Acta agriculturae Slovenica, 92(december 2008)2, 151–162.

http://aas.bf.uni-lj.si

Agris category codes: / COBISS Code 1.01

UČINKOVITO KODIRANJE ZAPOREDIJ DNA

Boštjan MUROVEC [a)] in Blaž STRES [b)]

[a)] Univ. v Ljubljani, Fak. za elektrotehniko, Lab. za računalniško integracijo proizvodnje, Tržaška cesta 25, SI-1000 Ljubljana, Slovenija, doc. dr., e-pošta: bostjan.murovec@fe.uni-lj.si.

[b)] Univ. v Ljubljani, Biotehniška Fak., Odd. za zootehniko, Groblje 3, SI-1230 Domžale, Slovenija, as. dr., e-pošta: blaz.stres@bfro.uni-lj.si.

Delo je prispelo 23. januarja 2008, sprejeto 27. februarja 2008.
Received January 23, 2008, accepted February 27, 2008.

IZVLEČEK

V zadnjem obdobju smo priča znatnemu naraščanju uporabe mikroračunalnikov pri raziskavah in analizah zaporedij DNA. Molekule DNA so računalnikom najpogosteje predstavljene v obliki zapisov v formatu FASTA , ki kodirajo sekvence DNA v obliki ASCII niza štirih nukleotidnih oznak A, G, C in T, katerim se po potrebi pridružijo še degenerativne kode in znak za presledek, ko gre za množice med seboj poravnanih zaporedij DNA. Zapis FASTA je dojemljiv za biologa in enostaven za programerja, ki razvija računalniški program, saj si pri razvoju lahko pomaga z bogatim naborom obstoječih knjižnic za delo z znakovnimi polji. Kljub omenjenim prednostim ima zapis FASTA določene slabosti, kot je manj učinkovito iskanje zaporedij nukleotidov, še posebej ob prisotnosti degenerativnih kod. Druga slabost izvira iz dejstva, da vsak posamezni znak FASTA za presledek zasede po en zlog računalniškega pomnilnika, kar je ob prisotnosti velikega števila presledkov neučinkovito in tudi dodatno manjša hitrost iskanja nukleotidnih zaporedij. Zaradi omenjenih slabosti predstavljamo alternativni zapis zaporedij DNA, ki omogoča hitrejše iskanje nukleotidnih zaporedij in učinkovitejše shranjevanje informacij o poravnavi, kar vodi v hitrejše delovanje programov in odpira možnost shranjevanja večjega števila zapisov DNA v delovni pomnilnik računalnika.

Ključne besede: molekularna genetika / bioinformatika / DNK / kodiranje zaporedij

EFFICIENT CODING OF DNA

ABSTRACT

Microcomputers have become ubiquitous tools for DNA research and analysis. Before DNA sequences can be fed into computer programs they need to be suitably coded, which is usually done in a widely accepted FASTA format. According to this scheme, DNA sequence is represented as an ASCII string of four nucleotide characters A, G, C and T, possibly extended with additional codes for representation of degenerated sites, and a character code for FASTA blanks when dealing with aligned DNA sequences. FASTA representation is intuitive for biologists and it eases development of programs since developers can utilize a myriad of available libraries for working with ASCII strings. Despite the mentioned advantages, FASTA format possesses certain drawbacks like inefficient searching for substrings, especially in the presence of degenerative codes. The second disadvantage is inefficient storage of FASTA blank characters, since each such character occupies one byte of memory. Substring searching speed is also negatively affected in the case of excessive number of blanks. Due to the stated drawbacks, we propose an alternative coding of DNA sequences, which enables faster searching of substrings and efficient storage of FASTA blanks, with the result that a greater set of DNA sequences can be held in working memory of a computer and processed faster.

Key words: molecular genetics / bioinformatics / DNA sequences / coding

UVOD

Za mikrobne združbe sta v splošnem značilni visoka gostota celic (10^9/g) in visoka pestrost (10^7 vrst/g) (Gans in sod., 2005). Z razvojem novih tehnologij sekvenciranja (angl. 454 pyrosequencing technology) (Neufeld in sod., 2004; Greena in Kellera, 2006; Roesch in sod., 2007) se je število sekvenc v posamezni klonski knjižnici, narejeni iz vzorcev iz okolja, povzpelo do pred kratkim nedoumljive številke 300.000. S trenutno najbolj uporabljanimi metodami za analizo sekvenc in ugotavljanje filogenetskih odnosov so do nedavnega raziskovalci obdelovali le nekaj sto, redko več tisoč sekvenc naenkrat (Felsenstein, 2006; Ronquist and Huelsenbeck, 2003). Tako sedanje najboljše metode za analizo sekvenc čedalje bolj zaostajajo za tehnologijo pridobivanja podatkov. Zaradi vse večjih količin podatkov lahko pričakujemo, da današnji in bodoči mikroračunalniki ne bodo mogli izvesti želenih analiz v doglednem času. Ker je veliko takih projektov v teku in je posledično možno pričakovati bistveno povečanje števila sekvenc, je poleg novih načinov analiz potrebno tudi izboljšati algoritme iskanja, na osnovi katerih je moč bistveno pospešiti računske operacije.

V podatkovnih bazah so sekvence zapisane v obliki različnih formatov. Med najpogostejšimi so FASTA, EMBL, GCG, GenBank, IG in drugi (Felsenstein, 2006; Felsenstein, 2005; Ronquist, 2004; http://www.genomatix.de/). Format FASTA je splošno razširjen format, v katerem je kodirana večina sekvenc v podatkovnih bazah in ki ga bere večina filogenetskih programov, četudi so izhodne datoteke v drugih formatih (Tamura in sod., 2007; Felsenstein, 1989; Thompson in sod., 1999; Swofford, 2002). Tako upravičeno domnevamo, da je zaradi obsega raziskav in analiz sekvenc format FASTA med najbolj uporabljanimi formati v raziskavah.
Klub razširjenosti formata FASTA, le-ta ni računalniško najbolj učinkovit, zaradi česar tudi zmogljivosti mikroračunalnikov pri analizi zaporedij DNA niso optimalno izrabljene. To narekuje raziskavo možnosti drugačnega kodiranja zapisov, s katerim bi se hitrost analize sekvenc na mikroračunalnikih (tako osebnih računalnikih kot na strežnikih) bistveno povečala.

MATERIALI IN METODE

Opis zapisa FASTA.

Zaporedje DNA je v zapisu FASTA predstavljen kot niz znakov ASCII A, G, C in T, s katerimi zakodiramo zaporedje nukleotidov. V naboru so lahko vsebovani tudi znaki za degenerirana mesta, s katerimi popišemo negotovost pri določanju zaporedja DNA na sekvenatorjih, polimorfizem v primeru degeneriranih začetnih oligonukleotidov ter (ne)selektivnost prepoznavnih mest za rezanje nekaterih restrikcijskih endonukleaz. Za vsako kombinacijo nedoločenosti dveh, treh ali vseh štirih nukleotidov obstaja predpisan znak, s čimer dobimo množico petnajstih (2^4–1) možnih znakov, kot prikazuje pregl. 1.

