Elektrotehnički fakultet Beograd
Genomska informatika (13M111GI)

Aleksandar Kiridžić (2017/3199)

Seed-and-extend Aligner

Beograd, 2018.

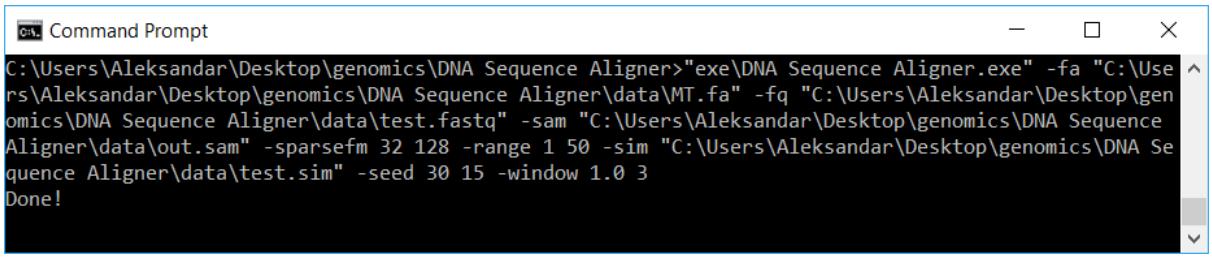
Sadržaj

Uvod

Aligner je alat koji vrši poravnavanje DNK sekvenci sa referentnim sekvencama. Referentna sekvenca se učitava iz FASTA fajla, a *read*-ovi iz FASTQ fajla. Koristi se tehnika *seed-and-extend* koja se sastoji iz dve osnovne faze:

- *seeding* - *Read* se deli na najčešće preklapajuće *seed*-ove za koje se traži *exact match* u referentnoj sekvenci. Na osnovu rezultata *matching*-a biraju se prozori (delovi referentne sekvence). Ovi prozori su favoriti za najbolje poravnanje i samo oni se koriste u narednoj fazi. Parametre *seed*-ova i prozora može postaviti korisnik.
- *extending* – Prozori iz prethodne faze se porede sa *read*-om. Računa se sličnost na osnovu *end-space-free variant* poravnanja. *Extend* koji da najbolji rezultat se uzima kao finalna vrednost poravnanja. Parametre sličnosti (*score matrix)* može postaviti korisnik.

Rezultati poravnavanja se upisuju u fajl u formatu sličnom SAM formatu.

