga.

2.1.5.5 Elektroakustični pomnilnik

Elektroakustični pomnilnik je za shranjevanje informacij uporabil zvočne valove v snovi, kot je živo srebro. Začasni pomnilnik vrstice je bil dinamično spremenljiv, ciklično zaporedno branje/zapisovanje, in je bil uporabljen za primarno shranjevanje.

2.1.5.6 Holografsko shranjevanje podatkov

Holografski pomnilnik shranjuje podatke optično znotraj kristalov (slika 3). Holografska shramba lahko porabi celoten volumen pomnilniškega medija, za razliko od optičnega diska, ki je omejen na majhno število površinskih plasti. Holografska shramba bi bila nehlapna, zaporedna dostopnost, bodisi pisanje enkrat ali shranjevanje/branje/pisanje. Uporablja se lahko za sekundarno shranjevanje in shranjevanje brez povezave.

Slika 3: Izgled holografske shrambe [3]

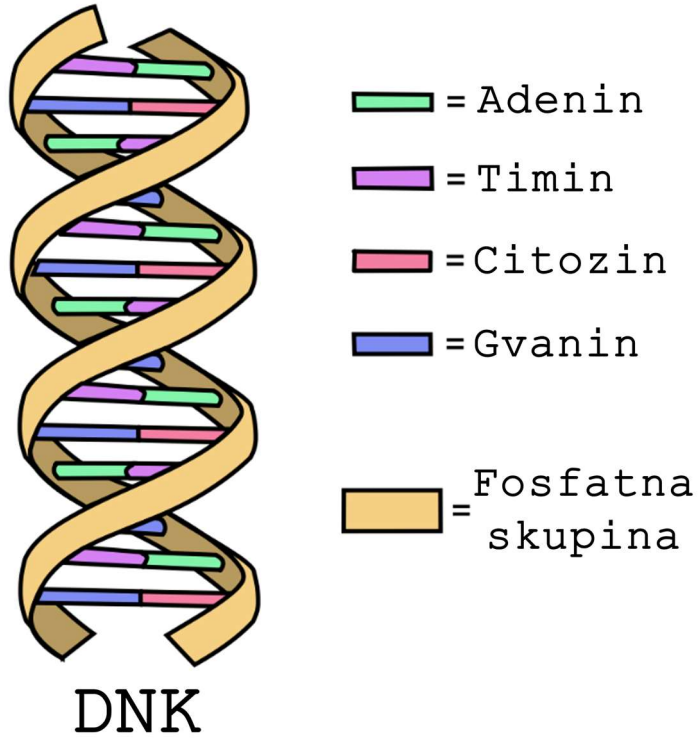
2.1.5.7 Molekularni spomin

Molekularni spomin shranjuje informacije v polimer, ki lahko shrani električni naboj. Molekularni pomnilnik je morda posebej primeren za primarno shranjevanje. Pomnilniška zmogljivost molekularnega pomnilnika je 10 terabitov na kvadratni palec.

2.2 DNK

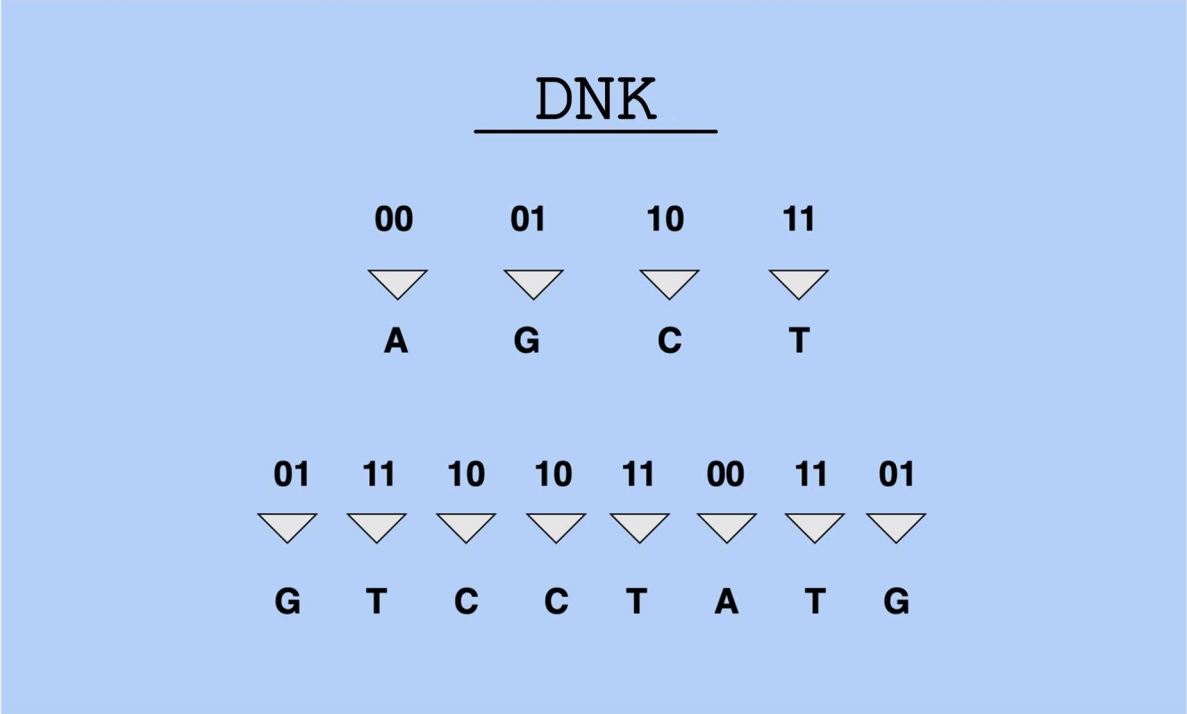
Deoksiribonukleinska kislina (DNK) je molekula, ki je nosilka genetske informacije v vseh živih organizmih. DNK skupaj z molekulo ribonukleinske kisline (RNK) spada med nukleinske (jedrne) kisline. Glavna vloga molekule DNK je shranjevanje bioloških informacij.

DNK je polimer, katerega osnovna enota je nukleotid. Nukleotid v DNK je sestavljen iz sladkorja, dušikove baze (adenin, citozin, gvanin in timin (slika 4)) in fosfatne skupine. Zaporedje nukleotidov določa pomen genetske informacije. V vseh živih organizmih ima DNK obliko dvojne vijačnice, pri čemer se dve molekuli DNK ovijeta druga okrog druge. Pri tem so dušikove baze znotraj vijačnice in se medsebojno vežejo v parih. Adenin se vedno pari s timinom in citozin vedno z gvaninom.