Nukleotidi so v nekaterih formatih označeni tudi z malimi črkami namesto z velikimi, vendar to za samo branje ni pomembno (Felsenstein, 2004). Velikost črk pri iskanju začetnih oligonukleotidov ali prepoznavnih mest za endonukleaze navadno ignoriramo.

Pri poravnavi dveh ali več zaporedij DNA med seboj potrebujemo tudi znak za presledek '–' (v novejših programih tudi '~') , s katerim označimo po eno vrinjeno mesto. Pri kodiranju prepoznavnih mest za endonukleaze, potrebujemo še znak '^', s katerim določimo mesto prekinitve verige (zaporedja) DNA.

Kot primer si oglejmo naslednji izsek iz bistveno daljše sekvence DNA:

```
5'—...----G---T---A----A---C----G----G---...—3'.
```

Pri nezanesljivem sekvenciranju se pojavijo degenerirana mesta:

```
5'—...----G---T---W----A---C----K----G---...—3'.
```

Preglednica 1. Kode nukleotidov v zapisu FASTA
Table 1. Nucleotide codes according to FASTA format

znak symbol	ime name	pomen meaning
popolnoma določen nukleotid *totally determined nucleotide*		
A	Adenin Adenosine	A
G	Gvanin Guanine	G
C	Citozin Cytosine	C
T	Timin Thymidine	T

znak symbol	ime name	pomen meaning
nedoločenost dveh nukleotidov *indeterminism among two nucleotides*		
Y	Pirimidin Pyrimidine	C ali T C or T
R	Purin Purine	A ali G A or G
W	šibek weak	A ali T A or T
S	močan strong	G ali C G or C
K	keto keto	G ali T G or T
M	amino amino	A ali C A or C

znak symbol	ime name	pomen meaning
nedoločenost treh nukleotidov *indeterminisem among three nucleotides*		
D	ni C not C	A, G ali T A, G or T
V	ni T not T	A, G ali C A, G or C
H	ni G not G	A, C ali T A, C or T
B	ni A not A	G, C ali T G, C or T
nedoločenost štirih nukleotidov *indeterminisem among four nucleotides*		
N (X)	neznan unknown	katerikoli any

Glede na pregl. 1 pomeni znak W nedoločenost nukleotidov adenin in timin, medtem ko znak K pomeni negotovost med nukleotidoma timin in gvanin. Primer prepoznavnih mest za endonukleaze v formatu FASTA predstavlja vzorec `CG^CG`, ki poleg zaporedja nukleotidov določa tudi mesto rezanja zaporedja DNA med drugim in tretjim nukleotidom znotraj prepoznavnega mesta za endonukleaze.

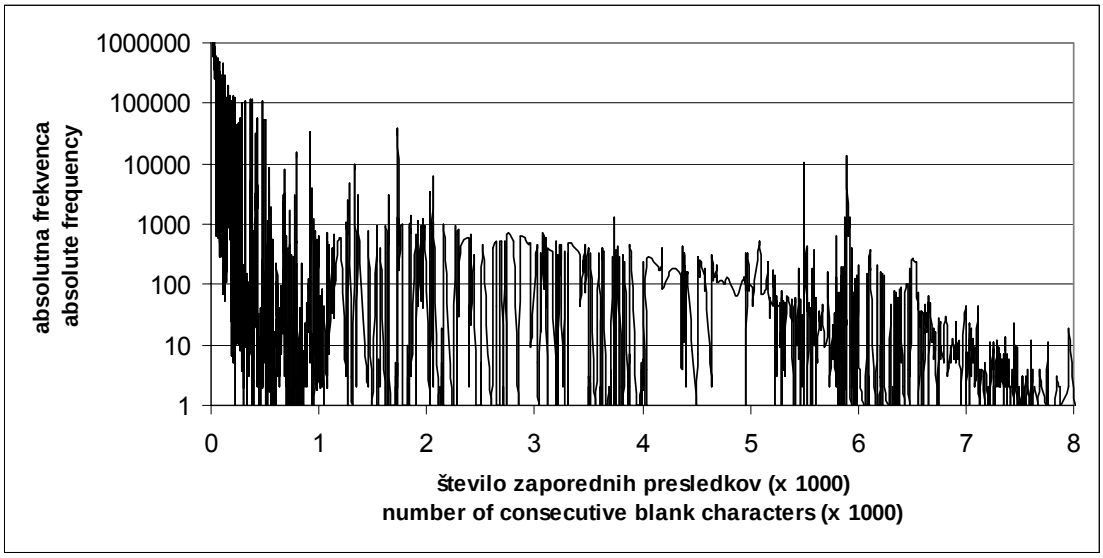
Slabosti zapisa FASTA.

Zapis FASTA kljub prikladnosti na prvi pogled ni najprimernejši s stališča implementacije učinkovitih računalniških postopkov analize vzorcev DNA, kot je iskanje nizov (prepoznavnih mest za endonukleaze, mesta prileganja začetnih oligonukleotidov) v vzorcu. Slabost izvira iz dejstva, da so simboli v pregl. 1 izbrani glede na biološki pomen (informativnost in spremenljivost mest znotraj kodona; nekodirajoča zaporedja) in ne glede na vsebino bitnega vzorca, ki pripada določeni črki, kar bi bolje ustrezalo delovanju računalnikov. Npr. črki G in K (slednja pomeni G ali T; pregl. 1) sta v računalniškem pomnilniku predstavljeni s številkama 71 ($0100\ 0111_2$) in 75 ($0100\ 1101_2$), pri čemer na ravni dvojiškega zapisa simbol G ne označuje podmnožice nukleotidov simbola K, saj enice (ali ničle) kode črke G niso npr. podmnožica istoležnih enic (ali ničel) kode črke K.

Posledica opisanega je, da pri iskanju nizov ne moremo primerjati ujemanja vzorca prepoznavnega mesta za endonukleazo z zaporedjem DNA s preprostimi bitnimi operacijami, ki jih računalniki izvajajo hitro in učinkovito. Namesto tega moramo pri testiranju ujemanja uporabiti kompleksne (sestavljene) pogojne stavke, ki se izvajajo počasneje od preprostih bitnih operacij. Alternativna možnost je uporaba tabele, v kateri so v smislu kartezijevega produkta zajeti vsi možni pari simbolov v pregl. 1, katerim je prirejen indikator ujemanja oziroma neujemanja. Uporaba tabel najprej zahteva izračun lokacije polja na podlagi primerjanih simbolov (črk) in nato dostop do tako določene pomnilniške lokacije v tabeli; noben od teh korakov se niti sam zase ne more izvesti hitreje od preprostih bitnih operacij. S prisotnostjo tako velikih kot malih črk se postopki še neznatno upočasnijo zaradi pretvorbe vseh simbolov v eno velikost črk (alternativno je možno povečati tabelo ujemanja, kar tudi lahko upočasni dostop zaradi slabše izrabe predpomnilnika).

