SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Antonio Kovačić

DNA KRIPTOGRAFIJA

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, srpanj, 2014.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj diplomski rad posvećujem svojim roditeljima, sestri, braći, prijateljima, mentoru, profesorima, kolegama s faksa, kao i svim ljudima koji su doprinijeli kako mom intelektualnom rastu, tako i mom rastu kao cjelovite osobe.

Sadržaj

Uvod

U današnje vrijeme svjedoci smo nagloga porasta razmjene podataka. Naglim napretkom današnjih računala, javila se potreba za povećanjem sigurnosti, odnosno zaštite podataka, koji putuju preko komunikacijskog kanala. Današnji kriptosustavi omogućuju siguran prijenos takvih podataka, a ključ njihovog razbijanja zapravo leži u faktoriziranju nekog *velikog* broja (na primjer RSA kriptosustav). Na današnjim računalima, takav problem nije lako riješiv - pa su ti sustavi još uvijek sigurni. Razvojem novih teoretskih modela računala - koji se pokušavaju i u praksi realizirati - uočeno je da problem faktorizacije neće biti više takav problem. Primjer jednog takvog računala je kvantno računalo za kojeg postoji algoritam (*Shorov algoritam*) koji faktorizira broj u polinomnom vremenu. Time se javila potreba za osmišljanjem novih teoretskih modela računala - odnosno kriptosustava - koji bi bili otporni na kvantno izračunavanje - ne bi se mogli probiti uporabom kvantnog računala u nekom razumnom vremenu. Takve kriptosustave ćemo zvati *kvantno rezistentnima*. Tema ovog diplomskog rada biti će DNA kriptografija. Kratko rečeno, radi se o teoretskom modelu kriptografskog sustava koji pomoću DNA izračunavanja šifrira podatke. Prednost takvog sustava jest upravo što je kvantno rezistentan.

U ovom radu najprije ćemo se ukratko upoznati s pojmom DNA računala, odnosno DNA izračunavanja, složenosti DNA računala te algoritmom za enkripciju, odnosno dekripciju podataka pomoću DNA računala.

Poglavlje 1

DNA računalo

1.1 DNA izračunljivost

DNA stroj, kao ni DNA izračunavanje nećemo striktno definirati već će definicija biti opisna - u definiciji ćemo reći koje operacije DNA stroj može izvršavati, i što pri tome mora biti zadovoljeno.
Prije nego definiramo DNA stroj moramo definirati neke pojmove iz logike sudova i kombinatorike.

Definicija 1.1.1. ***Alfabet*** *je proizvoljan konačan skup, čije elemente nazivamo* ***simboli****.*
Neka je $n \in \mathbb{N}$ *proizvoljan te A proizvoljan alfabet, proizvoljni element* $w \in A^n$ *zovemo* ***riječ alfabeta A****. Neka su* $s_1, ..., s_n \in A$*, riječ* $w = (s_1, ..., s_n)$ *alfabeta A još zapisujemo kao* $w = s_1 s_2 ... s_n$*. Smatramo da postoji riječ alfabeta A, koju ćemo označavati s* ε*, koja se ne sastoji ni od jednog simbola i zovemo je* ***prazna riječ****. Po dogovoru smatramo da je* $A^0 = \{\varepsilon\}$*. Skup svih riječi alfabeta A označavamo sa* A^**. Neka su* $a = a_1 ... a_m$*, te,* $b = b_1 ... b_k \in A^*$*, kažemo da je riječ* $c \in A^*$ *nastala* ***konkatenacijom*** *riječi a i b ako vrijedi* $c = ab = a_1 ... a_m b_1 ... b_k$*. Kažemo da je riječ* $c \in A^*$ *podriječ riječi* $a \in A^*$*, ako postoje riječi*