Slika 4: Prikaz DNK [4]

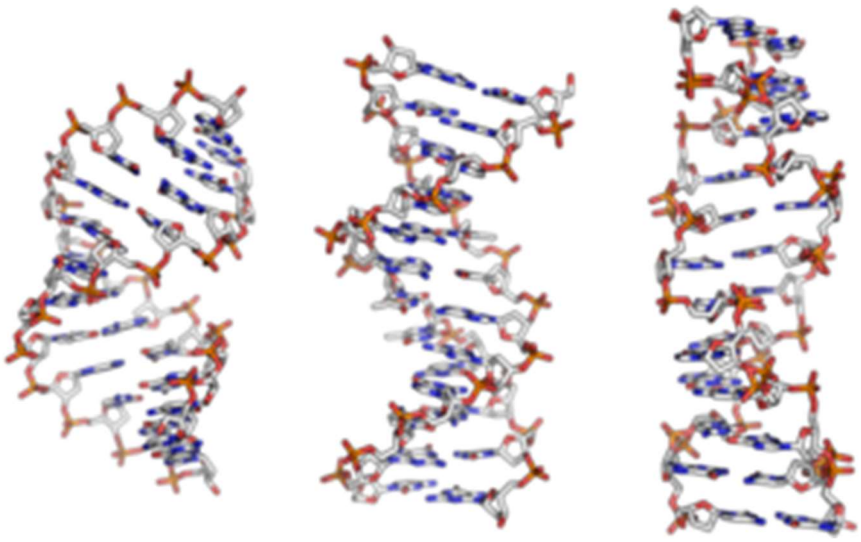
DNK lahko ob pomoči drugih sestavnih delov celice, ob dotoku hranilnih snovi ter energije, v obliki molekul ATP sintetizirajo različne beljakovine v različnih zaporedjih. DNK ima vse nadzorne mehanizme, ki jih sicer poznamo iz računalniških programskih jezikov, in ki omogočajo, da DNK nadzoruje procese v celici in njenem okolju. Če so računalniški programi zapisani v obliki dvojiških zaporedij in je osnovna enota pri njih bajt (8 bitov), so genetski zapisi zapisani v obliki štiriških zaporedij in je osnovna enota pri le-teh kodon (trije pari nukleotidov (slika 5)).



Slika 5: Prikaz štiriškega zaporedja [4]

2.2.1 Lastnosti

Osnovna enota deoksiribonukleinske kisline (DNK) je nukleotid, sestavljen iz fosfatne skupine ter ene od štirih dušikovih baz, adenina, citozina, gvanina ali timina. Glede na slednje uporabljamo nukleotidne oznake A, C, G in T. Ogrodje polimerne DNK tvori par nasprotno obrnjenih ogrodij, ki so oviti tako, da so dušikove baze znotraj, kjer se medsebojno povežejo s šibkimi vodikovimi vezmi. Ker je teh šibkih vezi veliko, je končna povezava med dvema molekulama DNK zelo močna in razpade pri segrevanju. Verigi dvojne vijačnice sta nasprotno obrnjeni (slika 6), kar opisujemo s številkami ogljikovih atomov, zaporedji nukleotidov obeh verig pa se povezujejo le v parih A-T in C-G. DNK zaporedje običajno zapisujemo s 5', zato bi se zaporedje 5'-AATTGGCC-3' v dvojno vijačnico zvilo z nasproti obrnjenim zaporedjem 5'-GGCCAATT-3'. Za boljše razlikovanje med enoverižno in dvoverižno DNK uporabljamo tudi oznaki ssDNK (angl. single strand) in dsDNK (angl. double strand).



Slika 6: Prikaz različnih vijačnic DNK [5]

2.2.2 Funkcija

Vse funkcije DNK so posledice interakcij s proteini. Interakcije so lahko nespecifične ali specifične, torej se protein veže na točno določeno nukleotidno zaporedje. Proteini lahko spreminjajo obliko in zvijanje DNK, režejo ali lepijo verige, sintetizirajo komplementarne verige DNK, vplivajo na potek prepisovanja.

Torej DNK ima lahko več različnih funkcij. V tem poglavju bom naštel raziskovanju pomembne funkcije ter jih na kratko opisal.

2.2.2.1 Kodiranje genov

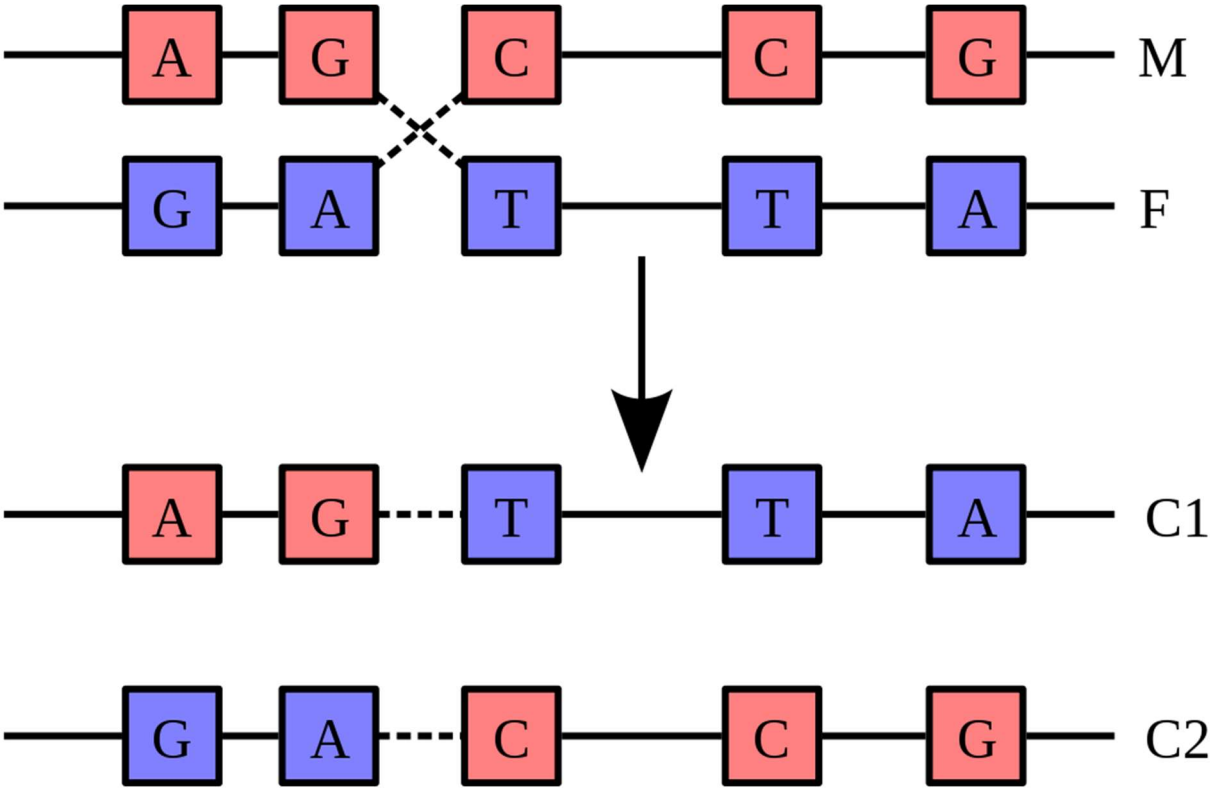
Gen je zaporedje nukleotidov molekule DNK in se večinoma dalje prevaja v aminokislinsko zaporedje - beljakovino. Pri prevajanju trije zaporedni nukleotidi - kodoni kodirajo eno aminokislino, kombinacije prevajanja imenujemo genetski kod.

2.2.2.2 Podvojevanje

Podvojevanje DNK je ključen proces prenosa dedne informacije na hčerinske celice oziroma pri višjih organizmih na potomce. Dvoverižna DNK se odvije in razpre, na osnovi enoverižne DNK se po principu komplementarnih nukleotidov sintetizira druga veriga v smeri 5' proti 3'. Tako dobimo dve dvoverižni vijačnici DNK, ki sta enaki originalu.

2.2.3 Mutacije

V najširšem pomenu je mutacija katerokoli trajna sprememba nukleotidnega zaporedja DNK. Pri večceličnih organizmih mutacije v telesnih celicah lahko privedejo do nenadzorovanega deljenja celic - rakavih obolenj, s stališča evolucije pa so bolj pomembne mutacije v klični liniji, torej tiste, ki se prenesejo na potomce. Mutacije se razlikujejo v obsegu od spremembe posameznega nukleotida do večjih prerazporejanj kromosomov v obsegu več milijonov nukleotidov. Razlike so tudi v posledicah, od nezaznavnih do pozitivnih. Številni popravljalni mehanizmi zagotavljajo, da se pri podvajanju zgodi malo sprememb in da se na potomce prenese malo mutacij, po drugi strani pa so mutacije tudi vir genetske raznolikosti, zato obstajajo tudi mehanizmi, ki jih ustvarjajo (slika 7).