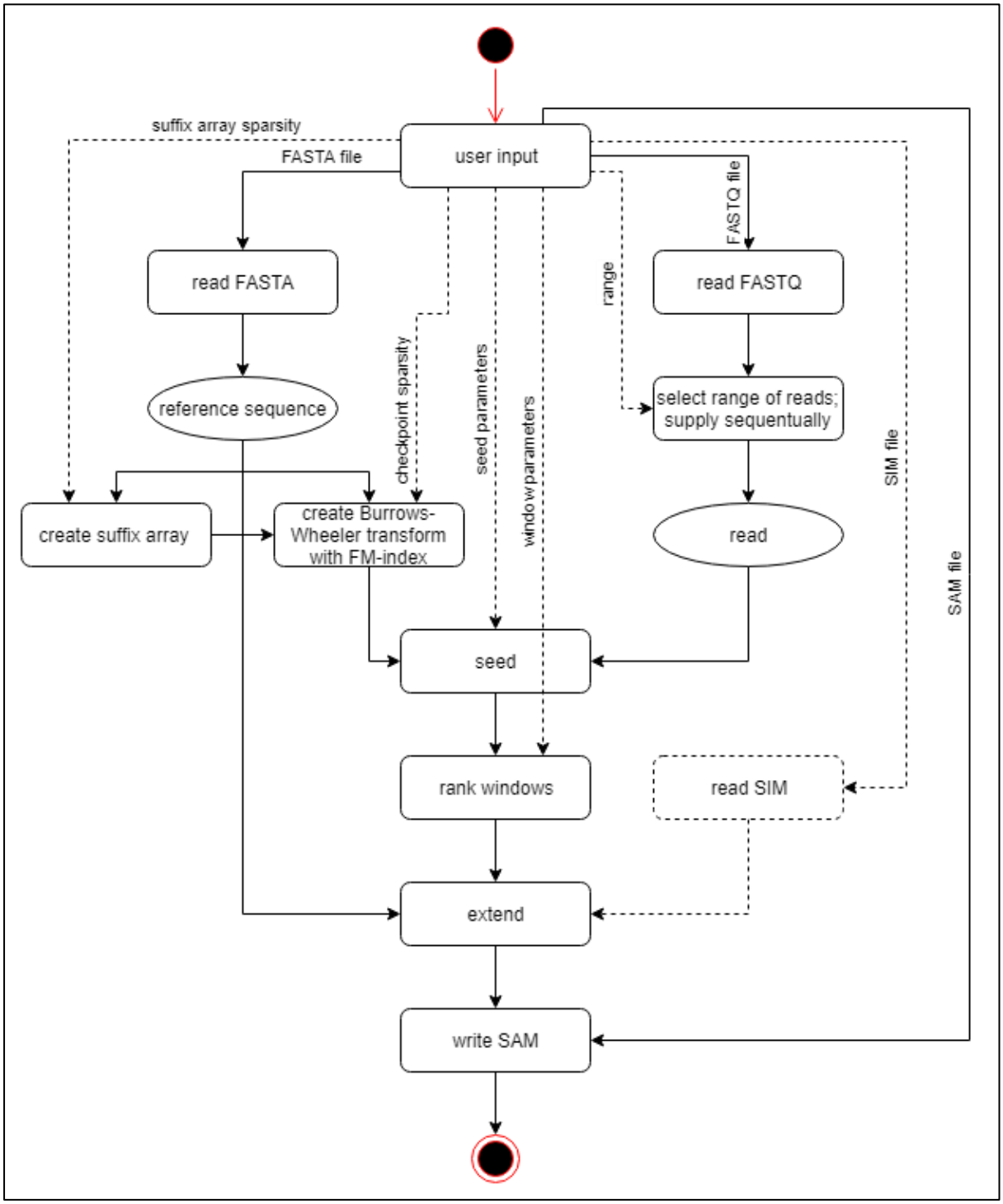


Slika 1. Primer korišćenja alata

read_id	strand	position	score	cigar	read_sequence
ST-E00185:49:H5LVWCCXX:5:1101:3273:34448/2	REV	7565	300	150M	GAAAAGGGCATACAGGACTAGGAAG
ST-E00185:49:H5LVWCCXX:5:1101:4715:41480/2	FOR	5938	300	150M	ACAAAGACATTGGAACACTATACCT
ST-E00185:49:H5LVWCCXX:5:1101:7100:23038/2	REV	6103	298	149M	AGCAGATGCGAGCAGGAGTAGGAGA
ST-E00185:49:H5LVWCCXX:5:1101:7120:42447/2	FOR	12270	298	149M	TTAAAGGATAACAGCTATCCATTGG
ST-E00185:49:H5LVWCCXX:5:1101:8896:5159/2	FOR	14828	284	3M1X40M1X85M1X18M	TCCACATGATGAAACTTCGGCTCAC
ST-E00185:49:H5LVWCCXX:5:1101:10876:12877/2	REV	5520	300	150M	CCCATTGGTCTAGTAAGGGCTTAGC
ST-E00185:49:H5LVWCCXX:5:1101:11474:21790/2	FOR	7533	300	150M	CCATTTCATAACTTTGTCAAAGTTA
ST-E00185:49:H5LVWCCXX:5:1101:11535:3752/2	FOR	13797	280	83M1X23M1X31M2X9M	ACTCACAGCCCTCGCTGTCACTTTC
ST-E00185:49:H5LVWCCXX:5:1101:15331:31213/2	FOR	5574	300	150M	TTTCTGTAACAGCTAAGGACTGCAA
ST-E00185:49:H5LVWCCXX:5:1101:15392:59798/2	REV	12248	298	149M	AAGGATGGGGGGAATTAGGGAAGTC
ST-E00185:49:H5LVWCCXX:5:1101:15555:14863/2	FOR	5712	292	102M1X46M	TCAACTGGCTTCAATCTACTTCTCC
ST-E00185:49:H5LVWCCXX:5:1101:16945:15654/2	FOR	6886	300	150M	ACGGAAGCAATATGAAATGATCTGC
ST-E00185:49:H5LVWCCXX:5:1101:16955:34131/2	FOR	6922	298	149M	GAGCCCTAGGATTCATCTTTCTTTT
ST-E00185:49:H5LVWCCXX:5:1101:21300:71577/2	REV	14590	300	150M	CTTGTAGTTGAAATACAACGATGGT
ST-E00185:49:H5LVWCCXX:5:1101:24060:9659/2	REV	3257	288	6M1X15M1X127M	GTAGTTGTATATAGCCTAGAATTTT
ST-E00185:49:H5LVWCCXX:5:1102:5527:47984/2	REV	14976	298	149M	TGTTGCTATAGTTGCAAGCAGGAGG
ST-E00185:49:H5LVWCCXX:5:1102:9810:47422/2	REV	13435	300	150M	GGTAGCGATGAGAGTAATAGATAGG
ST-E00185:49:H5LVWCCXX:5:1102:10003:66075/2	FOR	2830	300	150M	AGAACCCAACCTCCGAGCAGTACAT
ST-E00185:49:H5LVWCCXX:5:1102:10937:15936/2	FOR	9298	300	150M	TAGCCATGTGATTTCACTTCCACTC