$b, d \in A^*$ *tako da je* $a = bcd$. ***Duljina riječi*** *se definira kao funkcija* $d : A^* \to \mathbb{N}$ *sa:*

$$d(\varepsilon) := 0$$
$$d(wa) := d(w) + 1$$

Definicija 1.1.2. *Neka je S proizvoljan konačan skup, a* $m : S \to \mathbb{N}$ *proizvoljna funkcija.* ***Multiskup M na skupu S*** *je uređeni par* $M = (S, m)$. *Za proizvoljan* $x \in S$, $m(x)$ *zovemo* ***kratnost*** *od x.* ***Kardinalnost multiskupa M*** *(broj elemenata), u oznaci* $|M|$, *se definira kao:*

$$|M| := \sum_{x \in S} m(x)$$

Definicija 1.1.3. ***DNA lanac*** *je proizvoljna riječ alfabeta {A (adenin), G (gvanin), T(timin), C (citozin)}. DNA stroj se sastoji od konstantnog broja konačnih skupova koje nazivamo* ***epruvete****, a čiji su elementi DNA lanci. Za proizvoljnu epruvetu K DNA stroja definiramo multiskup* $MulS(K)$ *kao multiskup svih riječi koje predstavljaju DNA lance sadržane u epruveti K. U DNA stroju su definirane slijedeće instrukcije:*

- *Kopiraj(*K_1,K_2*)* $\to$ *uz pretpostavku da je* $K_2 = \emptyset$ *, kopira MulS(*K_1*) u MulS(*K_2*) time više* K_2 *nije prazan*
- *Spoji(*K_1, K_2, K*)* $\to$ *uz pretpostavku da* $K=\emptyset$*:*

$$MulS(K) = MulS(K_1) \bigcup MulS(K_2)$$

- *Uoči(K)* $\to$ *ispituje je li* $MulS(K) \neq \emptyset$*, ako je rezultat operacije je* $\top$*, inače* $\bot$*. Također se može pročitati sadržaj epruvete* $MulS(K)$*.*
- *Odvoji(K,w)* $\to$ *za skup K i riječ w (iz* $MulS(K)$*) izbacuje sve riječi iz K koje kao podriječ ne sadrže riječ w*

- *Izvadi(K,w)* $\rightarrow K \backslash Odvoji(K,w)$ *→izbacuje sve riječi iz K koje sadrže w*
- *Odvoji_Pref(K,w)* $\rightarrow$ *izbacuje sve riječi iz K koje ne sadrže w kao prefiks*
- *Odvoji_Suff(K,w)* $\rightarrow$ *izbacuje sve riječi iz K koje ne sadrže w kao sufiks*
- *Proširi(K)* $\rightarrow$ *multiskupu MulS(K) još jednom dodaje elemente od K*
- *Izdvoji_po_duljini(K, l)* $\rightarrow$ *iz K izbacuje sve riječi čija je duljina različita od l*
- *Konkatenacija(K)* $\rightarrow$ *na slučajan način izvodi operaciju konkatenacije nad riječima iz MulS(K) tako da duljina novonastalih riječi ne bude veća od neke konstante, a vraća multiskup koji sadrži sve riječi nastale tom konkatenacijom. Vjerojatnost nastajanja duljih riječi je veća. Ukoliko MulS(K) prije izvođenja ove operacije nad epruvetom K sadrži veliki broj kopija svake od riječi, tada će MulS(K) nakon izvođenja ove operacije nad epruvetom K sadržavati sve moguće kombinacije elemenata iz K.*
  - ***Biološki komplement*** *DNA lanca H definiramo kao DNA lanac koja ima jednako znakova kao i H, ali je svaki znak A zamijenjen znakom T, a svaki znak C znakom G i obratno, i označavamo je s* $\overline{H}$
  - *Neka riječ H ima duljinu* $n \in 2\mathbb{N}$*, tada definiramo* ***biološki prefiks*** *riječi H kao biološki komplement riječi sastavljene od prvih* $\frac{n}{2}$ *znakova iz H, slično definiramo i* ***biološki sufiks*** *riječi H kao biološki komplement riječi sastavljene od zadnjih* $\frac{n}{2}$ *znakova riječi H*
  - *Smatramo da je operacija konkatenacije nad riječima H i J dopuštena ako postoji riječ L takva da je biološki sufiks od H prvih* $\frac{n}{2}$ *znakova od L, a biološki prefiks od J prvih* $\frac{n}{2}$ *znakova od L*
- *Izreži(K)* $\rightarrow$ *na slučajan način "skraćuje" riječi iz MulS(K) do neke fiksne duljine*

