Binarno stablo :

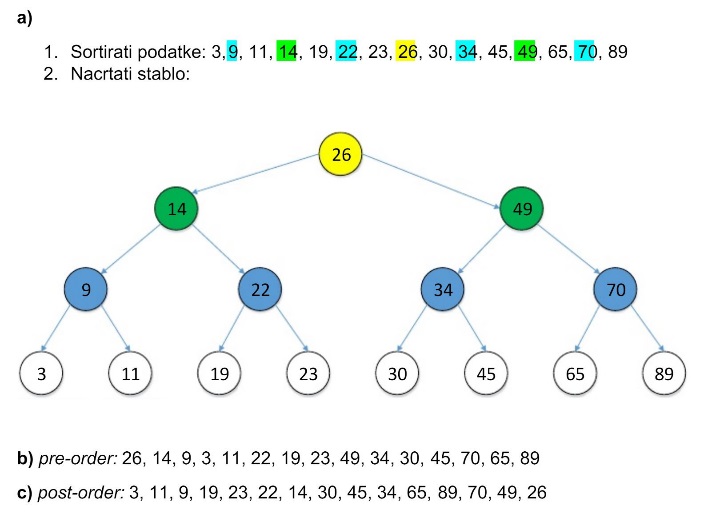
1. Sortirati listu ulaznih brojeva
2. Uzeti sredisnji element -> postaviti ga za root
3. Uzeti sredisnji element s lijeve strane postaviti ga lijevo, s desne strane desno
4. Repeat do kraja

Pre order prolazak -> Ispiši sve node-ove od roota pa do najmanjeg elementa (skroz lijevo) zatim se postupno vracati i ispisivati sljedeci najmanji (desno dijete).

Post order -> Prvo otići najlijevije u stablu do najmanjeg elementa, ispisati ga, zatim desno dijete ako postoji i onda roditelja

Uravnoteživanje stabla je bitno radi brzine pretraživanja, nedostatak je potreba za sortiranjem ulaznih podataka, potreba za dodatnom memorijom i što rezultat ne mora biti popunjeno stablo

Najgori slučaj je kralježnica (backbone)



Brisanje čvora:

* Deletion by copying - Pronaći največeg prethodnika ili najmanjeg sljedbenika, kopirati vrijednost i premostiti pokazivač na dijete zamjenika
* Deletion by merging - Pronaći najmanji veći (najlijeviji u desnoj grani) ili najveći manji (najdesniji u lijevoj grani) i postaviti na mjesto čvora koji se briše te zatim izbrisati čvor -> lošije jer ruši stablo

B-Stabla:

* Root ima najmanje 2 djece, osim ako je ujedno i list (sam/solo)
* Svaki čvor sadrži bar k-1 ključeva i k pokazivača (m/2 <= k <= m, najvece cijelo)
* Svi listovi sadrže bar k-1 ključeva
* Svi listovi su na istoj razini

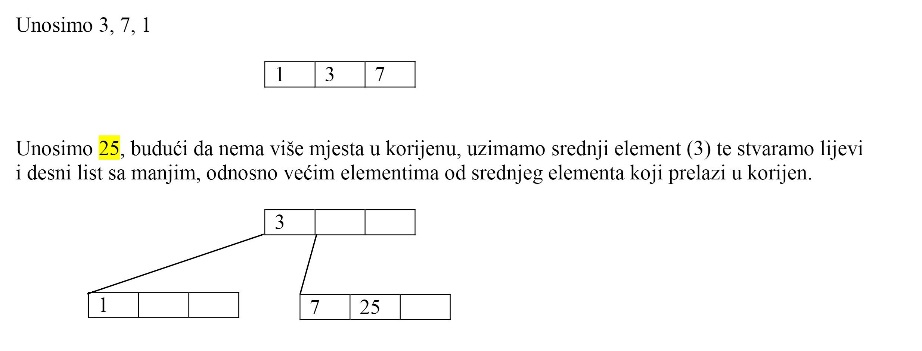
Osobitosti (Važna svojstva):

* Popunjenost je bar 50%
* Savršeno su uravnotežena

1. Savršeno uravnoteženo stablo (Perfectly Balanced Tree) – uravnoteženo stablo kojemu su svi listovi u najviše dvije razine.
2. Uravnoteženo stablo (Balanced Tree) – stablo u kojemu je razlika visina podstabala svakog čvora najviše jedan.