ST-E00185:49:H5LVWCCXX:5:1102:11180:46385/2	FOR	14926	300	150M	AACCGCCTTTTCATCAATCGCCCAC
ST-E00185:49:H5LVWCCXX:5:1102:12185:26255/2	REV	14711	292	37M1X82M1X29M	GCCAAGGAGTGAGCCGAAGTTTCAT

Slika 2. Primer izlaznog fajla

Opis rešenja



Slika 3. Dijagram toka izvršavanja

Na slici 3 je prikazana dijagram toka izvršavanja programa. Pravougaoni elementi predstavljaju procese, a elipsoidni objekte. Prikazani su samo objekti najveće važnosti, referentna sekvenca i pojedinačni *read*. Strlice predstavljaju prosleđivanje podataka između procesa. Labelirane su samo u slučaju da proces proizvodi više raznih objekata, u suprotnom predstavljaju logičan rezultat procesa. Pune linije označavaju obaveznu zavisnost, a isprekidane opcionu.

U nastavku će biti detaljnije opisani svaki od navedenih procesa.

Čitanje FASTA fajla

Putanju do FASTA fajla zadaje korisnik. Podrazumeva se jednostavna struktura fajla. Linija se ignoriše ukoliko je prazna ili predstavlja komentar, što se može prepoznati na osnovu početnog znaka u liniji. Sve ostale linije se ulančavaju u sekvencu. Referenca je uvek *forward strand*.

Čitanje FASTQ fajla

Putanju do FASTQ fajla zadaje korisnik. Podrazumeva se jednostavna struktura fajla. Svaki *read* je definisan sa četiri linije:

1. linija koja počinje znakom '@' i predstavlja identifikator *read*-a;
2. linija koja predstavlja sekvencu;
3. linija koja sadrži samo znak '+';
4. linija koja predstavlja kvalitet *read*-a i mora sadržati isti broj znakova kao druga linija.

Za razliku od referentne sekvence, *read*-ovi iz FASTQ fajla mogu doći sa oba *strand*-a. Zato se odmah izračunava i *reverse complement* pročitane sekvence.

Kreiranje *suffix array*-a

Za referentnu sekvencu se kreira *suffix array*. Ova struktura se prvobitno koristi pri konstrukciji *Burrows-Wheeler transforma*, a posle za određivanje pozicije *exact match*-a u *seeding* fazi.