Slika 7: Prikaz mutacije [6]

2.2.4 DNK-tehnike

V laboratoriju se DNK lahko izolira iz biološkega materiala ali sintetizira poljubno nukleotidno zaporedje omejene dolžine. Bolj pogosta je uporaba rekombinantne DNK, sestavljene iz odsekov izolirane in sintetizirane DNK. Dobljeni material se v primerni obliki lahko vstavlja v organizme, nastali genetsko spremenjeni organizmi lahko proizvajajo želene snovi ali pa so uporabni v celoti, na primer gensko spremenjene rastline v kmetijstvu

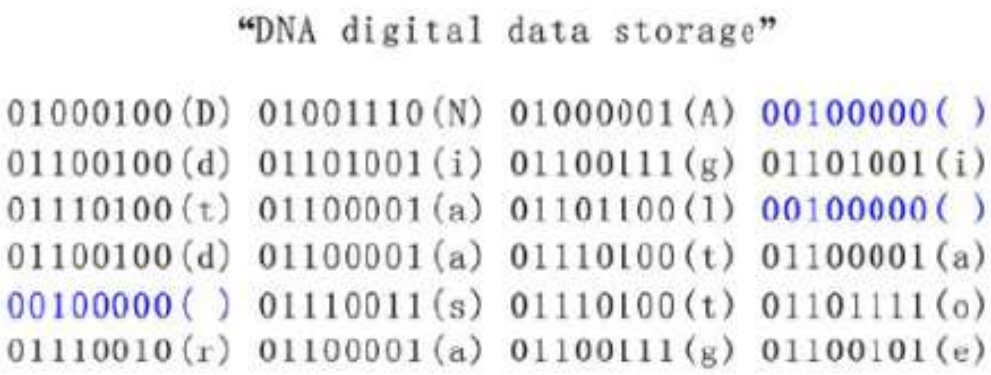
V forenziki je DNK uporaben zaradi razlik v nukleotidnem zaporedju med posamezniki. Na osnovi večjega števila izbranih točk na genomu določimo DNK-profil, za katerega je izjemno malo verjetno, da se pojavi pri dveh različnih osebah. S primerjavo DNK-profilov lahko določamo tudi starševstvo, saj vsak posameznik dobi polovico genetskega materiala od vsakega starša.

2.3 Digitalno shranjevanje podatkov na DNK

Shranjevanje DNA je postopek kodiranja in dekodiranja binarnih podatkov na in iz sintetiziranih verig DNK (deoksiribonukleinska kislina).

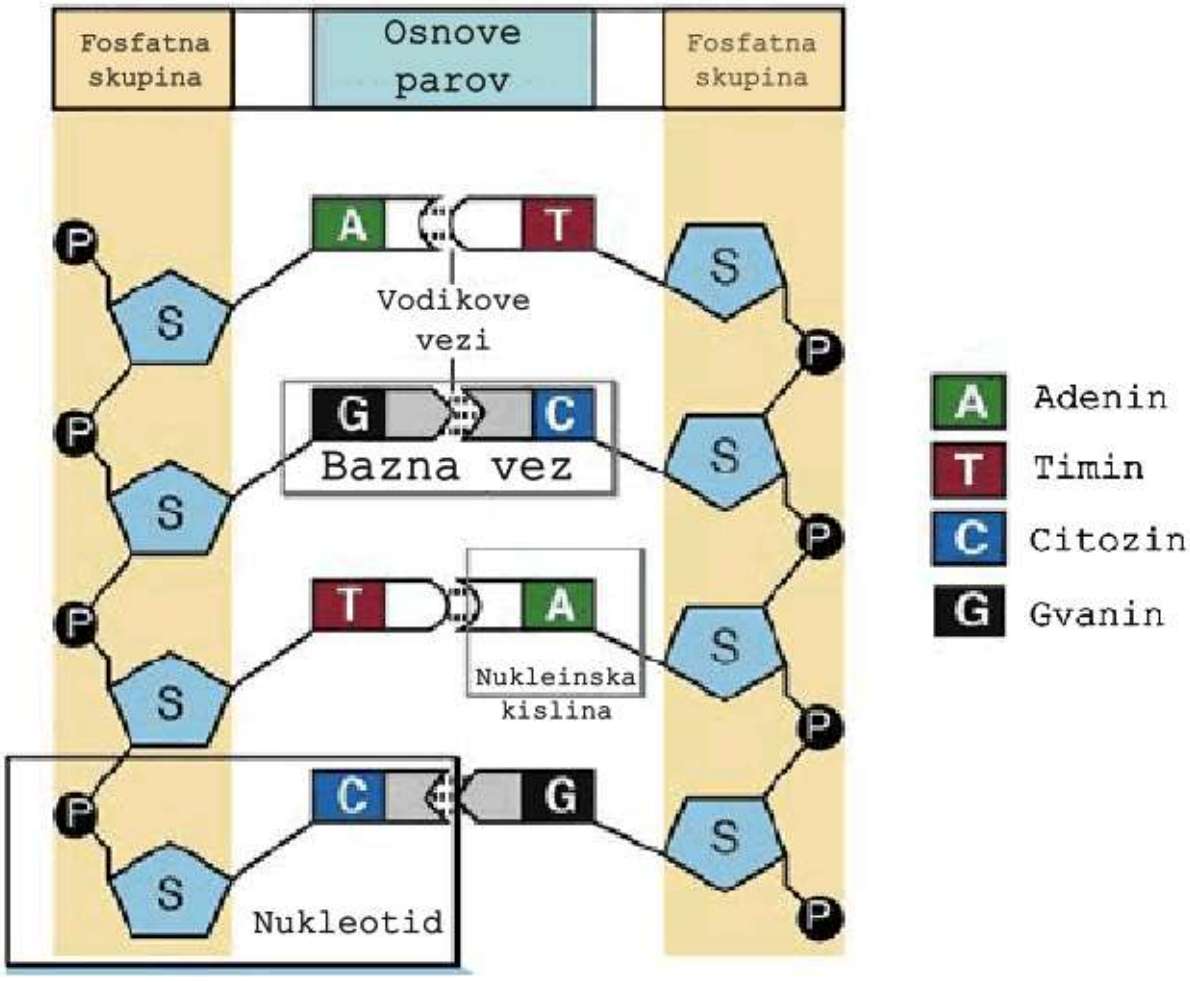
Za shranjevanje binarne digitalne datoteke kot DNK se posamezni biti (binarne števke) pretvorijo iz 1 in 0 v črke A, C, G in T. Te črke predstavljajo štiri glavne spojine v DNK: adenin, citozin, gvanin, in timin. Fizični medij za shranjevanje je sintetizirana molekula DNK, ki vsebuje te štiri spojine v zaporedju, ki ustreza vrstnem redu bitov v digitalni datoteki. Za obnovitev podatkov se zaporedja A, C, G in T, ki predstavljajo molekulo DNK, dekodirajo nazaj v prvotno zaporedje bitov 1 in 0.

Postopek za shranjevanje digitalnih podatkov DNA je kodiranje in dekodiranje binarnih podatkov v in iz sintetiziranih verig DNK. Besedila, številke, slike in drugo, berljivo ali vidno, se najprej pretvorijo v binarne jezike z 0 in 1 in nato kodirajo v DNA nukleotidna zaporedja, s štirimi bazami (A, C, G, T) namesto z 0 in 1. Na primer, velika črka "D" je v binarni obliki "01000100", mala črka "d" je "01100100", prazna "" je "00100000" (slika 8).