Max(k-1) -> broj kučica, max(k) -> broj pokazivača

Ako je popunjeno i treba splitati uzima se srednji (najvece cijelo) i stavlja se kao roditelj čvor



Maksimalan broj elemenata B-stabla () reda m na visini iznosi , gdje je

(\*RB STABLA su reda m=2)

Omjer B/RB stablo ->

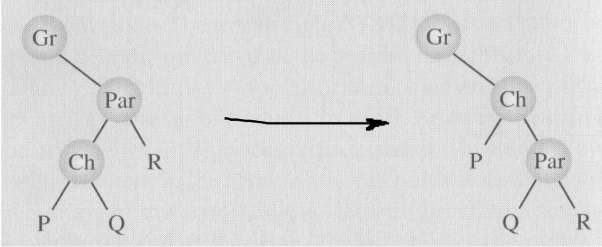
Optimalni red stabla za pretraživanje :

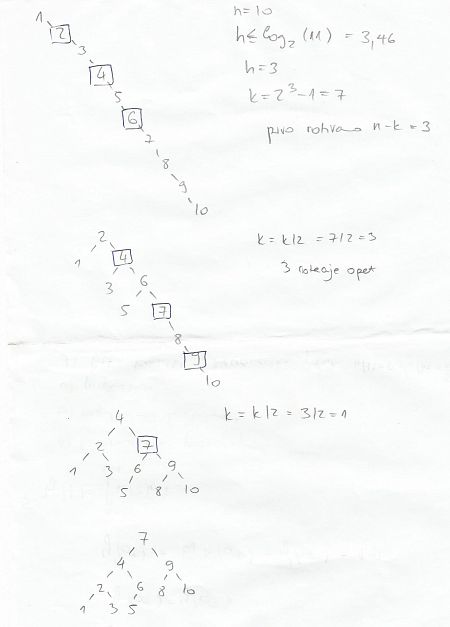
Sparse table: 100% popunjenost tablice

* Prednosti:
  + Racionalnije raspolaganje memorijom
  + Brzo pronalaženje svih podataka jedne skupine koji su u relaciji s jednim podatkom druge skupine
* Nedostaci:
  + Sporiji pristup pojedinačnim podacima
  + Čvorovi lista zauzimaju više memorije nego podatak u tablici

DSW algoritam

* Služi za savršeno uravnotežavanja stabla
* Osnova su rotacije u stablu (dijete postaje roditelj, a roditelj dijete)

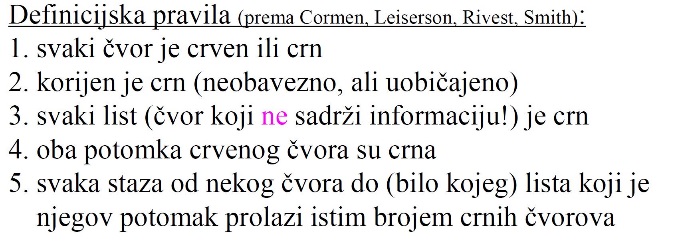


Koraci:

* Stvaranje backbone-a (kralježnice s desna na lijevo pomoću rotacija), uzmemo najveći prethodnik od Roota i rotiramo ga oko roota, i ponavljamo postupak
* Broj elemenata n -> h<=log2(n+1), k=2^h-1
* Napraviti n-k rotacija (za svaki drugi čvor)
* Dok je k>1, k=k/2 i napraviti k lijevih rotacija za svaki drugi čvor

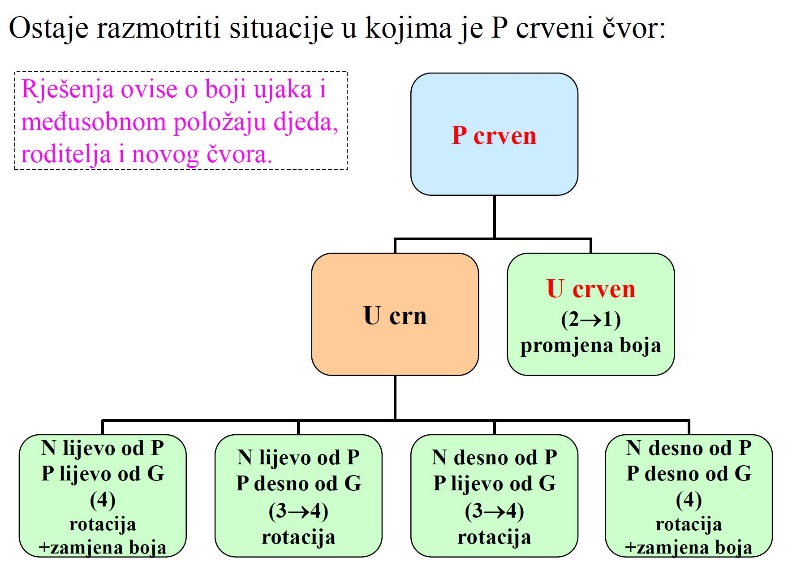
Složenost DSW algoritma je O(n) uvijek u svakoj fazi