Koristi se algoritam rangiranja povećavajućih prefiska svakog sufiksa. Za svaki sufiks se definiše rang iz dva dela različitih prioriteta. Inicijalno, viši i niži rang su jednaki prvom i drugom karakteru svakog sufiksa. Pri svakom pristupu van granica stringa, dodeljuje se negativan rang. Zatim se sufiksi sortiraju po rangu rastući.

Definiše se korak, inicajno jednak 2. Iterira se sa dupliranjem koraka dok on ne pređe veličinu stringa. U svakoj iteraciji ažuriraju se vrednosti rangova svih sufiksa po redosledu prethodnog sortiranja. Viši rang prvog sufiksa dobija vrednost 0. Viši rang svakog narednog sufiksa je jednak višem rangu prethodnog ukoliko su im isti rangovi iz prethodne iteracije ili uvećan za 1 ukoliko nisu. Niži rang svakog sufiksa je jednak višem rangu sufiksa koji je se nalazi korak udesno od njega u originalnom stringu. Na kraju iteracije sufiksi se sortiraju po rangu rastući.

Algoritam koristi činjenicu da su u pitanju sufiksi jednog stringa kako bi ubrzao poređenje korišćenjem rezultata prethodnih poređenja.

Pun *suffix array* zauzima *length(string) * sizeof(integer)* memorije što može biti mnogo u slučaju velikog stringa. Neophodno je da ova struktura bude puna samo dok se ne završi konstrukcija *Burrows-Wheeler transforma.* Nakon toga može da se koristi proređena verzija gde se čuva svaki n-ti član, ali se usporava pretraga pozicija *exact match*-a. Po *defaultu* nema proređivanja, već samo ukoliko korisnik eksplicitno traži zadavanjem stepena proređivanja.

Kreiranje *Burrows-Wheeler transforma* sa *FM-index*-om

L-kolona se prosto dobija iz još uvek punog *suffix array*-a. *F*-kolona se dobija prebrojavanjem pojavljivanja svakog karaktera u stringu. Poslednja struktura koja čini *FM*-indeks jeste *checkpoint* tabela. Broji isto redova koliko i *L*-kolona. Ona u svakom redu za svaki karakter koji se pojavljuje u stringu pokazuje koliko se puta pojavio do tog reda u *L*-koloni. Ovakva struktura omogućava provera postojanja *exact match*-a u linearnoj složenosti.

Problem je što, kao i *suffix array*, *checkpoint* tabela zauzima relativno mnogo memorije (*length(string) * sizeof(integer)* za svaki različiti karakter koji se pojavljuje u stringu). Zato se i ovde može koristiti proređena verzija gde se čuva svaki n-ti član, ali se usporava provera postojanja *exact match*-a. Po *defaultu* nema proređivanja, već samo ukoliko korisnik eksplicitno traži zadavanjem stepena proređivanja.

Definisanje opsega *read*-ova

Podrazumevano je da se svi *read*-ovi pročitani iz FASTQ fajla poravnavaju. Korisnik može eksplicitno da zada opseg *read*-ova sa kojima će se raditi. Svaki *read* (i pročitana i *reverse complement* sekvenca) zasebno prolazi kroz ceo *workflow.*

Seeding faza

Ova faza je parametrizovana vrednostima dužine *seed*-a i intervalom između dva susedna *seed*-a. Podrazumevane vrednosti ovih parametara su dužina = 20 i interval = 10. Korisnik ih može redefinisati argumentima pri pokretanju programa. *Seed*-ovi predstavljaju podstringove pročitane sekvence i *reverse complementa read*-a zadate dužine i intervala između početnih pozicija. Prvi *seed* kreće od prvog karaktera sekvence.