- *Izaberi(K)* $\rightarrow$ *na slučajan način iz MulS(K) izabire neku riječ te "generira" novi skup sastavljen od samo te riječi*

Program za DNA stroj *definiramo kao konačan niz gornje navedenih instrukcija. U svakom koraku programa se može izvesti točno jedna instrukcija. Kažemo da program P za DNA stroj* ***izračunava*** *funkciju* $f : S \subseteq N^k \rightarrow \mathbb{N}$ *ako vrijedi:*
$\vec{x} = (x_1, ..., x_k) \in S$ *ako i samo ako program P za DNA stroj s ulazom* $\vec{x}$ *(reprezentiranim pomoću DNA lanaca) u epruveti K završi s izvršavanjem te na kraju izvršavanja vrijedi Uoči(K)=* $\top$ *te je pri tome* $K = \{f(\vec{x})\}$.
Kažemo da je funkcija $f : \mathbb{N} \rightarrow \mathbb{N}$ ***DNA izračunljiva*** *ako postoji program za DNA stroj koji ju izračunava.*

Napomena 1.1.4. *Vidimo da se sve ove operacije izvršavaju nad jednom epruvetom u jednom koraku, odnosno multiskupom* $MulS(K)$*. Što je veća kardinalnost multiskupa* $MulS(K)$*, to se više operacija na riječima izvrši istovremeno, a u stvarnom svijetu sve te operacije imaju svoje "biokemijske analogone" - biokemijske reakcije. Takvo računalo zapravo možemo interpretirati kao superračunalo s izuzetno velikim brojem procesora. U pozadini svega toga se zapravo krije masivni paralelizam. Memoriju DNA računala zapravo predstavljaju epruvete. Jasno je odakle naziv epruvete.*
Uočimoo da je $MulS(K)$ *definiran nad konačnim skupom pa je i on konačan - no vidimo da se on zapravo može proširiti nizom operacija Proširi tako da je njegov kardinalitet izrazito velikog reda (nadeksponencijalnog) , ali u praksi se već sada zaključuje da to neće biti uvijek moguće - naime broj DNA lanaca u epruveti (laboratorijskoj) biti će ograničen volumenom te epruvete.*
Uočimo da operacija konkatenacije uključuje vjerojatnosni efekt - vjerojatnost nastajanja duljih riječi konkatenacijom je veća - odnosno dvije riječi iz skupa K koje će se konkatenirati neće biti izabrane na slučajan način - već tako da se pokuša dobiti riječ maksimalne duljine (maksimalna duljina je određena nekom konstantom). Iz toga očito možemo vidjeti

da sam ishod DNA računanja nije sasvim siguran - no u praksi se pokazuje (pri sintezi DNA lanaca) da je to moguće - u tu svrhu je i uvedena pretpostavka da će se, ukoliko $MulS(K)$ sadrži velik broj kopija od svake riječi iz K, dobiti svaka moguća konkatenacija riječi iz K.

1.2 O složenosti DNA računala

Kako bi nešto rekli o složenosti DNA računala, najprije ćemo navesti nekoliko osnovnih definicija iz teorije složenosti algoritama, odnosno referencirati se na [4].