Druga slabost zapisa FASTA pride do izraza ob prisotnosti velikega števila presledkov v zaporedjih DNA, kar je pogost rezultat poravnave sekvenc. Slika 1 prikazuje histogram dolžin zaporednih presledkov v bazi RDP II (http://rdp.cme.msu.edu/), ki je v izdaji 9.50 vsebovala 125 208 sekvenc gena za 16S rRNA, daljših od 1200 baznih parov. Slika je zaradi preglednosti

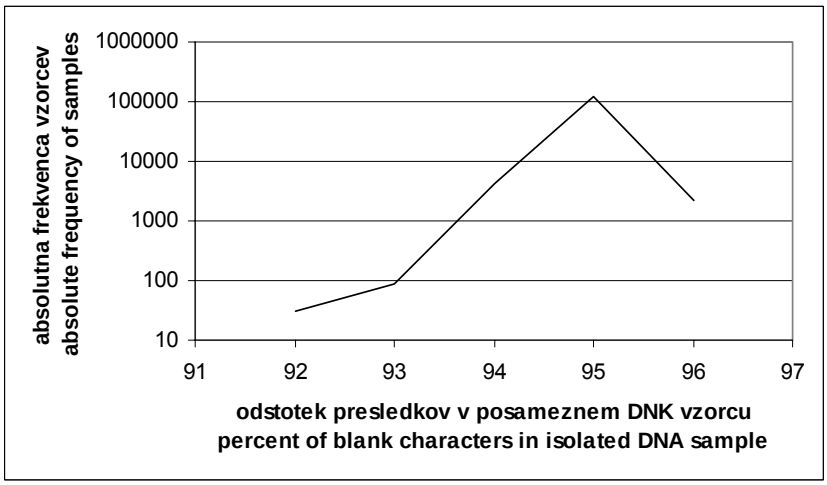
prirezana in vsebuje samo najzanimivejši del dejanskega histograma, ki se po abscisni osi razteza do deset tisoč in po ordinatni osi do petindvajset milijonov (logaritemsko merilo do sto milijonov), vendar je v odstranjenem območju tako malo podatkov, da se na sliki skoraj ne vidijo.



Slika1. Histogram dolžin zaporednih presledkov baze RDP s 125 208 zaporedji (prirezan).
Figure 1. Histogram of consecutive blanks for RDP database with 125 208 sequences. (trimmed).

S slike razberemo, da se najpogosteje pojavljajo zaporedja presledkov do dolžine okvirno 500 znakov, vidimo pa, da velikostni red 6000 zaporednih presledkov in več ni redkost. Najdaljše zaporedje presledkov v obravnavani bazi je dolgo 9922 znakov.

Pri zapisu FASTA se vsak presledek obravnava in shrani v pomnilniku kot ločen znak, zaradi česar lahko presledki zasedejo levji delež delovnega pomnilnika, kar ima za posledico manjšanje števila zaporedij DNA, ki jih lahko hkrati obdelujemo na učinkovit način. Razmere pri obravnavani RDP bazi prikazuje slika 2.



Slika 2. Histogram zaporedij DNA z določenim odstotkom vsebovanih presledkov.
Figure 2. Histogram of DNA sequences with certain percentage of blank characters.

Pri popolnoma vseh 125 208 zaporedjih se vsebovanost presledkov nahaja v intervalu med 92 % in 96 %, pri čemer je daleč najpogostejši delež enak 95 %. Ugotovitev je vredna razmisleka, saj nakazuje, da je izkoriščenost delovnega pomnilnika računalnika zgolj 5 % v primeru, da presledkov ne moremo zavreči. Poleg pomnilniške potratnosti so zaradi presledkov upočasnjeni postopki analize zaporedij. Pri iskanju nizov (prepoznavnih mest za endonukleaze ali mest prileganja začetnih oligonukleotidov) je potrebno vsaj s trivialnim pogojnim stavkom pregledati vse znake zaporedja, čeprav je 95 % le-teh nekoristnih za določitev mesta prileganja.

Opisane slabosti nakazujejo, da bi računalniški postopki analize zaporedij DNA postali učinkovitejši (časovno in pomnilniško) z izbiro drugačnega kodnega zapisa namesto ustaljenega formata FASTA. Tak zapis predstavljamo v nadaljevanju.

Izhodišča za razvoj novega kodnega zapisa

Pri snovanju novega kodnega zapisa zaporedij DNA smo upoštevali naslednja izhodišča. Zapis mora omogočati učinkovito iskanje nizov (mesta prileganja začetnih oligonukleotidov, prepoznavna mesta endonukleaz) s pomočjo preprostih bitnih operacij. Iskanje mora biti učinkovito v obeh smereh preiskovanja DNA verige, kar je pomembno pri lociranju prednjih in zadnjih začetnih oligonukleotidov in prav tako pri iskanju mest rezanja restrikcijskih endonukleaz v obeh smereh. Druga zahteva je kompakten in pomnilniško varčen zapis verige presledkov FASTA, ki bi iskanje nizov čim manj upočasnil. Pri tem je potrebno paziti, da je kljub prisotnosti presledkov še vedno možno izvajati iskanje podnizov v obeh smereh, kar je pri zapisu FASTA v osnovi zagotovljeno, saj je vsak znak za presledek avtonomen in ga lahko enostavno prepoznamo in preskočimo ne glede na smer preiskovanja, ni pa nujno da to velja za vsak kodni zapis.

Učinkovitejše kodiranje nukleotidov.

Omenili smo, da so v računalniškem pomnilniku podatki predstavljeni kot binarna števila in da je izvajanje bitnih operacij nad njimi učinkovito. Danes praktično vsi računalniki operirajo z operandi (registri in pomnilniškimi lokacijami) velikosti osmih bitov (en zlog) ali določenim mnogokratnikom tega števila (16, 32 in 64), zato je smiselno ohraniti lastnost zapisa FASTA, da se posamezna koda nukleotida shrani v ločen zlog pomnilnika, vendar moramo posameznim bitom znotraj zloga prirediti drugačen pomen.

Predlagamo shemo, po kateri od osmih bitov porabimo prve štiri (od 0 do 3) za shranjevanje informacije o nukleotidu, pri čemer je vsakemu nukleotidu prirejen ločen bit, kar prikazuje slika 3.

bit	7	6	5	4	3	2	1	0
pomen **meaning**	presledek blank	rezerviran reserved	degeneriran degenerated	mala črka lower case	Timin Thymidine	Gvanin Guanine	Citozin Cytosine	Adenin Adenosine

Slika 3. Predlagana shema kodiranja nukleotidov.
Figure 3. Recommended nucleotide coding scheme.