Nedostatak DSW algoritma je uravnotežavanje cijelog stabla, iako je ravnoteža najčešće narušena samo lokalno. AVL algoritam uravnotežava stablo lokalno, ali ne jamči savršenu uravnoteženost

RB stabla:

* Svaki novi čvor je crven
* Root node je uvijek crn
* Djeca crvenog čvora su uvjek crna
* Svaki put niz stablo mora imati isti broj crnih čvorova

Crveno crna stabla imaju logičko strukturnu vezu s B-stablima 4. reda

Restrukturiranje stabla nakon dodavanja čvora:

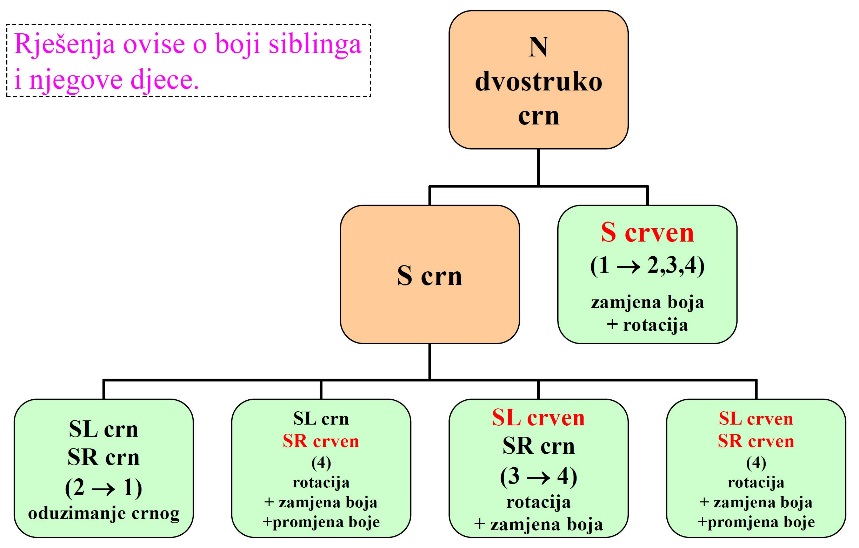
1. Root je crven -> prefarbati u crno
2. Roditelj (P) i ujak (U) su crveni
   * Prebojati P i U u crno i G u crveno
3. Roditelj(P) crven i ujak(U) crn
   * Rotacija Djeteta(N) oko Roditelja(P)
   * Korak 4
4. Roditelj(P) crven i ujak(U) crn
   * Rotacija P oko G
   * Zamjena boja P i G

Cvor brisemo tako da prepisemo podatke iz najblizeg prethodnika ili sljedbenika (zamjenski cvor X) i uklonimo zamjenski cvor X (ima najvise jedno dijete).

Ukoliko je X crven, nista ne treba raditi jer svojstva nisu narusena.

Ukoliko je cvor X crn prebacujemo crninu na njegovo dijete N te imamo sljedece situacije:

Restrukturiranje stabla nakon brisanja čvora:

1. Uklonjen je korijen, dijete postaje korijen i farbamo ga u crno
2. Ako je N crven, on postaje crveno-crn i mora se prefarbati u crno
3. Ako je N crn postaje dvostruko crn te imamo situaciju sa slike:

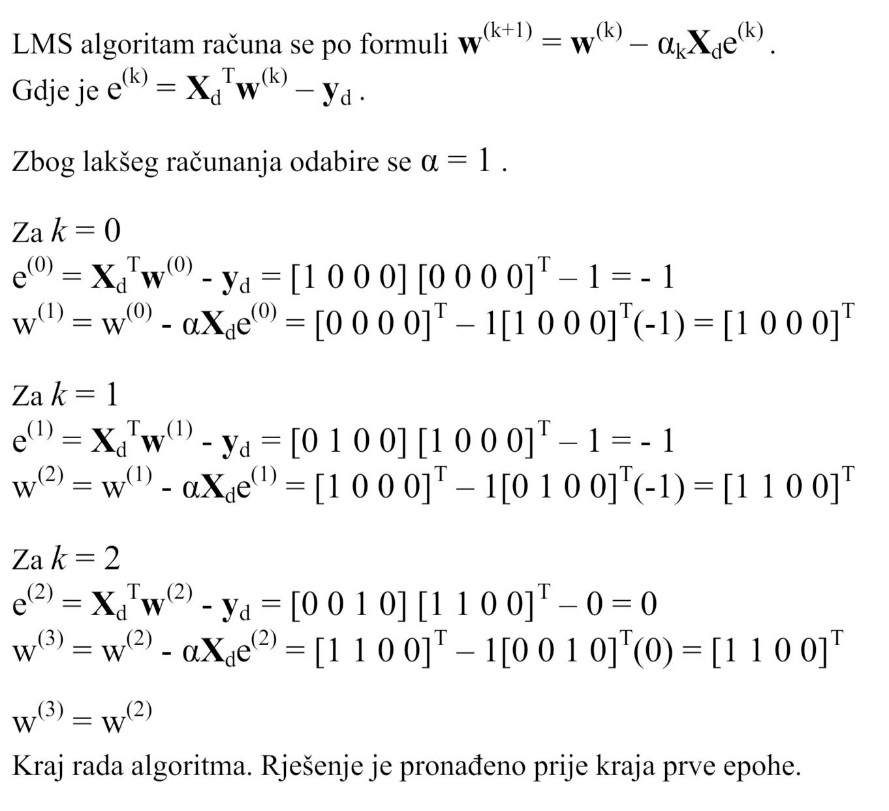
* S je crven:
  + Rotacija oko P i zamjena boja sa P
* S je crn:
  + Djeca od S crna:
    - P crven
      * Crnina od N i S prelazi u P, N postaje crn, a S crven. (prefarbati P u crno)
    - P crn
      * Crnina od N i S prelazi u P, N je crn, S crven..a kod P se odbacuje (prefarbati P u crno)
  + Sl crven, Sr crn, P nebitan
    - Rotacija Sl oko S i zamjena boja Sl i S
  + Sr crven, P nebitan, Sl nebitan
    - Rotacija S oko P
    - Sr prefarbati u crno
    - Zamjena boja S i P

Neuronske mreže:

Linearni neuron (Adaline)

* Nema izlazne funkcije, preslikavanje je
* Broj ulaza (parova za vjezbu) – p i broj tocaka – n
  + p=n, rjesenje je jedinstveno
  + p<n, rjesenja ima beskonacno i odabire se ono najmanje norme sto je rjesenje
  + p>n, točno rjesenje moze i ne mora postojati, ako izraz postoji onda je to optimalno rj.

Gradijentna metoda:

* , k=0,1,2..
* Sa izvršavanjem metode završavamo kada dobijemo 2 ista rezultata za redom

Potpuna mreza – Forward pass reverse pass …

Uvijek nalaze globalni optimum i više iteracija (uvijek) dovodi do boljeg (ili jednako dobrog) rješenja od onog dobivenog s manje iteracija NISU značajke genetskih algoritama

Koriste nasumične (slučajne brojeve), rad im ovisi o brojnim podesivim parametrima i načelo rada im je pokušaj oponašanja evolucije u prirodi JESU značajke genetskih algoritama

* Žuto highlightano iz scribd nasp zadataka

Genetski algoritam (GA)

- čine ga populacije jedinki predstavljenih kromosomima, mehanizmi reprodukcije, mehanizmi vrednovanja uspješnosti jedinki, a koristi se za rj teško rj problema

-evolucija nema globalnu memoriju, inf je sadržana u kromosomu, ako on ne uspije preživjeti evolucija ga ne može vratiti natrag  
-evolucija djeluje na kromosome, npr. (x,y) – jedinka podložna evoluciji