Definicija 1.2.1. ***Turingov stroj*** *je uređena sedmorka $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, gdje je redom:*

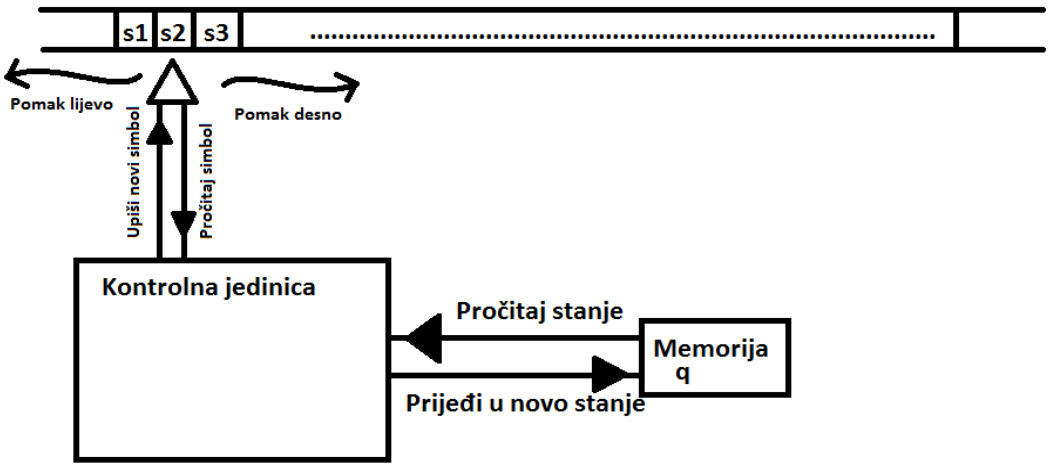
- *Q konačan skup čije elemente nazivamo stanja*
- *Σ je konačan skup, čije elemente nazivamo ulazni simboli, pretpostavljamo da Σ ne sadrži "prazan simbol" kojeg označavamo sa ε*
- *Γ je konačan skup kojeg nazivamo alfabet Turingovog stroja, pretpostavljamo da je $\varepsilon \in \Gamma$, te $\Sigma \subset \Gamma$*
- *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ koju nazivamo funkcija prijelaza*
- *$q_0 \in Q$ nazivamo početnim stanjem*
- *$q_{DA} \in Q$ nazivamo stanjem prihvaćanja*
- *$q_{NE} \in Q$ nazivamo stanjem odbijanja, te $q_{NE} \neq q_{DA}$*

Napomena 1.2.2. *(Opis rada Turingovog stroja)*
Turingov stroj zapravo ima četiri glavna dijela: kontrolnu jedinicu (koja zapravo oponaša dijelovanje funkcije δ), beskonačnu traku, neograničenu s lijeve i desne strane, takvu da se

u svakom trenutku rada stroja na jednom registru trake nalazi točno jedan simbol, memoriju u kojoj se pamti trenutačno stanje stroja te glavu za čitanje koja se u jednom koraku rada stroja može pomicati za točno jedno mjesto na traci: desno, lijevo ili ostati na istom simbolu. Glava se na početku nalazi na nekom mjestu na traci (unaprijed definiranom), zatim čita simbol. Pročitani simbol, u paru s trenutnim stanjem stroja "se šalje" u kontrolnu jedinicu. Glava nakon toga, najprije zamijeni pročitani simbol nekim drugim simbolom, stroj prelazi u novo stanje, a glava se pomiče na drugi registar (L (lijevo), D (desno)) ili ostaje na istom mjestu (S).

Vidimo da opisani Turingov stroj može stati u dva završna stanja q_{DA}, odnosno q_{NE}, takav Turingov stroj se naziva još odlučitelj. Uočimo da Turingov stroj ne mora nužno uvijek stati. Shematski prikaz Turingovog stroja možete vidjeti na slici 1.1.

Nedeterministički Turingov stroj se definira na analogan način, samo što je funkcija prijelaza definirana sa: $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D, S\})$.