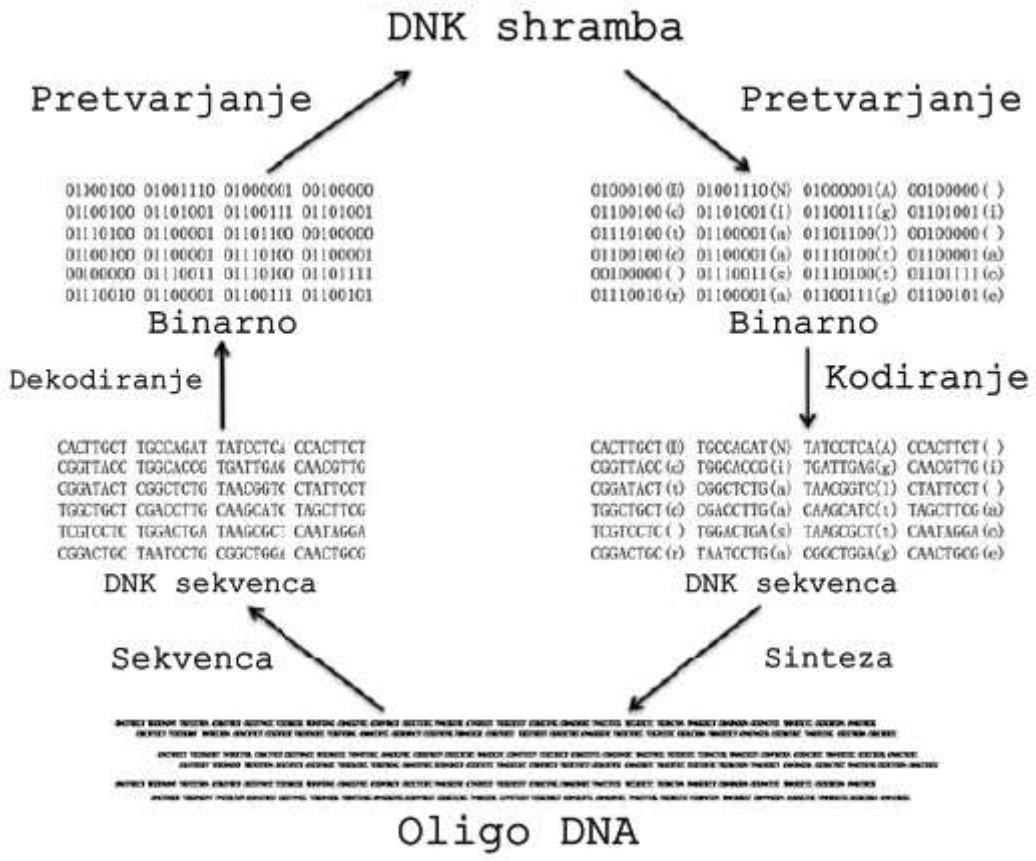
Slika 8: Prikaz kodiranja niza znakov "DNA digital data storage" [7]

Sliki 9 in 10 kažeta: stavek "DNA digital data storage" je bil pretvorjen v binarno različico, da bi pridobil binarne kode s 24 bajti. Nato se binarne kode (binarni biti) kodirajo v DNK. Vsaki od štirih baz (A, C, G in T) je treba dodeliti bodisi 1 ali 0. Na primer, purin (A, G) je dodeljen kot 1, pirimidin (C, T) pa kot 0. Ali pa sta obe bazi G in T dodeljeni kot 1s, drugi dve A in C pa sta 0s.



Slika 9: Podrobni prikaz DNK [7]

Za kodiranje podatkov je bil besedilni niz s 24 bajti ("DNA digital data storage ") pretvorjen v binarne bite, ki so pozneje kodirani v DNK z uporabo 1 bita na bazo, purin (A, G) pa je dodeljen kot 1 in pirimidin (C, T) kot 0s. DNK oligo je bil kemično sintetiziran, vsebina besedila pa je bila shranjena kot fragmenti DNK za dolgoročno shranjevanje. Za dekodiranje podatkov so fragmente DNK pomnožili s PCR, sekvencirali in dekodirali na binarne bite enkrat na dan, morali pridobiti podatke, da bi izšli binarni podatki, da so berljivi. Sčasoma mora branje podatkov iz knjižnice zaporedja DNK sekvencirati edinstvene molekule DNK, pretvoriti zaporedne informacije v izvirne digitalne podatke, če je potrebno ali je zahteva (slika 10).



Slika 10: Proces pretvarjanja podatkov [7]

2.4 Prednosti za shranjevanje na DNK

Kot že omenjeno, se globalni podatki močno povečujejo z eksponentno hitrostjo. Tradicionalni mediji se ne morejo dovolj spoprijeti z zahtevami po velikem shranjevanju podatkov. DNK lahko služi kot možen medij za shranjevanje digitalnih podatkov s svojimi potencialnimi prednostmi, kot so visoka gostota, visoka učinkovitost podvajanja, dolgoročna obstojnost in dolgoročna stabilnost. DNK s svojo največjo teoretično zmogljivostjo lahko kodira približno dva bita na nukleotid.

Zaradi visoke gostote pa lahko DNK, ki deluje kot medij za shranjevanje podatkov, shrani veliko količino podatkov v majhni velikosti. En sam gram DNK v svojem teoretičnem maksimumu lahko shrani približno 200 PB-podatkov, kar je skoraj dvakrat več kot v celotnem IBM-ovem podatkovnem centru. Z drugimi besedami, vse informacije, posnete po vsem svetu, se lahko shranijo v več kilogramih DNK ali so enake samo eni škatli za čevlje v primerjavi z zahtevami milijonov velikih centrov za shranjevanje podatkov za tradicionalne medije.

Podatkovno kodiran medij DNK je lahko dolgotrajno shranjen, ker ima visoko trajnost. DNK lahko traja več tisoč let na hladnih, suhih in temnih krajih. Tudi v slabših razmerah je razpolovni čas DNK do sto let. DNK lahko ostane stabilen pri nizki ali visoki temperaturi, in sicer v širokem razponu od -800 °C do 800 °C. Mediji z DNK lahko tudi zavarujejo podatke več kot tradicionalni digitalni nosilci podatkov. Čeprav se novi podatki povečujejo z eksponentno hitrostjo, jih je večina shranjenih v arhivih za dolgoročno hrambo. Ti podatki ne bodo takoj naloženi ali pogosto uporabljeni.

Druga prednost je, da je DNK zelo ohranjen. Naravni DNK se lahko natančno razmnožujejo z visokim izkoristkom in vedno s pravilom združevanja osnov (A s T, C z G). Tako lahko DNK medij zelo dolgo ohranja zvestobo podatkov.

2.5 Izzivi za shranjevanje podatkov na DNK

Glede na svoje edinstvene značilnosti in v primerjavi s tradicionalnimi mediji je DNK lahko potencialni in obetavni medij za digitalno shranjevanje podatkov . Vendar je pred komercialno uporabo DNK še dolga pot. Izzivi, s katerimi se moramo spoprijeti, obstajajo v različnih vidikih, vključno z visokimi stroški, nizko prepustnostjo, omejenim dostopom do shranjevanja podatkov, kratkimi sintetičnimi fragmenti DNK in hitrostjo napak pri sintezi in zaporedju.

Uporaba DNK pri shranjevanju podatkov je veliko dražja od drugih tradicionalnih medijev, kot so trak, disk in trdi disk (trdi disk). Trenutno stane za kodiranje in dekodiranje podatkov skoraj 15.000 USD na megabajt (MB). Medtem je trenutna tehnologija sinteze DNK omejena, sintetizirati je treba le kratke sekvence DNK. Največja dolžina vsakega fragmenta DNK-ja je omejena na nekaj sto nukleotidov.