Primer. Če zlog predstavlja nukleotid adenin, ima bit 0 vrednost 1, medtem ko imajo ostali biti vrednost 0; binarna koda adenina je torej $0000\ 0001_2$. Podobno ugotovimo, da je koda citozina enaka $0000\ 0010_2$. Gvaninu in timinu pripadata kodi $0000\ 0100_2$ in $0000\ 1000_2$.

Za popis nukleotida porabimo štiri od osmih bitov, zato lahko v preostale štiri bite shranimo dodatne podatke. Bit 4 nosi podatek o velikosti črke, pri čemer njegova vrednost 0 pomeni veliko črko in 1 malo črko. Na ta način smo podatek o velikosti črke povsem ločili od kode nukleotida, zato lahko velikost shranimo, ne da bi ta manjšala učinkovitost iskanja nizov v zaporedju DNA.

V primeru, da zlog hrani podatek o degeneriranem mestu v zaporedju, se priredi vrednost 1 vsem bitom, ki pripadajo možnim nukleotidom, poleg tega se bit 5 nastavi na 1. Po tej shemi FASTA simbolu K, ki pomeni nedoločenost med gvaninom in timinom, pripada binarna koda $0010\ 1100_2$. Na podoben način ugotovimo, da FASTA simbolom Y (C ali T), W (A ali T) in H (A, C ali T) pripadajo kode $0010\ 1010_2$, $0010\ 1001_2$ in $0010\ 1011_2$, medtem ko simbolu N (katerikoli nukleotid) pripada koda $0010\ 1111_2$.

Pomen bita 5 je v tem, da lahko s preverjanjem enega samega bita ugotovimo, ali je nukleotidno mesto degenerirano ali ne. To ponavadi ne pomaga pri iskanju nizov v zaporedju DNA, ampak je koristno v drugih primerih. S pomočjo bita 5 lahko učinkovito preštejemo število degeneriranih mest v zaporedju DNA ali v njegovem določenem odseku (na primer med prednjim in zadnjim začetnim oligonukleotidom), kar je pomembno pri kontroli kakovosti zaporedja DNA. Zapis FASTA tega ne omogoča na preprost način, saj simboli A, C, G, T in degenerativni simboli (pregl. 1) niso ločeni niti glede na binarni zapis niti glede na zaporedje njihovih ASCII kod (npr. tako da bi črke A, B, C in D pomenile štiri osnovne nukleotide in nadaljnje črke degenerirana mesta).

Bit 6 je neizkoriščen in prestavlja možnost za bodoče razširitve kodnega zapisa. Ta bit mora biti nastavljen na nič, sicer kasneje predstavljene operacije ne delujejo pravilno.

Tudi iskane podnize zaporedja DNA (mesta prileganja začetnih oligonukleotidov, prepoznavna mesta endonukleaz) moramo ustrezno kodirati. Pri tem uporabimo enako shemo kot pri kodiranju zaporedja DNA (slika 3), pri čemer bit 5 ne nastavimo na 1 v primeru degeneriranih oziroma neselektivnih mest prileganja. Na primer, če se na določenem mestu začetni oligonukleotid ujema z nukleotidoma A in T, bo to mesto iskanega podniza zakodirano s kodo $0000\ 1001_2$ in ne s kodo $0010\ 1001_2$.. Isto pravilo velja za neselektivnost prepoznavanja mest endonukleaz.

Prav tako morata biti bita 4 in 6 vedno na nič. Zakaj je temu tako, bomo videli pri opisu operacij v nadaljevanju.

Zaradi razlogov, ki jih bomo predstavili kasneje, je včasih (vendar ne vedno) bolje za kodiranje iskanih podnizov uporabiti nekoliko spremenjeno kodno shemo, po kateri bite od 0 do 3 na sliki 3 negiramo (zamenjamo ničle z enicami in obratno). Na ta način bi simbola A in W (nedoločenost med A in T) označili s kodama $0000\ 1110_2$ in $0000\ 0110_2$, simbol N za povsem nedoločen nukleotid, pa je predstavljen s kodo $0000\ 0000_2$. Poudarjamo, da tako kodno shemo uporabljamo samo za kodiranje iskanih nizov in nikoli za kodiranje zaporedja DNA.

Kodiranje presledkov.

V poravnani bazi zaporedij DNA lahko večinski delež kod v formatu FASTA predstavljajo presledki (sliki 2 in 1), zato moramo njihovemu učinkovitemu kodiranju posvetiti posebno pozornost. V ta namen predlagamo kodno shemo, po kateri se v pomnilnik računalnika ne shranjuje vsak posamezni presledek posebej ampak število le-teh v neprekinjenem zaporedju.

Ker moramo zapis števila presledkov ločiti od zapisa nukleotida, porabimo za namen razločevanja en bit zloga, ki je v našem primeru bit 7 (skrajno desni bit na sliki 3). Ko je ta bit nastavljen na vrednost 0, nosi zlog informacijo o nukleotidu, tako kot prikazuje slika 3, v nasprotnem primeru spodnjih sedem bitov (od 0 do 6) nosi število zaporednih presledkov v zapisu DNA. S sedmimi biti lahko zapišemo število v območju med 0 in 127, kar pomeni, da 127 zaporednih presledkov lahko shranimo v en sam zlog, medtem ko bi v zapisu FASTA zanje potrebovali 127 zlogov.

Za primer si oglejmo naslednji izsek zaporedja DNA: A- - -*...80 zaporednih presledkov...*- - - C. V zapisu FASTA bi tak izsek zasedel 82 zlogov pomnilnika, pri čemer bi po en zlog porabili za shranjevanje začetne črke A in končne črke C, med njiju pa bi vrinili 80 znakov za presledek '-'. Po predlagani kodni shemi porabimo za isti odsek samo 3 zloge. Prvi zlog vsebuje binarno

število $0000\,0001_2$, ki predstavlja kodo A (slika 3). Naslednji zlog vsebuje binarno število $1101\,0000_2$, pri čemer bit 7 (skrajno desni) označuje, da gre za niz presledkov in ne za nukleotid, medtem ko preostanek bitov 101 0000 predstavlja v binarnem številskem sistemu zapisano desetiško število 80. Sledi tretji zlog zapisa, ki vsebuje binarno število $0000\,0100_2$ za predstavitev kode C (slika 3). Vidimo, da je pri velikem številu zaporednih presledkov prihranek pomnilnika znaten, saj smo v prikazanem primeru porabili sedemindvajsetkrat (82/3) manj zlogov, kot bi jih pri uporabi kodnega zapisa FASTA.