-**pseudo-kod**: inicijaliziraj početnu generaciju kromosoma, izračunaj dobrotu (vrednuj jedinke) za tu generaciju – ako ispunjava uvjete zaustavi proces, inače odaberi jedinke za razmnožavanje, križaj ih i mutiraj – dobiješ djecu – novu generaciju, zamjeni staru generaciju novom i vrednuj novu...(proces se ponavlja odavde)

- kromosom – niz 0 i 1 (binarna reprezentacija) koji kodira moguće rj problema  
- funkcija dobrote/vrednovanja predstavlja uspješnost jedinke u konkretnom problemu

- faktori GA utječu jedan na drugoga i na sam algoritam:

- kodiranje rj kromosomom (binarna reprezentacija)  
- inicijalizacija populacije (slučajna birnarna inicijalizacija)  
 - odabir jedinki za križanje (roulette wheel parent selection)  
- križanje (s prekidnom točkom ili uniformno)  
 - mutacija bita (2 varijante)  
 - brisanje starih jedinki  
 - vrednovanje jedinke

- uvjeti zaustavljanja GA – broj generacija, vremenski, pad dobrote rj ispod neke granice...

- roulette wheel parent selection – slučajni odabir pri čemu je vjerojatnost odabira razmjerna dobroti jedinke – pie chart jedinki prikazuje dobrotu jedinke (tj vjerojatnost odabira pi = dobrotai/suma svih dobrotai) – uspješnije jedinke imaju veću šansu za križanjem

- križanje i mutacija – genetski operatori (binarni i unarni, tim redoslijedom), gdje je mutacija često dio operacije križanja (nastanak jedinke u 2 koraka), ali može biti i zaseban operator  
- križanje (u poč fazi je bitnije) heterogenu popilaciju izkonvergira u homogeniju, dok mutacija (u zav fazi je bitnija) tu homogeniju populaciju može promijeniti i pomaknuti rj bliže optimumu  
- križanje s 1 prekidnom točkom – uzmeš 2 kromosoma, nađeš random prekidnu točku i zamjeniš desne strane kromosoma i dobiješ 2 djece (1 prekidna točka, 2 parenta 🡪 2 djece!!!) – tamo gdje bitovi ostanu jednaki križanje ne unosi nikakve promjene, križanje s više prekidnih točaka može proizvesti ograničene kombinacije jedinki  
100.1110 + 111.1010 = 100.1010 & 111.1110

- uniformno križanje – odabrani bitovi jednog parenta zamjenjuju se s bitovima drugog parenta (efektivno kao da ima n prekidnih točaka, a kromosom n+1 bit)  
1100101 1000011 = 1100011 & 1000101

- mutacija – djeluje na zasebne bitove nezavisno, može se provoditi na svakoj novoj jedinki, postoje dvije varijante: 1. za svaki bit se odlučuje s odr vjerojatnošću hoće li mutirati – ako da, onda se slučajno odabire njegova nova vrijednost (0 ili 1) 🡪 50% da se neće promijeniti  
 2. mutirani bit se mijenja

- funkcija dobrote (suprotno je funkcija greške!!!)– cons: može dati negativne vrijednosti 🡪 translacija, slaba učinkovitost selekcije za male razlike u dobrotama velikih vrijednosti 🡪 linearna normalizacija ili windowing  
- linearna normalizacija (100, interval): poredaš početne dobrote jedinki po padajućoj vrijednosti, onoj s najvećom vrijednosti daš novu dobrotu od 100, a ostale računaš tako da oduzmeš interval od najbliže veće  
Početna dobrota: 300 12 10 10 2  
Linearna dobrota(100, 10): 100 90 80 70 60

- windowing: poredaš početne dobrote jedinki po padajućoj vrijednosti, nađeš najmanju dobrotu u populaciji te računaš dobrote od ostalih jedinki, dobrota(i) = poč dobrota(i) – min dobrota(i) ILI se postavi neka granična najmanja vrijednost i ona se oduzima  
Originalna dobrota: 300 12 10 10 2 🡨 min je 2  
Windowing: 298 10 8 8 2  
Windowing sa granicom 5: 295 7 5 5 5

- zamjena starih jedinki novima: generacijska zamjena – sve stare se brišu i zamjenjuju s novom; mogućnost nestanka dobre jedinke uslijed križanja i mutacije postane loša 🡪 sačuva se elitizmom ili dobre jedinke ne dobiju šansu za križanje ili ih mutacije unište 🡪 ravnotežna reprodukcija – manji broj (određuje parametar) populacije zamijeni se novim jedinkama, zabranjuje se stvaranje duplikata!