Tako lahko za shranjevanje ene arhivirane datoteke zlasti 1 velika datoteka potrebuje stotine tisoč DNK fragmentov. Poleg tega je zamudno, da se podatki vpišejo in pridobijo iz DNK.

Pri pretvarjanju je več korakov, vključno s pretvorbo podatkov v binarno, kodiranje binarnega sistema v DNK, sintetiziranje in shranjevanje zaporedja DNK, zaporedja in dekodiranja podatkov v berljive podatke. Tradicionalni mediji, kot sta disk in trak, imajo svoje podatke o logičnem naslavljanju, vendar DNK ne.

Tako je zelo težko obravnavati edinstveno kodirano zaporedje DNK, za katero pričakujemo, da bo imel. Medtem je pomemben naključni dostop do shranjevanja podatkov na osnovi DNK, vendar nima zmožnosti naključnega dostopa.

S trenutnimi pristopi je za shranjevanje DNK na voljo le obsežni dostop. Celotno shranjevanje podatkov na osnovi DNK mora biti razvrščeno, zaporedno in dekodirano iz shranjevanja podatkov DNK, čeprav moramo samo prebrati en bajt. Zato je potreben pravi primer, ki se uporablja za selektivno pridobivanje pravega zaporedja DNK. To bo omogočilo tudi naključen dostop med zaporedjem DNK in pridobivanjem podatkov. Zaporedje z edinstvenim primerom lahko selektivno odčita le zahtevano DNK, ne pa celotne knjižnice DNK. In trenutno sinteza in sekvenca DNK nista povsem popolna. Med sintezo in sekvenciranjem DNK lahko pride do vstavitve, brisanja, substitucije in drugih napak, pri čemer je stopnja napake približno 1 % na nukleotid. Tehnologija in stroški sinteze in zaporedja DNK niso primerni za trenutno shranjevanje podatkov.

3 MATERIAL IN METODE DELA

V tem poglavju bom natančno opisal potek raziskovanja. Opisal bom programski jezik, v katerem sem naredil program, program pretvorbe ATCG-formata v binarni format, algoritem zaznavanja napak in redundanc in enačbe, potrebovane za zaznavanje napak pretvorbe.

Na začetku sem si moral izbrati programski jezik, v katerem bo možno najbolj optimalno zapisati program pretvarjanja formatov. Zaradi enostavnosti in preglednosti sintakse in semantike sem si izbral Python.

3.1 Python

Python je tolmačitveni skriptni programski jezik, ki ga je ustvaril Guido van Rossum leta 1990. Python ima popolnoma dinamične podatkovne tipe, samodejno upravlja s pomnilnikom in podpira funkcionalno, imperativno oziroma proceduralno, strukturirano in objektno orientirano računalniško programsko paradigmo. Zaradi dinamičnih podatkovnih tipov je podoben jezikom Perl, Ruby, Scheme, Smalltalk in Tcl. Razvili so ga kot odprtokodni projekt, ki ga je upravljala neprofitna organizacija Python Software Foundation (slika 11). Python se v glavnem uporablja za računalniško analitiko in razvijanje internetnih aplikacij.

Slika 11: Tukaj vidimo logotip Python programskega okolja [8]

3.2 Program za pretvorbo

V naslednjem koraku sem izdelal svojo različico pretvorbe formatov. Kot sem omenil v enem izmed prejšnjih poglavij, sem program izdelal v Python skriptnem jeziku. Zaradi razumljivosti sem izdelal še podaljšek kode, ki potem pretvori binarni format v ASCII-format oziroma v črke in besede.

Na prvi sliki sem deklariral niz ATCG-formata. Nato sem izpisal dolžine sekvence, ki jo bomo pozneje potrebovali. Izpisal sem tudi še niz in prvo črko niza (slika 12).

```
1   niz =
    "CTGCTACACGTCAGATAGTAGTACAGTCTGCACGTCTGTGCAGCACACAGTGAGTGCGTCTGT
    GCGTGCATCAGTCGTCACTGACTCA"
2
3   print("Dolžina sekvence: %i" % len(niz))
4   print("DNK format: " + niz)
5   print(niz[0:1])
```

Slika 12: Prvi del kode, lasten vir

Na naslednji sliki vidimo izpis simulacije programa (slika 13). Torej dolžina sekvence našega testa je 88, prva črka pa C.

```
Dolžina sekvence: 88
DNK format: CTGCTACACGTCAGATAG
CGTCTGTGCGTGCATCAGTCGTCACTGACT
C
```

Slika 13: Izpis prvega dela kode, lasten vir

V naslednjem delu kode vidimo, da sem najprej deklariral list pretvorba, v katero bom shranjeval vsako številko, ki bo zamenjala A, C, T ali pa G. V nadaljevanju vidimo »for« stavek, ki prelista čez vsako črko v ATCG-nizu in za njegovo vrednost zapiše številko 0 ali pa 1 v list pretvorba. Torej če je črka enaka 'A' ali pa 'C', potem se zapiše 0, če pa je črka enaka 'T' ali pa 'G', se zapiše 1 v list (slika 14).

```
 9   pretvorba = []
10   for i in range(len(niz)):
11       if niz[i] == 'A':
12           pretvorba.append(str(0))
13       elif niz[i] == 'C':
14           pretvorba.append(str(0))
15       elif niz[i] == 'T':
16           pretvorba.append(str(1))
17       elif niz[i] == 'G':
18           pretvorba.append(str(1))
```

Slika 14: Pretvarjanje ATCG-formata, lasten vir

Izpis tega lista zgleda sedaj tako (slika 15). Kot vidimo, to ni ustrezen tip binarnega formata, torej ga moremo še spremeniti.

```
['0', '0', '1', '1', '0', '1', '0', '0', '0', '0',
', '1', '0', '1', '0', '1', '1', '0', '1', '1', '0'
'1', '0', '1', '1', '0', '0', '0', '1', '1', '0', '
, '0', '0', '1', '0', '0', '0', '0', '0', '0', '1',
1', '1', '1', '0', '1', '1', '0', '1', '1', '1', '1
 '1', '0', '0', '1', '0', '0', '1', '1', '0', '1',
', '1', '1', '0', '0', '1', '0', '0']
```

Slika 15: Izpis lista pretvorba, lasten vir

Nato sem moral združiti list v skupen niz. To sem naredil s funkcijo ».join«, ki po izvedbi združi list v en skupen niz (slika 16).

```
24    skupen_niz = "".join(pretvorba)
25    print("Binarni format: " + skupen_niz)
```

Slika 16: Združitev elementov lista, lasten vir

Na naslednji sliki pa lahko vidimo izpis končnega binarnega formata (slika 17).

```
Binarni format: 001101000011010111011101111011100100110110
```

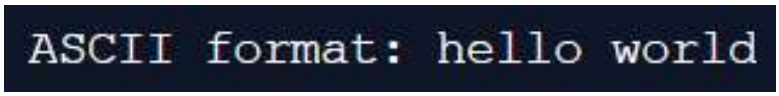
Slika 17: Izpis končnega binarnega formata, lasten vir

Zatem pa sem se odločil narediti še podaljšek kode, ki nam omogoča pretvorbo podatkov iz binarnega formata v ASCII-format. To pretvorbo lahko vidimo na naslednji sliki (slika 18).

```
28    skupen_niz = skupen_niz[2:]
29    skupen_niz = -len(skupen_niz) % 8 * '0' +
      skupen_niz
30
31    for j in range(1):
32        bloki = (skupen_niz[(i + j):(i + 8 + j)]
          for i in range(0, len(skupen_niz), 8))
33        prevod = ''.join(chr(int(char, 2)) for
          char in bloki)
34        print(' ')
35        print('ASCII format: ' + prevod)
```

Slika 18: Pretvorba v ASCII-format, lasten vir

Tukaj pa vidimo se izpis ASCII-formata (slika 19).