S slike 1 razberemo, da se v poravnanem zapisu DNA pogosto pojavljajo zaporedja presledkov, ki so bistveno daljša od 127 (do okvirno 10 000), zato se postavi vprašanje, kako jih kodirati. Ena možnost je, da za vsakih 127 zaporednih presledkov porabimo en zlog. Na ta način bi najdaljše zgoraj omenjeno zaporedje 9922 presledkov zakodirali s 79 zlogi, pri čemer bi prvih 78 zlogov vsebovalo binarno število $1111\,1111_2$ (127 presledkov), zadnji zlog pa bi vseboval število $1001\,0000_2$ (16 presledkov), saj je število 9922 enako $78 \times 127 + 16$.

Bistveno boljši izkoristek pomnilnika dosežemo, če binarno število, ki predstavlja dolžino zaporedja presledkov, razbijemo na sedem bitov dolge odseke in te odseke shranimo v zaporedne zloge, ki imajo nastavljen bit 7 na 1. Na primer, število 9922 se zapiše kot $10\,0110\,1100\,0010_2$, torej zanj potrebujemo 14 bitov. Glede na predlagano kodno shemo bi prvih (skrajno desnih) sedem bitov shranili v prvi zlog, ki označuje zaporedje presledkov, medtem ko bi zadnjih sedem (skrajno levih) bitov shranili v naslednji zlog. Zaporedje 9922 presledkov bi bilo tako predstavljeno z zlogoma $1100\,0010_2$ in $1100\,1101_2$, pri čemer pri obeh zlogih skrajno levi bit, nastavljen na 1 indicira zaporedje presledkov in ne nukleotid.

Glede na predlagano shemo zapišemo odsek: `A---`*...9922 zaporednih presledkov...*`---C` s samo štirimi zlogi, od katerih prvi in zadnji zlog kodirata nukleotida A in C, medtem ko srednja dva kodirata 9922 znakov dolgo zaporedje presledkov. Pri zapisu FASTA potrebujemo za isti odsek 9924 zlogov pomnilnika, kar je 2481-krat več.

Po opisani shemi lahko dva zloga shranita največ 16 383 (2^{14}–1) zaporednih presledkov. Če to ni dovolj, lahko trije zlogi shranijo 2 097 151 (2^{21}–1) zaporednih presledkov, kar bi moralo zadostovati za vse potrebe. Nadalje lahko štirje zlogi shranijo absurdnih 268 milijonov (2^{28}–1) zaporednih presledkov.

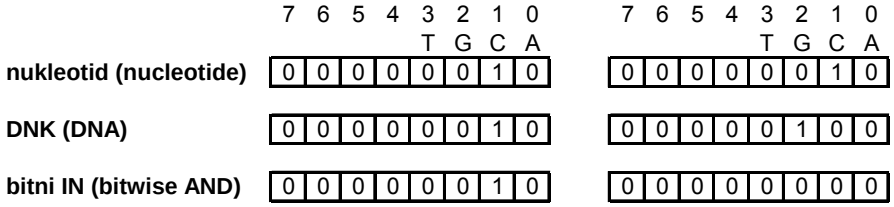
Uporaba predlagane kodne sheme

Smisel vpeljave nove kodne sheme je učinkovito izvajanje postopkov analize zaporedij DNA. Omenili smo že štetje degeneriranih mest v zaporedjih ali določenem odseku le-tega, za kar predlagana kodna shema nudi direktno podporo. Pri štetju mora postopek samo preleteti zaporedje DNA oziroma njegov usterzni odsek in prešteti nukleotidne zloge, pri katerih ima bit 5 vrednost 1 (slika 3). Pri tem moramo paziti, da ne štejemo zlogov, ki kodirajo zaporedje presledkov, zato štejemo zloge, pri katerih je bit 5 enak 1 in hkrati je bit 7 enak 0, kar je še vedno dovolj enostavno za realizacijo. Zapis FASTA ne omogoča tako enostavne izvedbe tega opravila.

Bolj zanimivo od štetja degeneriranih mest je iskanje nizov v zaporedju DNA, sestavni del česar je preverjanje, ali se na določenem mestu nahaja ustrezen nukleotid. Predlagana kodna shema je tako zasnovana, da omogoča hitro preverjanje ustreznosti nukleotida tako v ekzaktnem kot degeneriranem primeru ne glede na to, ali je degenerirana sama DNA veriga (negotovost sekvenciranja), iskani niz (degenerirani začetni oligonukleotid ali neselektivno prepoznavno mesto endonukleaze) ali oboje, kar v zapisu FASTA zahteva relativno zapletene pogojne stavke ali že omenjeno uporabo tabele ujemanja.

Denimo, da želimo preveriti, ali se na določenem mestu verige DNA nahaja nukleotid C. Da dobimo odgovor na vprašanje, izvedemo operacijo bitni IN med kodo nukleotida C in vsebino DNA verige ter pogledamo, ali je rezultat (celoten zlog) različen od nič. Dogajanje prikazuje slika 4, pri čemer je uporabljena kodna shema, pri kateri nukleotidni biti niso degenerirani. Pri

izvajanju operacije bitni IN je posamezni bit rezultata enak 1, če sta istoležna bita obeh operandov enaka 1.



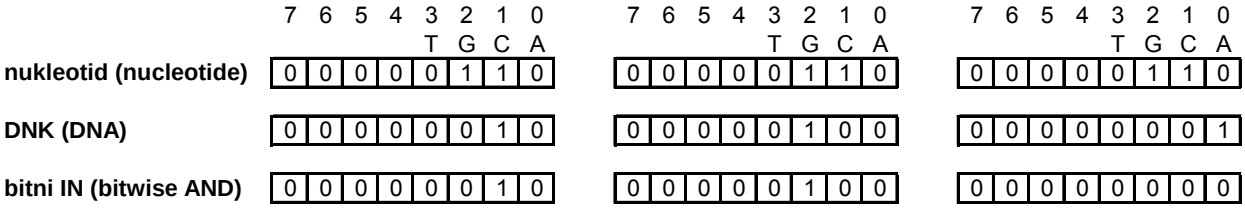
Slika 4. Preverjanje prisotnost nukleotida C (levo: prisoten C, desno: prisoten G).
Figure 4. Checking presence of nucleotide C (left: C is present, right: G is present).

V primeru, da se na mestu preverjanja DNA verige resnično nahaja nukleotid C (slika 4 levo), sta ustrezna bita v kodi nukleotida in v DNA verigi nastavljena na vrednost 1, zato je istoležni bit rezultata operacije bitni IN enak 1 in s tem je celotni zlog rezultata različen od nič, kar nakazuje ujemanje.

Slika 4 desno prikazuje situacijo, ko se na mestu preverjanja DNA verige nahaja nukleotid G. V tem primeru nima noben bit nukleotidove kode vrednosti 1 na istem mestu kot bit DNA verige in rezultat operacije bitni IN je enak nič, kar nakazuje neujemanje.