Slika 1.1: Shematski prikaz Turingovog stroja

Definicija 1.2.3. *Neka su* $f, g : \mathbb{N} \to \mathbb{R}^+$ *dvije funkcije. Kažemo da je funkcija g* ***asimp-***

totska gornja međa *za funkciju f ako postoje $c > 0$ i $n_0 \in \mathbb{N}$ tako da za svaki $n \geq n_0$ vrijedi*

$$f(n) \leq cg(n)$$

Činjenicu da je g asimptotska međa od f označavamo sa $f(n) = O(g(n)$.

Osnovne definicije (što je alfabet logike sudova, interpretacija, ispunjivost formule, konjunktivna, odnosno, disjunktivna normalna forma i tako dalje) se mogu naći u [5, s. 12-25].
Više o Turingovom stroju te nekim pojmovima na koje se pozivamo u idućim rezultatima se mogu naći u:

- Turing prepoznatiljivost, Turing odlučivost [4, s. 141-142]
- Vremenska složenost Turingovog stroja determinističkog se može naći u [4, s. 248], a nedeterminističkog u [4, s. 255]
- Klase vremenske složenosti:
  - $TIME(f(n))$ u [4, s. 251]
  - P u [4, s. 258]
  - Vezano uz klasu NP u [4, s. 265-267]
- Polinomna reducibilnost u [4, s. 272]
- NP-potpunost u [4, s. 276]

Označimo sa SAT skup definiran na idući način:

$$SAT = \{F : F \text{ je ispunjiva formula logike sudova } \}$$

Formulacija *problema SAT* glasi:

Za danu formulu logike sudova F koja je u konjunktivnoj normalnoj formi odrediti vrijedi li $F \in SAT$.

Konjunktivnu normalnu formu koja u svakoj svojoj elementarnoj disjunkciji sadrži točno $k \in \mathbb{N}\backslash\{0\}$ literala nazivamo k-knf. Formulacija problema $k - SAT$ glasi:

Za proizvoljnu formulu F koja je k-knf odrediti je li F ispunjiva.

U [4, s. 276-283] se može vidjeti da je problem SAT *NP-potpun* problem, kao i $3 - SAT$. Sljedeći teorem govori zapravo o tome da DNA računala, u pogledu vremenske složenosti, imaju bolja svojstva nego deterministički Turnigovi strojevi:

Teorem 1.2.4. *(Lipton) Za svaku konjunktivnu normalnu formu F u kojoj se pojavljuje n propozicionalih varijabli i m klauzula, u $O(m+1)$ separacija i $O(m)$ spajanja te jednim uočavanjem možemo odlučiti vrijedi li $F \in SAT$*

Dokaz navedenog teorema se može naći u [3].
S pogleda odlučivosti jezika ipak nemamo takav rezultat, odnosno postoji slutnja koja kaže:

Slutnja 1.2.5. *(Kvantna i biološka slutnja) Problem je odlučiv na DNA računalu ili kvantnom računalu ako i samo ako je Turing-odlučiv.*

Postavlja se prirodno pitanje kako izračunati složenost DNA računala. Odgovor je jednostavan: složenost DNA računala procijenjujemo brojem instrukcija koje DNA stroj izvrši, te s kardinalosti skupa $MulS(K)$. Zbog toga što kardinalnost skupa $MulS(K)$ može izrazito brzo rasti (samo jedna operacija *Proširi(K)*, za pripadnu funkciju kratnosti m multiskupa $MulS(K)$ vrijedi da je: $m_{nova}(x) = 2 \cdot m_{stara}(x)$, gdje je $m_{nova}(x)$ kratnost od x nakon

izvršenja operacije *Proširi*,a $m_{stara}(x)$ kratnost od x prije izvršenja te iste operacije) prostorna složenost nekog programa za DNA stroj obično doseže nadeksponencijalnu veličinu (vidjet ćemo u idućem podpoglavlju takav slučaj).
U sljedećem potpoglavlju ćemo procijeniti složenost jednog programa za DNA stroj.