Slika 19: Izpis končne pretvorbe, lasten vir

3.3 Program za odkrivanje in popravljanje napak

V tem poglavju se bom osredotočil na kode. Koda je način zapisa simbolov, s pomočjo kodiranja predstavljamo neke informacije. Na primer, večina računalnikov uporablja ASCII-kode za predstavitev znakov.

Koda je množica elementov, ki predstavljajo informacije.

Kodne besede so elementi kode.

Število n po navadi predstavlja dolžino kodne besede, število M pa število kodnih besed neke kode.

Blok koda dolžine n, ki vsebuje M-kodnih besed nad abecedo A, je množica, ki vsebuje M n, kjer ima vsak n komponente iz abecede A. Pravimo jim [n, M] - kode nad A. Največkrat je A binarna abeceda, to pomeni A = {0, 1}. Oklepaje ( ) uporabljamo za zapis posebne vrste kode, ki se imenuje linearna koda. Koda C naj ima besede dolžine n nad abecedo A. Torej n nad A, ki so v C, so kodne besede. Kanalski kodirnik zmeraj prenaša kodne besede, medtem ko so n, ki jih kanalski dekodirniki prejme, lahko kodne besede ali pa tudi ne. Besedo ”vektor” bomo uporabljali za n.

Pomemben pojem, povezan z [n, M] - kodo C, je Hammingova razdalja za C.

Hammingova razdalja d(x, y) med dvema kodnima besedama x in y je število pozicij komponent, v katerih se besedi razlikujeta.

Definicija 1: Definicija hammingove kode

Za lažje razumevanje bom naredil 2 primera.

Primer 1:

d(x, y) = 3 , kjer je

A = {0, 1}, za katere sta

x = (10110) in

y = (11011).

Hammingova razdalja je v tem primeru d(x, y) = 3, saj se besedi razlikujeta na drugi, tretji in peti komponenti.

Primer 2:

d(u, v) = 4, kjer je

A = {0, 1, 2}, za katere sta

u = (21012) in

v = (12001).

Hammingova razdalja pa je v tem primeru 4, saj se besedi razlikujeta na prvi, drugi, četrti in peti komponenti.

Ko dekodirnik prejme vektor x, mora sprejeti določene odločitve. Na razpolago ima sledeče:

1. ni bilo napake, x sprejmemo za kodno besedo,

2. prišlo je do napak, zato x popravimo v kodno besedo c,
3. prišlo je do napak, popravki niso mogoči.

Torej, da strnem, Hammingova koda je koda, za iskanje in popravljanje napak v pretvorbi. Hammingovo kodo bom tudi opisal na primeru programa, ki je tudi napisana v Python programskem jeziku.

Najprej je treba zakodirati podatke z matriko G. To lahko vidimo na naslednji sliki (slika 20).

```
143     def hamming_encode(self,data):
144         return np.mod(dot(data,self.G),4)
```

Slika 20: Kodiranje z matriko G, vir [9]

Nato moramo preveriti napake v Hammingovi kodi. To naredimo v naslednjem delu algoritma za preverjanje in popravljanje napak (slika 21).

```
170     def _check_error(self,data):
171
172         H = self.H
173         checksum = (np.mod(dot(H,data.T),4)).T)[0]
174         loc = (checksum!=0).astype(int)
175         ind = 2**np.arange(len(loc))
176
177         h_ind = dot(loc,ind)
178         if h_ind ==0:
179             return (0,0)
180
181         bitpos = np.arange(0,H.shape[1]+1)
182         with np.errstate(divide='ignore'):
183             nparody = np.ceil(np.log2(bitpos))
184         error_bit_lut = bitpos-nparody
185         for (_x,_i) in zip(2**np.arange(self.parody_bits),np.arange(self.parody_bits)):
186             error_bit_lut[_x] = _i+self.data_bits+1
187
188         err_pos = int(error_bit_lut[h_ind]-1)
189
190         err_val = checksum[checksum!=0][0]
191         if np.all(err_val == checksum[checksum!=0]):
192             return (err_pos,err_val)
193         else:
194             return (np.nan,np.nan)
```

Slika 21: Preverjanje napak Hammingove kode, vir [9]

Iz prejšnjega dela kode vidimo, da funkcija lahko vrne 0 napak, pozicije napak in vrednost napak ter tretjo možnost, ki pa vrne napake.

To funkcija vrne naslednjemu delu kode, ki pa popravi te napake (slika 22).

```
170     def _check_error(self,data):
171
172         H = self.H
173         checksum = (np.mod(dot(H,data.T),4).T)[0]
174         loc = (checksum!=0).astype(int)
175         ind = 2**np.arange(len(loc))
176
177         h_ind = dot(loc,ind)
178         if h_ind ==0:
179             return (0,0)
180
181         bitpos = np.arange(0,H.shape[1]+1)
182         with np.errstate(divide='ignore'):
183             nparody = np.ceil(np.log2(bitpos))
184         error_bit_lut = bitpos-nparody
185         for (_x,_i) in zip(2**np.arange(self.parody_bits),np.arange(self.parody_bits)):
186             error_bit_lut[_x] = _i+self.data_bits+1
187
188         err_pos = int(error_bit_lut[h_ind]-1)
189
190         err_val = checksum[checksum!=0][0]
191         if np.all(err_val == checksum[checksum!=0]):
192             return (err_pos,err_val)
193         else:
194             return (np.nan,np.nan)
```

Slika 22: Popravljanje napak, vir [9]

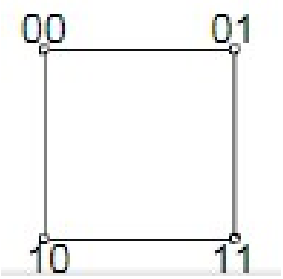
3.3.1 Geometrijska predstavitev števil

Binarno število podano z n-biti lahko predstavimo s točko v n-dimenzionalnem prostoru. Primeri teh dimenzionalnih prostorov so:

- 0, 1 (slika 23):

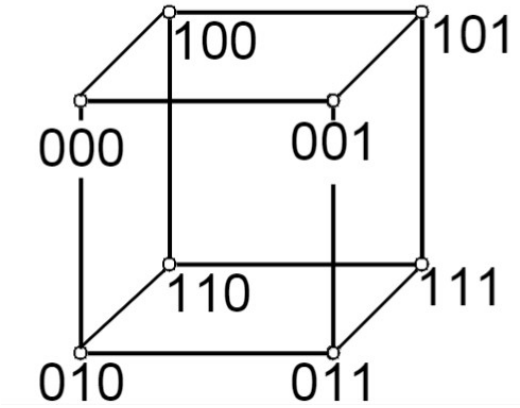
Slika 23: Prikaz v 1 dimenzionalnem prostoru, lasten vir.

- 00, 01, 10, 11 (slika 24):



Slika 24: Prikaz v 2 dimenzionalnem prostoru, lasten vir.

- 000, 001, 010, 011, 100, 101, 110, 111 (slika 25):



Slika 25: Prikaz v 3 dimenzionalnem prostoru, lasten vir.

Definicija n-kocke, ki jo dobimo z metodo projekcije je sledeča (definicija 1):

1. kocka reda 0 je posamična točka brez označbe
2. kocko reda n dobimo s projekcijo (n-1) kocke. Pri tem je 0 dodana kot predpona označbam točk originalne (n-1) kocke in 1 označbam projekcije.

Definicija 2: Definicija n-kocke

Torej n-dimenzionalna kocka ima 2^n ogljišč oziroma točk. Sub-kocko n-dimenzionalne kocke pa bomo označili z »p«. Kocka je definirana kot zbirka katerihkoli 2^p točk, ki imajo natančno (n-p) bitov enakih.