Primera na sliki 4 jasno ponazorita, zakaj je koristno uporabiti kodno shemo, pri kateri je vsakemu nukleotidu prirejen ločen bit. Bitne operacije spadajo med najhitrejše operacije, ki jih poznajo digitalni računalniki in preverjanje ujemanja verige DNA z vzorcem na ta način je teoretično najhitrejše možno in tudi pomnilniško učinkovito (ne potrebujemo tabele ujemanja).

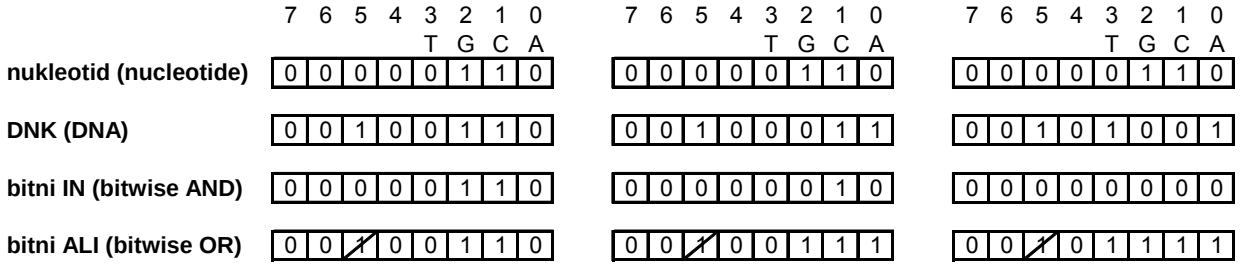
Zgornji primer razširimo tako, da dovolimo ujemanje DNA verige z večimi nukleotidi (npr. v primeru degeneriranega začetnega oligonukleotida), kar prikazuje slika 5; v prikazanem primeru želimo preveriti, ali se na izbranem mestu v DNA verigi nahaja eden od nukleotidov C ali G, čemur ustreza nukleotidna koda, ki ima oba bita C in G nastavljena na 1.



Slika 5. Preverjanje prisotnost nukleotida C ali G (levo: prisoten C, sredina: prisoten G, desno: prisoten A).
Figure 5. Checking presence of nucleotide C or G (left: C is present, middle: G is present, right: A is present).

Levi primer na sliki prikazuje dogajanje ob prisotnosti nukleotida C v zaporedju DNA , kjer je rezultat operacije bitni IN različen od nič, saj sta bita C nastavljena na 1 tako v nukleotidni kodi kot v zaporedju DNA. Analogno dogajanje lahko spremljamo na sredini slike 5, le da tokrat opis velja za bit nukleotida G. Desni primer kaže situacijo ob prisotnosti nukleotida A v zaporedju DNA. Sedaj noben bit nukleotidne kode ni hkrati na vrednosti 1 z istoležnim bitom DNA verige in rezultat operacije bitni IN je enak nič, zato ujemanja ni.

Najsplošnejši primer nastopi, ko imamo degenerirana mesta tako v specifikaciji nukleotida kot v zaporedju DNA. V tem primeru samo ena bitna operacija ne zadostuje oziroma ne daje nujno želenega rezultata. Situacijo prikazuje slika 6, pri čemer bomo zadnjo vrstico uporabili kasneje.



Slika 6. Preverjanje prisotnost nukleotida C ali G (levo: možen C ali G, sredina: možen C ali A, desno: možen T ali A).

Figure 6. Checking presence of nucleotide C or G (left: possible C or G, middle: possible C or A, right: possible T or A).

Levi primer na sliki ponazarja situacijo, kjer se v zaporedju DNA lahko nahaja nukleotid C ali G, zato se mesto ujema s specificirano množico nukleotidov. Rezultat operacije bitni IN je različen od nič (tokrat sta dva bita rezultata različna od nič), kar nakazuje prileganje.

Srednji primer na sliki ponazarja možnost prisotnost nukleotida C ali A v zaporedju DNA. Ker se vsaj en nastavljen bit zaporedja DNA ujema z istoležnim bitom nukleotidne kode, je rezultat različen od nič, kar nakazuje ujemanje. Tak rezultat ni nujno zaželen, saj se v dejanskem zaporedju DNA lahko na tem mestu nahaja nukleotid A namesto nukleotida C.

Desni primer na sliki prikazuje situacijo, kjer je v zaporedju DNA možen nukleotid A ali T. Ker je množica dovoljenih nukleotidov C in D tuja množici možnih nukleotidov A in T, je rezultat operacije bitni IN enak nič in ujemanja zanesljivo ni. Sedaj tudi vidimo, zakaj bita 5 (tudi 4 in 6) ne smemo uporabljati pri kodiranju podnizov; v desnem primeru na sliki 6 bi postopek razglasil ujemanje, saj bi bil bit 5 nastavljen na 1 v obeh operandih, zaradi česar bi imel tako vrednost tudi pripadajoči bit rezultata in celotni rezultat bi bil različen od 0 (ujemanje).

Zaključimo, da je rezultat opisanega testa ujemanja pozitiven, čim je vsaj en možen nukleotid v zaporedju DNA enak vsaj enemu dovoljenemu nukleotidu v iskanem nizu. Tako iskanje je koristno v primeru, da želimo identificirati vsa mesta v DNA verigi, kjer bi se iskani niz lahko nahajal, čeprav obstaja možnost, da se ne.

Navadno si želimo, da bi postopek iskanja niza ugotovil ujemanje samo, če ni nikakršne možnosti napačnega ujemanja, torej če množica možnih nukleotidov v zaporedju DNA predstavlja podmnožico dovoljenih nukleotidov. Da to dosežemo, potrebujemo pri obravnavani kodni shemi vsaj dve bitni operaciji. Postopek ponovno prikazuje slika 6, pri čemer sedaj uporabimo njeno zadnjo vrstico.

Želeni test prileganja dobimo tako, da že opisani postopek nekoliko spremenimo. Namesto operacije bitni IN izvedemo operacijo bitni ALI. Pri tej operaciji je posamezni bit rezultata nastavljen na 1, če je vsaj v enem operandu (ali obeh) ustrezni bit nastavljen na 1. Zaporedje DNA se prilega kodi nukleotida, če je tako dobljeni rezultat enak izhodiščni kodi nukleotida, sicer je možno, da se v DNA verigi nahaja nukleotid, ki ga koda ne dovoli.