Algoritam određivanja *exact match*-eva u referentnoj sekvenci je sledeći:

- Za poslednji karakter *seed*-a se odrede pozicije svih pojavljivanja u *F*-koloni. Kako je *F*-kolona sortirana, pojavljivanja su uzastopna i mogu se opisati opsegom.
- Iterira se dalje unazad kroz *seed* dok se ne dođe do početka seed-a ili ne završi sa praznim opsegom. U svakoj iteraciji radi se sledeće:
    - U istom redu *L*-kolone nalazi se karakter koji prethodi karakteru u *F*-koloni. Zato se u prethodno dobijenom opsegu traže sva pojavljivanja trenutnog karaktera u *L*-koloni. Koristeći svojstvo *LF*-mapiranja i vrednosti iz *checkpoint* tabele i *F*-kolone, novi opseg se dobija u konstantnom vremenu, doduše sa većom konstantom ukoliko je tabela proređena, jer se svaki put odrađuju dodatna prebrojavanja.
- Ukoliko je krajnji opseg prazan, ne postoje *exact match*-evi. Ukoliko nije, za svaki element opsega je potrebno odrediti poziciju u referentnoj sekvenci. Proba se čitanje pozicije iz *suffix array*-a. Ukoliko je niz proređen i ne postoji tražena vrednost, proba se sa pozicijom prethodnika koji se nalazi paralelno u *L*-koloni i na ovu poziciju se dodaje +1. Ovo se ponavlja dok se eventualno ne stigne do vrednosti. Ovo je očekivano da se desi u konstanom vremenu, ali ponovo sa većom konstantom.

Za svaki *seed* se definiše važnost kao vrednost obrnuto srazmerna broju *exact match*-eva tog *seed*-a.

Rangiranje prozora

Jezgro prozora predstavlja opseg pozicija *exact match*-eva u referentnoj sekvenci koji ulaze u rezultat prozora. Dužina jezgra prozora se meri u odnosu na dužinu *read*-a. Ovaj odnos je podrazumevano 1.0, ali se može redefinisati argumentom programa. Rezultat prozora jeste suma važnosti svih pripadajućih *exact match*-eva. Prozori se dobijaju proširivanjem jezgra prozora sa leve i desne strane za dužinu *read*-a.

Ne rangiraju se svi mogući prozori, već samo oni čije jezgro počinje u poziciji nekog *exact match*-a. Iterira se kroz rastuće sortiran niz pozicija svih *exact* match-eva za odgovarjući smer (*forward* ili *reverse*) sa dva pokazivača. Prvi pokazuje na početnu poziciju, drugi na prvu poziciju *exact* match-a koja ne upada u jezgro prozora. U linearnom vremenu se računaju rezultati svih željenih prozora. Naravno, zajedno se rangiraju prozori za oba smera datog *read*-a .Broj najboljih prozora koji prolaze u sledeću fazu je podrazumevano 3, ali se može redefinisati argumentom programa.

Čitanje SIM fajla

SIM (skraćeno od *similarity*) format je uveden kao način korišćenja proizvoljnih matrica sličnosti karaktera. Podrazumevane vrednosti predstavljene matrica sličnosti karaktera u SIM formatu se predstavljene na slici 4.

```
A C G T

#   |   A   C   G   T   _    *
A   |   1   -4  -2  -4  -7   -10
C   |   -4  1   -4  -2  -7   -10
G   |   -2  -4  1   -4  -7   -10
T   |   -4  -2  -4  1   -7   -10
_   |   -7  -7  -7  -7  -1000   -1000
*   |   -10 -10 -10 -10 -1000   -1000
```

Slika 4. Podrazumevane vrednosti matrice sličnosti karaktera

Prazne linije i linije koje počinju znakom '#' se ignorišu. U prvom neignorisanom redu fajla navedeni su svi karakteri za koje se definiše sličnost. Osim ovih karaktera, tabela sadrži redove za prazninu ('_') i asterisk ('*') koji predstavlja sve ostale znake. Asterisk vrednosti su uvedene iz predostrožnosti (ukoliko ulazni fajlovi sadrže greške) i ne bi trebalo da budu korišćene tokom izvršavanja. Vrednosti se nalaze u narednim redovima nakon znaka '|'.