Primer: 000, 001, 100, 101; 2 je subkock; 3 je kock. Vsaki kocki reda n pripada natančno (enačba 1):

$$\frac{n!2^{n-p}}{(n-p)!p!}, \qquad C_{n-p}^{n} = \frac{n!}{(n-p)!p!}$$

Enačba 1: Enačba generacije kocke n-reda

Rezultat bi bil v tem primeru (3!22)/(2!1!) = 12 1- kock; to je robov in (3!21)/(2!1!) = 6. 2 - kock; to je ploskev – kvadratov.

1. 1. 2 Kodna razdalja

Razdalja med točkama n - dimenzionalne kocke je preprosto število koordinat (položajev bitov) v katerih se dvojiška predstavitev teh dveh točk razlikuje. Zato sem v prejšnjem poglavju razložil definicijo n-kocke.

Primer: 10110 in 01101 se razlikujeta v vseh razen v tretji koordinati (od leve ali od desne strani). Ker se torej točki razlikujeta v štirih koordinatah, je njuna medsebojna razdalja 4. Najprej definiramo vsoto po modulu 2 dveh bitov z ():

$$0 \oplus 0 = 0, \;\; 0 \oplus 1 = 0, \;\; 1 \oplus 0 = 0, \;\; 1 \oplus 1 = 0$$

Definicija 3: Matematična definicija hammingove kode.

Sedaj pa vzemimo binarno predstavitev dveh točk. Najprej definirajmo točki T_i in T_j ():

$$T_i = (a_{n-1}, a_{n-2}, \dots, a_1, a_0); \quad T_j = (b_{n-1}, b_{n-2}, \dots, b_1, b_0)$$

Definicija 4: Definicija točk Ti in Tj.

Nato pa sledi enačba T_k ():

$$T_k = T_i \oplus T_j = (a_{n-1} \oplus b_{n-1}, a_{n-2} \oplus b_{n-2}, \dots, a_0 \oplus b_0)$$

Enačba 2: Enačba Tk.

Ta vsota predstavlja neko drugo točko n - dimenzionalne kocke. Število enic binarne predstavitve točke Ti definirajmo kot utež točke Ti in mu dajmo simbolno oznako IT_iI. Razdaljo med točkama pa definirajmo kot enačbo ():

$$R(T_i, T_j) = |T_i \oplus T_j|$$

Enačba 3: Enačba izračuna razdalje med točkama.

Razdalja ima naslednje lastnosti ():

$$R(T_i, T_j) = 0, \text{če in samo če je } T_i = T_j$$

$$R(T_i, T_j) = R(T_i, T_j) > 0, \text{če he } T_i \neq T_j$$

$$R(T_i, T_j) + R(T_i, T_j) \geq R(T_i, T_j); \textit{trikotniška neenakost}$$

Definicija 5: Lastnosti izračuna razdalje.

Dve sosednji točki (povezani med seboj samo z enim robom) n - dimenzionalne kocke se med seboj razlikujeta natančno v eni koordinati, zato je njuna medsebojna razdalja 1. Vidimo torej, da med vsakima dvojicama točk n - dimenzionalne kocke, ki sta na razdalji R obstaja pot sestavljena iz R robov. Še več, poti je lahko več, toda nobena pot (za R > 1 in n > 2) med tema dvema točkama ni krajša kot je razdalja R. Najkrajša pot pa ima tudi to lastnost, da ne seka sama sebe in da se na njej nahaja R+1 oglišč vključno s končno točko.

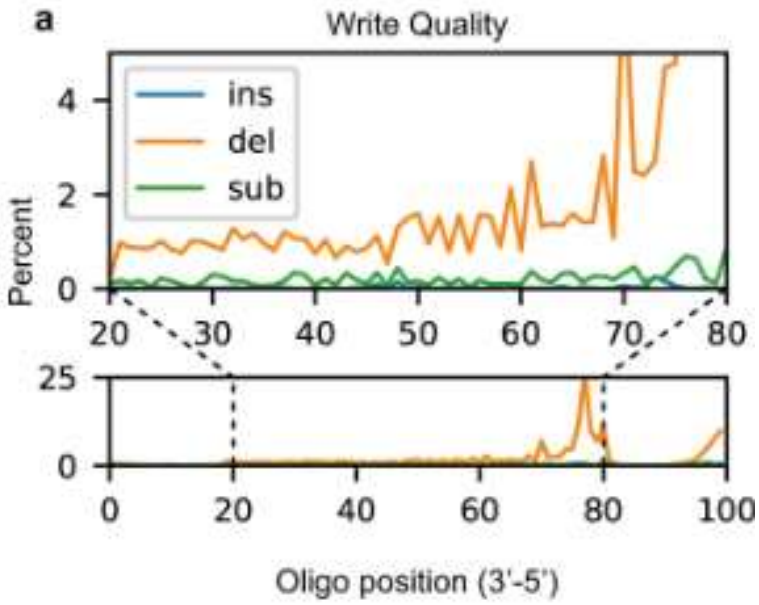
Rezultati

Raziskovanje zapisovanja podatkov na DNK sem začel z izdelavo svoje različice programa dekodiranja podatkov. Ko sem se prvič poglobil v teorijo, sem ugotovil, da je izdelava veliko težja, kot pa izgleda na prvi pogled, saj moramo v pretvarjanju upoštevati veliko možnih napak, ki se lahko zgodijo, najpogostejše izmed njih so redundance.

Preko leta sem izdelal svojo različico pretvorbe ATCG-formata v binarni format. Poleg tega sem se še poglobil v delovanje algoritma iskanja in popravljanja napak. Tega sem pozneje tudi implementiral.

Ampak glavni rezultat raziskovalnega dela še ni bil dosežen. Ker mi ni uspelo testirati algoritma na pravi napravi, sem se odločil da se bom skliceval še na raziskovalno delo Microsoft Research ekipe, saj so svojo različico programa za pretvarjanje formatov testirali.

Na naslednjem grafu lahko vidimo kvaliteto vstavljanja in brisanja podatkov (slika 26).

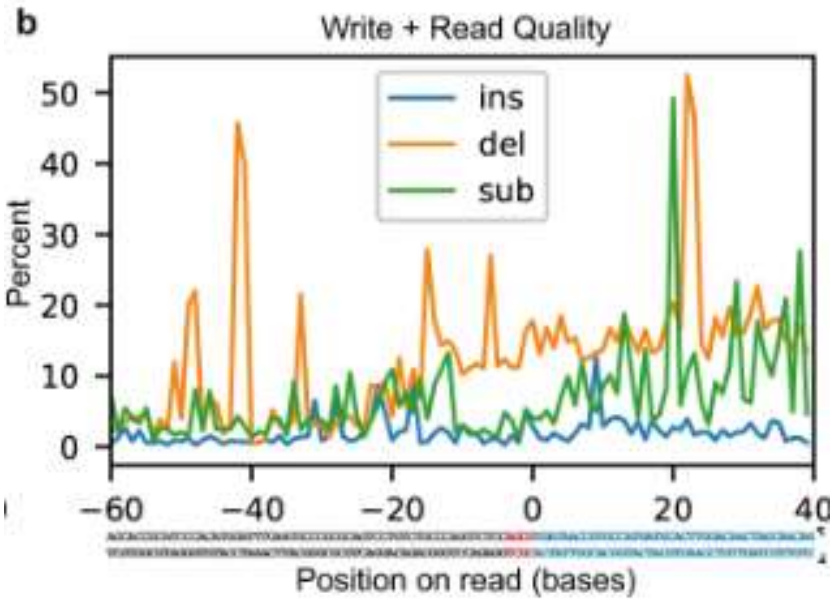


Slika 26: Kakovost procesa sinteze, vir [10]

Na tem grafu vidimo kakovost vstavljanja in brisanja. Vstavljanje podatkov je označeno z modro barvo, brisanje podatkov pa je označeno z oranžno barvo.