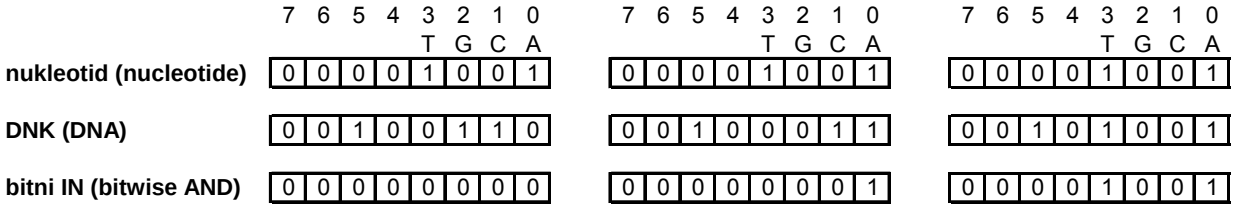
Na sliki 7 levo si ponovno oglejmo primer, ko množica dovoljenih nukleotidov vsebuje C in G. Prav ta nukleotida vsebuje tudi množica možnih nukleotidov v zaporedju DNA. Rezultat operacije bitni ALI je enak kodi nukleotida, zato razglasimo ujemanje. Na sliki vidimo, da bit 5, ki označuje degenerirano mesto, moti, zato ga pred preverjanjem enakosti brezpogojno izbrišemo. Isto bi veljalo za bit 4, ki nakazuje malo črko namesto velike. V splošnem izbrišemo vse štiri zgornje bite, kar je možno izvesti z eno samo dodatno operacijo.

Na srednji sliki je rezultat operacije bitni ALI različen od kode nukleotida, zato je možno, da se v zaporedju DNA nahaja nedovoljen nukleotid in razglasimo neujemanje. Na desni sliki sta

oba možna nukleotida v zaporedju DNA različna od dovoljenih in v tem primeru spet pravilno razglasimo neujemanje.

Na videz smo tudi restriktivnejše iskanje rešili na še vedno eleganten način samo z uporabo bitnih operacij. Kljub temu je predlagani postopek vreden razmisleka, saj poleg osnovne bitne operacije potrebujemo še pomožno (brisanje zgornjih štirih bitov). Poleg tega je za določitev ujemanja potrebno rezultat primerjati z vrednostjo nukleotida, torej ne z vrednostjo nič, za kar so računalniki posebej optimirani. Zaključimo, da se na ta način test ujemanja izvrši okvirno trikrat počasneje od izvedbe ene same binarne operacije.

Situacijo lahko izboljšamo z uporabo že omenjene kodne sheme, kjer nukleotide iskanega podniza zakodiramo negirano. To interpretiramo, kot da biti kode iskanega podniza, ki so enaki 1, pomenijo nukleotide, ki jih v DNA verigi ne sme biti na pripadajočem mestu. Ujemanje DNA z iskanim nizom testiramo tako, da izvedemo operacijo bitni IN med obema kodama in pogledamo, ali je rezultat enak nič. Situacijo prikazuje slika 7.



Slika 7. Ekzaktno preverjanje prisotnost nukleotida C ali G z negiranim zapisom iskanega niza (levo: možen C ali G, sredina: možen C ali A, desno: možen T ali A).

Figure 7. Exact checking of presence of nucleotide C or G with negated nucleotide code of searched substring (left: possible C or G, middle: possible C or A, right: possible T or A).

Levi primer na sliki ponovno prikazuje situacijo, kjer se v zaporedju DNA lahko nahaja nukleotid C ali G; ta pogoj je ekvivalenten pogoju, da se na tem mestu ne sme nahajati niti nukleotid A niti nukleotid T. Pogoj je prikazan tako, da sta bita, ki pripadata nukleotidoma A in T, nastavljena na 1. Rezultat operacije bitni IN je enak nič, kar pomeni, da noben od možnih nukleotidov v zaporedju DNA ni vsebovan v množici prepovedanih nukleotidov iskanega (ujemanje).

Srednji primer na sliki prikazuje situacijo, ko se v zaporedju DNA lahko nahaja nukleotid C ali A. Rezultat operacije bitni IN, ki je različen od nič, nakazuje, da je možno prepovedano stanje, zato razglasimo neujemanje.

V desnem primeru na sliki sta dva možna nukleotida v zaporedju DNA vsebovana v množici prepovedanih nukleotidov, zato sta dva bita rezultata različna od nič, s čimer je tudi celotni rezultat različen od nič in s tem je neujemanje pravilno detektirano.

Z opisano kodno shemo smo tudi restriktivno testiranje ujemanja uspeli realizirati z eno samo bitno operacijo, kar pomeni, da je tak test teoretično najhitrejši možen.

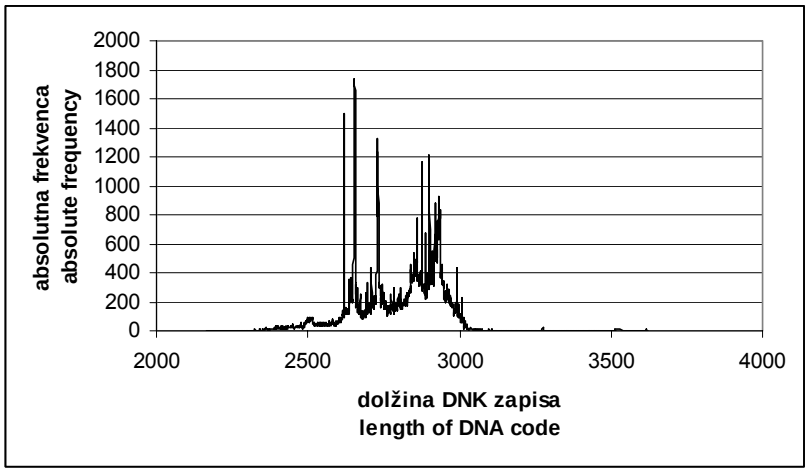
REZULTATI IN RAZPRAVA

Predhodno podani opis nove kodne sheme sekvenc DNA nakazuje, da le-ta odpravlja določene slabosti zapisa FASTA. Na tem mestu nas zanima, kaj in koliko pridobimo z vpeljavo predlagane sheme v računalniške programe. Primerjavo naredimo s pomočjo predhodno omenjene baze RDP II (9.50).

Pomnilniško učinkovitost ocenimo tako, da primerjamo velikost pomnilnika, ki ga v povprečju potrebujemo za shranjevanje DNA v obeh zapisih. Ker je baza RDP II (9.50)

poravnana, imajo vsi zapisi DNA enako dolžino, in sicer 27 934 znakov (skupaj nukleotidnih kod in presledkov); to je tudi število zlogov pomnilnika, ki jih potrebujemo za pomnenje enega zaporedja DNA v primeru uporabe kodne sheme FASTA.

Ker naša kodna shema shranjuje presledke učinkoviteje, pričakujemo znatno zmanjšanje potrebnega pomnilniškega prostora, kar potrjuje histogram dolžin zapisa DNA na sliki 8.