Extending faza

U ovoj fazi radi se *end-space-free variant* poravnanje *read*-a sa podstringovima referentne sekvence određim prozorima dobijenim u prethodnoj fazi. Izabrana je *end-space-free* varijanta gde prozor ima oba slobodna kraja, jer se traži približno pojavljivanje *read*-a u prozoru. Za najbolje poravnanje se računa CIGAR string i ove vrednosti se prosleđuju dalje za upis u izlazni fajl.

DP-algoritam je definisan:

- *r = read; w = window;*
- *n = |r| + 1*; *m = |w| + 1*;
- tabela $D_{nxm,}$ matrica sličnosti *S*
- $(\forall j \in [0,m])\ D_{0,j} = 0;\ (\forall i \in [1,n])\ D_{i,0} = D_{i-1,0} + S(r_i, \_)$
- $(\forall i \in [1,n])\ (\forall j \in [1,m])\ D_{i,j} = \max\begin{Bmatrix} D_{i-1,j} + S(r_i, \_) \\ D_{i,j-1} + S(\_, w_j \\ D_{i-1,j-1} + S(r_i, w_j) \end{Bmatrix}$

Paralelno sa popunjavanjem tabele vrednosti, popunjava se i tabela smerova za *backtracking*. Najbolje poravnanje se u tabeli traži u poslednjem redu, a od njega se *backtrack*-uje dok se ne stigne do prvog reda. Ponovo, ovako se radi, jer je u pitanju je *end-space-free* varijanta gde prozor ima oba slobodna kraja. Kolona konačnog polja određuje poziciju najboljeg poravnanja u referentnoj sekvenci.

Pisanje SAM fajla

Izlazni fajl je uprošćeni SAM fajl. Sadrži zaglavlje koje opisuje kolone i red za svaki poravnati *read*. Kolone su odvojene tabulatorima i sadrže sledeće vrednosti:

- identifikator *read*-a,
- smer *stranda*
- pozicija u referentnoj sekvenci
- rezultat poravnanja
- CIGAR string poravnanja
- poravnatu sekvencu,
- kvalitet čitanja poravnate sekvence.

Uputstvo za korišćenje

Opšti oblik pokretanja

```
“DNA Sequence Aligner.exe” {-<command> <args>}
```

Komande

komanda	argumenti	obavezna
`-fa <filepath>`	Argument je putanja do ulaznog FASTA fajla.	DA
`-fq <filepath>`	Argument je putanja do ulaznog FASTQ fajla.	DA
`-sam <filepath>`	Argument je putanja do izlaznog SAM fajla.	DA
`-sparsefm <arg1> <arg2>`	Prvi argument je stepen proređivanja *suffix array*-a. Drugi argument je stepen proređivanja *checkpoint* tabele. Za obe vrednosti uzima se najveći stepen dvojke ne veći od argumenta. Podrazumevane su pune strukture. Korišćenjem ove opcije smanjuje se iskorišćenje memorije nauštrb slabijih vremenskih performansi.	NE
`-range <arg1> <arg2>`	Argumenti predstavljaju donju i gornju granicu opsega rednih brojeva *read*-ova koji će se poravnavati. Ako se ne navede, poravnavaju se svi.	NE
`-sim <filepath>`	Argument je putanja do ulaznog SIM fajla. Ukoliko se ne navedi, koristi podrazumevana matrica sličnosti karaktera (Slika 4).	NE
`-seed <arg1> <arg2>`	Prvi argument predstavlja dužinu *seed*-a (podrazumevano 20). Drugi argument predstavlja interval između susednih *seed*-ova (podrazumevano 10).	NE
`-window <arg1> <arg2>`	Prvi argument predstavlja odnos jezgra prozora i dužine *seed*-a (podrazumevano 1.0). Drugi argument predstavlja maksimalan broj najboljih prozora jednog *read*-a koji će se *extendovati* (podrazumevano 3).	NE