Kot vidimo, sta na sliki dva grafa. Spodnji predstavlja pregled napak med pozicijami 3' do 5', in sicer do maksimalne odstotne vrednosti, ki pa je 25 odstotkov. Zgornji del grafa pa je razširjen pogled na osrednjih 60 baz.

Poleg tega bom predstavil še graf, ki prikazuje kombinirano kakovost sinteze, ligacije in sekvenciranja med pisanjem in branjem (slika 27).



Slika 27: Kombinirana kakovost, vir [10]

Na tem grafu pa vidimo, da so od baze -60 pa do -4 adaptorske baze. To pomeni, da ta del baz ni možno spreminjati, in da v primeru spreminjanja podatkov, obstaja tveganje izgube celotnega tovora. Te so spodaj označene s sivo barvo.

Baze od -3 pa do 0 pa so brazgotine tovora. To je del tovora, na katerem je prišlo do napak. Ta del po navadi poskušamo čimbolj zmanjšati. V primeru, da pride do tega, je potrebno prepoznati brazgotine za ohranitev celovitosti podatkov. Te baze so spodaj označene z rdečo barvo.

Od 0 pa do 39 je sintetizirano območje tovora, kar prikazuje popolne podatke. Ta del pa je na tovoru modro obarvan.

3.4.1 Analiza hipotez

Na začetku raziskovanja sem si zastavil štiri hipoteze.

Prvo hipotezo (shranjevanje na DNK je veliko počasneje, kot pa shranjevanje na trde diske) sem potrdil, saj je samodejno shranjevanje zaenkrat še počasnejše od ostalih tehnologij za shranjevanje.

Drugo hipotezo (na DNK lahko shranimo neskončno število podatkov) sem zavrnil, saj lahko na določeno količino DNK shranimo samo določeno število podatkov. Je pa res, da je količina podatkov, ki jih lahko shranimo na DNK, veliko večja, kot pa na ostale tehnologije.

Tretjo hipotezo (shranjevanje na DNK lahko avtomatiziramo oziroma shranjevanje lahko naredimo samodejno) sem potrdil, saj je teoretično možno, kot sem opisal v nalogi. Poleg tega je Microsoft Research z univerzo Washington ustvaril prototip avtomatizacije marca 2019.

Četrto hipotezo (z uporabo računalniških algoritmov lahko pretvorimo podatke iz ATCG-formata v binarni format) pa sem potrdil, saj sem kljub mnogim izzivom izdelave ustvaril svojo različico pretvorbe.

4 ZAKLJUČEK

Da ponovim v raziskovalni nalogi sem opisal tehnologije za shranjevanje podatkov in njihove učinkovitosti. Nato sem opisal DNK in kako je sestavljen, ter nujno znanje, ki ga potrebujemo za izdelavo programa.

Za opisom sem začel z opisom izdelave algoritma za pretvorbo podatkov iz ATCG formata v binarni format, zaradi boljše preglednosti pa sem dodal še pretvorbo iz binarnega formata v ASCII format.

Poleg tega pa sem razložil še Hammingovo kodo, ki pa je za odkrivanje napak v pretvorbi, pisanju in branju podatkov. Za izdelavo raziskovalne naloge pa sem uporabil samo potrebovani del Hammingove kode.

Na koncu pa sem še zapisal rezultate raziskovalne naloge. Za pomoč sem se sklical na raziskovalen projekt ekipe Microsoft Research, saj nisem moral izdelati avtomatizacije branja in pisanja podatkov.

Z izdelovanjem te raziskovalne naloge sem pridobil veliko znanja o računalništvu, matematiki, biologiji ter biotehnologiji. Na začetku leta sem bil veliko bolj osredotočen na teoretičen del, saj mi je bilo zelo zanimivo, za praktičen del pa mi je še primanjkovalo znanja.

Čez nekaj časa, ko sem obvladal teorijo, sem pričel z izdelavo svoje različice pretvorbe podatkov. Takoj po začetku izdelave sem že naletel na prve težave, ampak zaradi motivacije nisem obupal in nadaljeval z delom.

V prihodnosti želim svoj algoritem še izpopolniti s Hammingovo kodo, kar bi omogočalo pretvorbo z minimalnimi napakami. Razvoj algoritma lahko spremljate še naprej:

https://github.com/SamoPritrznik/DNA-to-binary-conversion

Prav tako si v prihodnosti želim ustvariti še svojo različico avtomatizacije shranjevanja podatkov na DNK.

5 POVZETEK

Da povzamem, shranjevanje na DNK je rešitev, ki jo bomo v bližnji prihodnosti skoraj zagotovo morali začeti uporabljati, saj ne bomo mogli več shranjevati takšnih količin podatkov. Ampak da bo to možno, bo potrebno zmanjšati ceno izdelave avtomatizacije, saj je trenutno predrago.

V raziskovalni nalogi sem izdelal algoritem za pretvorbo formatov. Združil sem ga tudi s Hammingovo kodo za preverjanje in odpravljanje napak.

6 ZAHVALA

Za pomoč pri izdelavi raziskovalne naloge se zahvaljujem:

- mentorju Islamu Mušiću, ki mi je pomagal pri izdelavi raziskovalne naloge,
- učiteljici Marjetki Herodež in učitelju Antonu Kališniku za pomoč pri razlagi matematičnih formul, uporabljenih za izdelavo pretvorbe,
- učiteljici Lidiji Šuster za lektoriranje,
- Katarini Perger za konstantno podporo in prenašanje mojega pritoževanja, saj brez nje ta naloga ne bi obstajala v taki širini kot je vidno sedaj.
- vsem prijateljem in sorodnikom za podporo pri nastajanju naloge ter
- ŠC Velenje – Elektro in računalniški šoli, ki mi je omogočala izdelavo raziskovalne naloge.

7 VIRI IN LITERATURA

1. https://www.computerhistory.org/revolution/memory-storage/8/249 (29. 10. 2019)
2. https://www.sciencedirect.com/topics/computer-science/tertiary-storage (29. 10. 2019)
3. https://www.colorado.edu/faculty/mcleod/research/holographic-optical-data-storage (29. 10. 2019)
4. https://hackernoon.com/dna-data-storage-d0f0e93513b (29. 10. 2019)
5. https://ghr.nlm.nih.gov/primer/basics/dna (10. 11. 2019)
6. https://www.khanacademy.org/science/biology/classical-genetics/chromosomal-basis-of-genetics/a/linkage-mapping (10. 11. 2019)
7. http://www.hsj.gr/medicine/dna-and-the-digital-data-storage.php?aid=24516 (11. 11. 2019)
8. https://bioinfotraining.bio.cam.ac.uk/images/bioinfo-python/image_view_fullscreen (12. 1. 2020)
9. https://www.nature.com/articles/s41598-019-41228-8 (12. 1. 2020)
10. https://www.nature.com/articles/s41598-019-41228-8/figures/2 (12. 1. 2020)
11. https://ashutoshviramgama.com/dna-data-storage-synthetic-dna-future-of-storage/ (12. 1. 2019)
12. https://www.researchgate.net/figure/Binary-transcoding-methods-used-in-DNA-based-data-storage-schemes-A-One-binary-bit-is_fig1_333908428 (13. 1. 2019)
13. Church, G. M., Gao, Y., Kosuri, S. 2012. Science. Next-generation digital information storage in dna.
14. Hamming, R. W. 1950. Error-detecting and error-correctiong codes.