**HIBRIDNI GENETSKI ALGORITMI**

**-** kod GA kromosom je binarni niz, operatori nisu prilagođeni konkretnom problemu  
- kod HGA to dvoje se prilagodi konkretnom problemu te se koristi neki postojeći algoritam za rj tog problema u kombinaciji sa GA (HGA je obično bolji od njih)  
- hibridizacija: prirodni prikaz rješenja (npr. 3 realna broja i 2 booleana), korištenje postojećeg algoritma svugdje gdje je moguće (npr. generiranje poč populacije), prilagodba gen operatora konkretnom problemu (npr. križanje uprosječivanjem – koordinate djeteta su prosjek odg koordinata roditelja, mutacija - promjena koor za ograničeni iznos)

Primjena na kombinatorne probleme: pronaći optimalnu kombinaciju  
- problem bojanja grafa – vrhovi imaju težine, susjedni ne smiju biti obojani istom bojom – želi se maximizirati ukupna težina vrhova (vrhovi se obilaze redom iz kromosoma i bojaju prema pravilu te se izračunava dobrota)  
- mutacija – zamjena elemenata u listi ili random promjena redoslijeda podliste  
- križanje – nije moguće uniformno – nelegalna rj (ponavljanje istih elemenata), no moguće je npr. odabrati bitove jednog roditelja (i komplementarne u drugom) te preslikati ostale bitove u djecu, a praznine nadopuniti elementima koji fale onim redom kojim se pojavljuju u roditelju od kojeg nije dobilo početne elemente  
roditelj 1: ABCDEFGH  
roditelj 2: BFDAHGCE  
dijete 1: A \_ \_ DE\_ GH (nedostaju BCF, a u roditelju 2 pojavljuju se redom BFC) 🡪 ABFDECGH  
dijete 2: \_FD\_ \_ G\_ \_ (nedostaju ABCEH, a pojavljuju se tim redom u roditelju 1) 🡪 AFDBCGEH

-problem trgovačkog putnika – n gradova 🡪 n kromosoma: mutacija (može kao i u GA) – odabir podliste bitova čiji se poredak invertira npr. ABCDEFG 🡪 ABEDCFG  
-križanje: odabir podliste bitova na istom mjestu u roditeljima (ABCEFGD i BGADCEF) i one se preslikaju u djecu (u križ, znači desni roditelj daje lijevom djetetu i obratno) na isto mjesto  
(\_ \_ \_ DCE \_ i \_ \_ \_ EFG\_), iz tih podlisti definira se preslikavanje (EFG i DCE: E-D, F-C, G-E), kopiraju se elementi koji se ne nalaze u djeci (elementi desnog roditelja u desno dijete, isto s lijevim): (AB\_DCE\_ i B\_AEFG\_), ostale praznine popune se iz preslikavanja(iz desnog roditelja preslika se u desno dijete):   
(ABFDCE\_ i B\_AEFGC) – budući da u lijevom djetetu već postoji slovo E, zadnje slovo ne preslikavamo jer bi dobili ABFDCEE, nego gledamo preslikavanja i vidimo da se D preslikava u E, a E u G koje nam fali u lijevom djetetu i s njim popunimo prazninu i dobijemo ABFDCEG i BDAEFGC

ISPITI

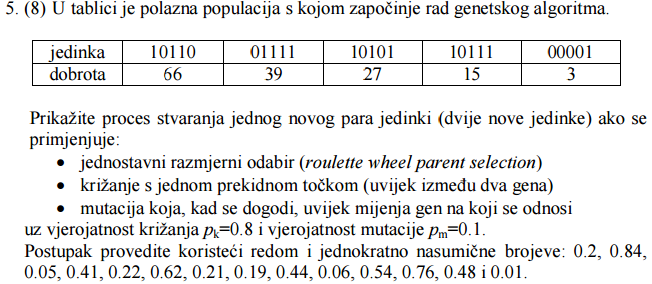
Koje tvrdnje o genetskim algoritmima nisu istinite?