Problem Hamiltonovog puta

Definicija 1.2.6. ***Konačan usmjereni graf*** *je uređeni par $G = (V, E)$ gdje je V proizvoljan konačan skup čije elemente nazivamo* ***vrhovi****, a $E \subseteq V \times V$ skup čije elemente nazivamo* ***bridovi****. Ako je $E = V \times V$ kažemo da je usmjereni graf G* ***potpuni graf****. Kažemo da je brid e* ***petlja*** *ako vrijedi: $(\exists x \in V) : e = (x, x)$.* ***Šetnja u grafu*** *G je $2n + 1$-torka $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$, pri čemu vrijedi:*

- $(\forall i \in \{0, ..., n\}) \quad v_i \in V$
- $(\forall i \in \{0, ..., n-1\}) \quad e_i \in E$
- $e_i = (v_i, v_{i+1}), \forall i \in \{0, ..., n-1\}$

Kažemo da šetnja $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$ ***prolazi kroz vrh*** *$x \in V$ ako postoji $i \in \{0, ..., n\}$ takav da je $x = v_i$, te da šetnja* ***počinje*** *s vrhom v_0 i* ***završava*** *s vrhom v_n. Duljina šetnje se definira kao broj bridova koji se pojavljuju u njoj.*
Staza *u grafu je šetnja $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$ za koju vrijedi*

$$(\forall i, j \in \{0, ..., n-1\})\,(i \neq j) \rightarrow e_i \neq e_j$$

Put *u grafu je šetnja $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$ za koju vrijedi:*

$$(\forall i, j \in \{0, ..., n\})\,(i \neq j) \rightarrow v_i \neq v_j$$

Hamiltonov put *je put koji prolazi kroz sve vrhove grafa G.*

Napomena 1.2.7. *Neusmjereni graf se definira analogno, ali se skup bridova definira kao:*

$$E \subseteq \{\{x, y\} : x, y \in V\}$$

Također, radi jednostavnosti, pretpostavili smo da između dva vrha $x, y \in V$ može biti najviše dva usmjerena brida i u tom slučaju vrijedi: $(x, y) \in E$ *i* $(y, x) \in E$.

Problem Hamiltonovog puta glasi:

Postoji li u proizvoljnom konačnom grafu $G = (V, E)$ za vrhove $x, y \in V$ Hamiltonov put koji počinje s x, a završava s y.

U [4, s. 286-291] se može vidjeti da je problem Hamiltonovog puta NP-potpun problem. U ovom poglavlju analiziramo *Adlemanov algoritam* koji rješava problem u $O(n \log(n))$ operacija. U kasnijim poglavljima ćemo obrazložiti reprezentaciju podataka pomoću DNA lanaca, za sada ćemo samo reći da su naši podaci reprezentirani lancima parne duljine l. Sada ćemo prezentirat Adlemanov algoriatam za traženje Hamiltonovog puta koji počinje s vrhom v_{in}, a završava s v_{out} u usmjerenom označenom grafu $G = (V, E)$. Ali prije toga ćemo reći nešto o vezi bridova i vrhova. Ako su dani vrhovi A i B i reprezentirani riječima H i J, tada je brid (A, B) reprezentiran riječju koja je nastala konkatenacijom (u smislu operacije nad riječima) biološkog sufiksa riječi H i biološkog prefiksa riječi J. Kako bi mogli razlikovati koji brid povezuje koje vrhove, uviđamo da svaki vrh mora imati jedinstveni prefiks i sufiks, a ne samo jedinstven prikaz jednom riječju. Nakon što smo objasnili vezu između bridova i vrhova moramo najprije "pripremiti" epruvetu za algoritam.

$$K = V \cup E$$

U početku je upravo $MulS(K) = K$.