Slika 8. Histogram dolžin zapisa zaporedij DNA v bazi RDP II (9.50).
Figure 8. Histogram of length of DNA sequence code in database RDP II (9.50).

Natančno izračunana povprečna dolžina zapisa je enaka 2798 zlogov (standardni odklon je 138 zlogov). To pomeni, da lahko s predlagano kodno shemo v isti pomnilniški prostor shranimo skoraj desetkrat več zaporedij DNA, kot pri zapisu FASTA.

Za primerjavo hitrosti procesiranja v primeru uporabe obeh kodnih shem smo vsakemu zaporedju DNA v bazi RDP II (9.50) poskusili določiti lokacijo šestih parov začetnih oligonukleotidov (vse možne kombinacije med prednjima začetnima oligonukleotidoma 968f in 341f in zadnjimi začetnimi oligonukleotidi 1492r, 1062r in 926r) (http://www.microbial-ecology.net/probebase/). Za vsak par začetnih oligonukleotidov, ki se je prilegal zaporedju DNA, smo poskusili locirati prvo in zadnje mesto rezanja endonukleaz AbaI, AbsI in AccII (http://rebase.neb.com). Tako smo za vsako zaporedje DNA potencialno poskusili locirati osemnajst različnih kombinacij oligonukleotidov in encimov, pri čemer smo encim iskali dvakrat (enkrat za prednjim in drugič pred zadnjim začetnim oligonukleotidom). Nadalje smo pri vsaki od lociranih kombinacij prešteli število vsake nukleotidne kode v segmentu DNA od začetka prednjega začetnega nukleotida do najbližjega mesta rezanja endonukleaze ter podobno od zadnjega mesta rezanja endonukleaze do konca zadnjega začetnega oligonukleotida; skupno smo imeli šestnajst števcev: enega za vsoto vseh nukleotidov (dolžino segmenta) in ostalih petnajst za posamezno nukleotidno kodo v pregl. 1. Test je potekal na prenosnem računalniku s procesorjem Intel (R) Celeron (R) M 1,5 GHz in delovnim pomnilnikom (RAM) 512 MB.

Izvršni čas analize zaporedij DNA pri uporabi kodne sheme FASTA znašal 387,1 s, medtem ko je pri uporabi naše kodne sheme znašal 42,5 s, kar pomeni 9,1-krat večjo hitrost delovanja pri uporabi naše kodne sheme. Poudarjamo, da je to zgolj čas obdelave podatkov, v katerega ni vključen čas branja zaporedij DNA z diska, ki nikakor ni zanemarljiv, saj baza RDP II (9.50) zavzame 3,4 GB prostora na disku. Zato že samo branje datoteke in preverjanje pravilnosti zapisa zaporedij DNA traja več minut in pri manjših obdelavah podatkov kodna shema ne igra nobene vloge pri celotnem času izvajanja, saj ozko grlo predstavlja strojna oprema. V tej luči predhodno prikazani rezultati hitrostnega testa nakazujejo, asimptotično pohitritev, ki jo lahko pričakujemo z vpeljavo naše kodne sheme pri obsežnih obdelavah zaporedij DNA.

Navedimo še test celotnega izvršnega časa, ki je bil potreben za izvedbo testa: 699,2 s v primeru kodiranja FASTA in 310,3 s pri uporabi naše kodne sheme, kar pomeni 2,25-krat hitrejše celotno izvajanje z uporabo naše kodne sheme. Pri obsežnejših analizah (npr. pri lociranju mest prileganja večjega števila parov začetnih oligonukleotidov in prepoznavnih mest rezanja endonukleaz) pričakujemo, da bi se to razmerje asimptotično približevalo predhodno podanemu razmerju 9,1, saj bi bilo potrebno zaporedja DNA prebrati z diska samo enkrat (fiksni del časa), nadaljnja analiza pa bi se vršila s hitrostjo, ki jo določa izbrana kodna shema.

SKLEPI

Članek opisuje novo kodno shemo za kodiranje zaporedij DNA, ki je bila razvita z namenom odpraviti glavne slabosti uveljavljene kodne sheme FASTA. Empirični testi kažejo, da predlagani pristop omogoča okvirno desetkrat boljšo izrabo delovnega pomnilnika računalnika in asimptotično omogoča več kot devetkrat hitrejše delovanje v članku preizkušenih postopkov analize zaporedij DNA (iskanje podnizov in štetje nukleotidov v določenih podsegmentih zaporedij DNA).

SUMMARY

The article presents new DNA coding scheme, which was developed as a worthy replacement for the well-established FASTA coding scheme. Empirical tests show that our proposition utilizes working memory of computer roughly ten times better than the original FASTA scheme. In addition, our scheme asymptotically increases the speed of DNA analysis for more than nine times on tested algorithms (searching for substring and counting of nucleotides in DNA segments).

VIRI

Felsenstein, J. PHYLIP – Phylogeny Inference Package (Version 3.2). Cladistics 5(1989), 164–166.

Felsenstein, J. PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle, 2005.

Felsenstein, J. Inferring Phylogenies. 2004, Sinauer Associates.

Greena, B.D. / Kellera, M. Capturing the uncultivated majority. Current Opinion in Biotechnology, 17(2006), 236–240.

Gans, J./ Wolinsky, M./ Dunbar, J. Computational Improvements Reveal Great Bacterial diversity and High Metal Toxicity in Soil. Science, 309(2005), 1387–1390.

Neufeld, J.D./ Yu, Z./ Lam, W./ Mohn, W.W. Serial analysis of ribosomal sequence tags (SARST): a high-throughput method for profiling complex microbial communities. Environ. Microbiol., 6(2004), 131–144.

Roesch, L.F.W./ Fulthorpe, R.R./ Riva, A./ Casella, G./ Hadwin, A.K.M./ Kent, A.D./ Daroub, S.H./ Camargo, F.A.O./ Farmerie, W.G./ Triplett, E.W. Pyrosequencing enumerates and contrasts soil microbial diversity. The ISME Journal, 1(2007), 283–290.

Ronquist, F./ Huelsenbeck, J.P. MRBAYES 3: Bayesian phylogenetic inference under mixed models. Bioinformatics, 19(2003), 1572–1574.

Ronquist, F. Bayesian inference of character evolution. Trends in Ecology and Evolution, 19(2004), 475–481.

Swofford, D. L. PAUP*: Phylogenetic Analysis Using Parsimony (and Other Methods) 4.0 Beta. Florida State University, 2002.

Tamura, K./ Dudley, J./ Nei, M./ Kumar, S. MEGA4: Molecular Evolutionary Genetics Analysis (MEGA) software version 4.0. Molecular Biology and Evolution, 24(2007), 1596–1599.

Thompson, J.D./ Plewniak, F./ Poch, O. A comprehensive comparison of multiple sequence alignment programs. Nucleic Acids Res., 27(1999), 2682–2690.