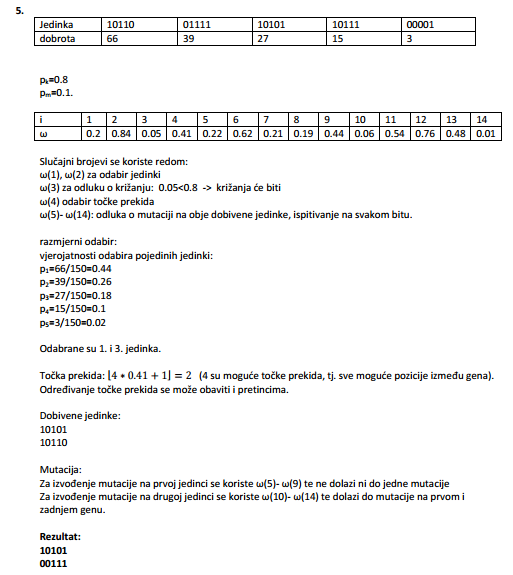
a) Uvijek nalaze globalni optimum.   
b) Da bi bili djelotvorni (korisni), moraju ostvariti (primijeniti) sve evolucijske mehanizme.   
c) Dobrota jedinke uvijek je jednaka vrijednosti ciljne funkcije. nisam sigurna  
d) Rad im ovisi o brojnim ugodivim (podesivim) parametrima koji se ne smiju mijenjati tijekom izvođenja algoritma.   
e) Jedno od područja u kojem su genetski algoritmi vrlo uspješni i često bolji od drugih optimizacijskih metoda jesu kombinatorne optimizacije. HGA se koriste za komb probleme pa zato mislim da je netocno

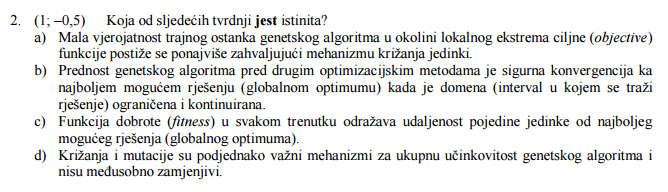
Što od navedenog nije značajka genetskih algoritama?

a) uvijek nalaze globalni optimum   
b) koriste nasumične (slučajne) brojeve   
c) više iteracija sigurno (dakle uvijek) dovodi do boljeg (ili jednako dobrog) rješenja od onog dobivenog s manje iteracija   
d) rad im ovisi o brojnim ugodivim (podesivim) parametrima   
e) načelo rada im je pokušaj oponašanja evolucije u prirodi

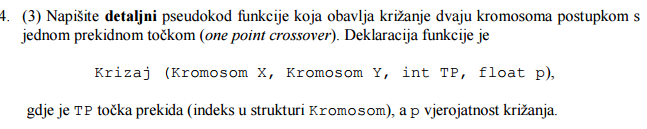
Napišite pseudokod genetskog algoritma koji: koristi razmjerni odabir jedinki (roulette wheel parent selection) primjenjuje križanje s jednom prekidnom točkom (one-point crossover) križanja obavlja s vjerojatnošću pc i mutacije s vjerojatnošću pm kada dođe do mutacije, mijenja gen s vjerojatnošću ps cijelu populaciju zamjenjuje novom (generational replacement). Možete pretpostaviti da su sve pomoćne funkcije na raspolaganju, samo pregledno ispišite njihove nazive i svrhu. Na primjer, pretpostavite da postoji funkcija rand() koja kao rezultat vraća jedan nasumični broj iz populacije U[0,1].



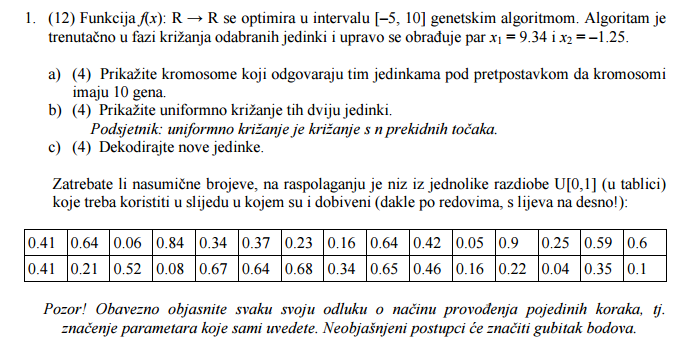




* Odgovor je d – križanja i mutacije su podjednako važni mehanizmi za ukupnu učinkovitost genetskog algoritma i nisu međusobno zamjenjivi



Jebote pseudokod



Jebote opet