Adlemanov algoritam:

1. Ulaz: Graf $G = (V, E)$, $|V| = n$, $v_1 = v_{in}$ vrh iz kojeg krećemo, $v_n = v_{out}$ vrh u kojem završavamo, stavi vrhove i bridove u K

2. $\lceil 2n \log_2(n) \rceil$ puta primjeni operaciju *Proširi(K)* tako da dobiješ barem $2^{2n \log_2(n)} = n^{2n}$ kopija svake riječi u $MulS(K)$

3. Primjeni operaciju *Konkatenacija(K)* da dobiješ šetnju u G, tako da duljina šetnje bude manja od n - broj bridova u šetnji može biti manji ili jednak n

4. Primjeni *Odvoji_Pref(K, v_{in})*: izbacujemo one šetnje koje ne počinju vrhom v_{in}

5. *Odvoji_Suff(K, v_{out})* : izbacujemo one šetnje koje ne završavaju s v_{out}

6. Primjeni operaciju *Izdvoji_po _duljini(K, $l \cdot n + l \cdot (n-1)$)* da iz $K(MulS(K))$ izbaciš sve one riječi koje u sebi ne sadrže točno n vrhova i $n-1$ bridova (šetnje čija je duljina točno $n-1$)

7. na v_i primjeni operaciju *Odvoji(K, v_i)* , $\forall i \in \{2, 3, ..., n-1\}$: iz $MulS(K)$ ukloni sve one šetnje u kojima se neki od vrhova ne pojavljuje

8. *Uoči(K)*: postoji li Hamiltonov put

Nama zapravo bridovi u ovom algoritmu, na ovaj način konstruirani daju mogućnost povezivanja dva vrha koja su povezana nekim birdom (u smislu biokemije, igraju ulogu enzima inače se vrhovi ne bi mogli povezati).
Brojimo korake algoritma:

- 2: $\lceil 2n \log_2(n) \rceil$ koraka
- 3-6: Po jedan korak svaka operacija

- 7: $n-2$ koraka
- 8: jedan korak

Ukupno: $\lceil 2n\log_2(n)\rceil + n - 2 + 5 = \lceil 2n\log_2(n)\rceil + n + 3 = O(n\log(n))$ operacija. No, rekli smo da se složenost DNA stroja mjeri i kardinalnošću skupa $MulS(K)$ koji u jednom trenutku sadrži i n^{2n} elemenata. Još je preostalo dokazati da algoritam radi:

Teorem 1.2.8. *Neka je G=(V,E) usmjeren označen graf, te v_{in} i v_{out} elementi iz V, tada Adlemanov algoritam odlučuje postoji li u usmjerenom grafu G=(V,E) Hamiltonov put od v_{in} do v_{out}.*

Dokaz.

- $|V| = n$, $K = V \cup E$ gdje smatramo da je svakom vrhu dodijeljen jedinstven prefiks i sufiks. Neka je minimalni DNA lanac duljine l.
- Definiramo rekurzivno skupove $MulS(K_n)$ odakle ćemo zapravo izvući kako izgleda naš skup $MulS(K)$ nakon primjene operacije *Proširi(K)* $\lceil 2n\log_2(n)\rceil$ puta

$$K_0 = K \rightarrow MulS(K_0) = K_0$$

$$MulS(K_{n+1}) = MulS(K_n) \cup MulS(K_n), n \in \mathbb{N}$$

Nakon ovog koraka, redefiniramo skup MulS(K)

$$MulS(K) = MulS(K_{\lceil 2n\log_2(n)\rceil})$$

Zapravo sada trebamo dokazati da je $2^{\lceil 2n\log_2(n)\rceil}$ dovoljan broj kopija skupa K za kreiranje svih šetnji u grafu:

$$2^{\lceil 2n\log_2(n)\rceil} \geq 2^{2n\log_2(n)} = n^{2n}$$

pa je dovoljno pokazati da je n^{2n} dovoljan broj kopija skupa K od kojih možemo kreirati sve šetnje u grafu. Bez smanjenja općenitosti u tu svrhu možemo pretpostaviti da je G potpuno povezan usmjeren graf (dakle svaki brid je povezan sa svakim u oba smjera). Zašto? Jer ako G nije potpuno povezan onda ima manji broj šetnji od potpuno povezanog usmjerenog grafa.
U tu svrhu definiramo skup A^k:

$$A^k = \{(b_1, ..., b_k) : b_i \in V\}$$

Uvidimo da smo u skupu A^k dozvolili i petlje! Dakle može postojati brid (v_i, v_i). Kada to ne bi dozvolili, na uređenu k-torku bi samo još stavili uvjet da je $b_i \neq b_{i+1}, \quad \forall i \in \{1, ..., k-1\}$. Sada, jer između svaka 2 vrha ima točno 1 brid za svaki smjer koji povezuje te vrhove, vidimo da su sve šetnje duljine $k-1$ jedinstveno određene skupom A^k. Pa su sve šetnje do duljine $n-1$ (jer ćemo tako izabrati operaciju konkatenacije da kreira šetnje duljine ne duže od $n-1$) reprezentirane idućim skupom:

$$\bigcup_{k=1}^{n} A^k$$

Preostalo je dokazati da kardinalnost gornjeg skupa nije veća od n^{2n}. Kardinalnost skupa A^k je lako odrediti. Naime za prvu komponentu uređene k-torke ima n mogućnosti, za 2 isto n, općenito za i-tu komponentu ima n mogućnosti.

$$|A^k| = n^k \rightarrow \left|\bigcup_{k=1}^{n} A^k\right| = \sum_{k=1}^{n} |A^k| = \sum_{k=1}^{n} n^k = \frac{n(n^n-1)}{n-1}$$

$$\frac{n(n^n-1)}{n-1} \leq \frac{n \cdot n^n}{n-1} \leq n \cdot n^n = n^{n+1} \leq n^{2n}$$

- Primjenom operacije *Konkatenacija(K)* dobili smo sve moguće šetnje u grafu G

(spremljene u $MulS(K)$)

- Operacijama *Odvoji_Pref(K, v_{in})* i *Odvoji_Suff(K, v_{out})* iz skupa $MulS(K)$ izbacujemo sve one šetnje koje ne počinju vrhovima v_{in} i v_{out}

- operacijom *Izdvoji_po_duljini(K, $l \cdot n + l \cdot (n-1)$)* uklanjamo sve preostale bridove i one šetnje čija je duljina strogo manja od $n-1$. Sada su ostale šetnje duljine $n-1$, ali to još nisu putevi (a onda ni Hamiltonovi putevi). Kako ima n vrhova, a šetnja je duljine $n-1$, to znači da je u šetnji točno n vrhova kroz koje šetnja prolazi, ako se neki vrh ne nalazi u šetnji, to znači da se neki drugi vrh pojavljuje dva puta. A kako smo već uklonili one šetnje koje ne počinju s v_{in} i ne završavaju s v_{out} jedino je preostalo ukloniti sve one šetnje koje ne sadrže neki $v_i \in V \backslash \{v_{in}, v_{out}\}$.

- Za svaki $x \in V \backslash \{v_{in}, v_{out}\}$ čini:

$$Odvoji(K, x)$$

- Ovim su korakom zapravo u $MulS(K)$ ostali samo Hamiltonovi putevi koji počinju s v_{in} i završavaju s v_{out}, ako takvih ima, operacijom *Uoči(K)* dobivamo rješenje.

□

Bibliografija

[1] M. Borda. *Fundamentals in Information Theory and Coding*. Springer, 2011.

[2] J. Hromkovic and W. M. Oliva. *Algorithmics for Hard Problems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2002.

[3] R. J. Lipton. DNA solution of hard computational problems. *Science*, 268(5210):542–545, Apr. 1995.

[4] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 2nd edition, 2006.

[5] M. Vuković. *Matematička logika*. Element, 1st edition, 2009.

[6] S. Yan. *Computational Number Theory and Modern Cryptography*. Wiley-HEP information security series. Wiley, 2012.