Upotreba evolucijskog algoritma u problemu traženja najkraćeg adicijskog lanca

Autor: Josip Užarević

Mentor: izv. prof. dr. sc. Domagoj Jakobović

Sadržaj

- Motivacija
- Adicijski lanac
- Potenciranje brojeva
- Pronalaženje kratkog adicijskog lanca
- Genetski algoritam
- Analiza
- Zaključak

Motivacija

- Potenciranje brojeva s velikim eksponentima od velikog je značaja za kriptografiju (funkcije sažetka)
- Računalno vrlo zahtjevan postupak, ukoliko se koriste deterministički postupci
- Cilj je heurističkim metodama naći postupak kojim je moguće potenciranje izvesti s manje računskih operacija nego što je to slučaj kod determinističkih postupaka
- Pomoću *adicijskog lanca* izračunava se rezultat potenciranja baze na veliki eksponent
- Genetskim algoritmom traži se što je moguće bolji adicijski lanac

Adicijski lanac

- Predstavlja niz brojeva kod kojih se svaki broj može dobiti zbrajanjem *bilo koja* dva prethodna broja
- Prvi član lanca je broj 1, a drugi član je uvijek 2
- Primjer adicijskog lanca:

$$1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 10 \rightarrow 15$$
$$2 = 1 + 1 = 2 \times 1$$
$$4 = 2 + 2 = 2 \times 2$$
$$5 = 4 + 1$$
$$10 = 5 + 5 = 2 \times 5$$
$$15 = 10 + 5$$

Potenciranje brojeva

- Potenciranje brojeva može se svesti na:
  - Množenje dvaju međurezultata (**zbrajanje eksponenata**)
  - Kvadriranje međurezultata (**množenje eksponenta s 2**)
- Adicijskim lancem predstavlja se potenciranje broja
- Primjer: $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 10 \rightarrow 15$
  - Eksponent 5 dobiven je zbrajanjem eksponenata 1 i 4
  - Eksponent 10 dobiven je udvostručivanjem eksponenta 5
- Pri računalnom izračunavanju, pojam adicijskog lanca omogućava ponovno korištenje već izračunatih rezultata

Potenciranje brojeva

- Primjer izračuna 3^{11}:

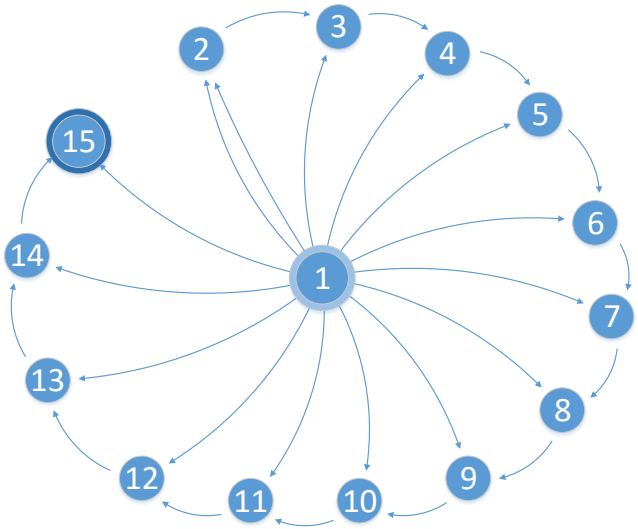
$$3^{15} = \left(\left((3^1)^2\right)^2 \times 3^1\right)^2 \times 3^5$$

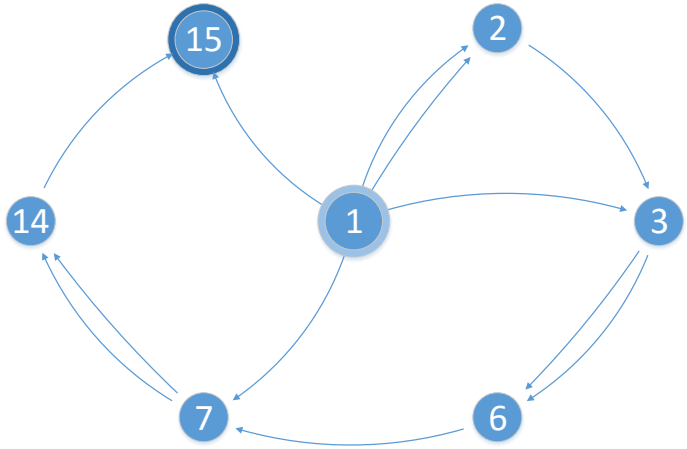
- Pri tome su 3^1 i 3^5, su ponovno korišteni nakon prve upotrebe
- U praksi su **eksponenti** do **2^{127}-3** (npr. $3^{2^{127}-3}$)
- Cilj je pronaći što kraći adicijski lanac budući je njegova duljina jednaka broju potrebnih računskih operacija kako bi se došlo do rezultata
- Kako pronaći što kraći adicijski lanac?

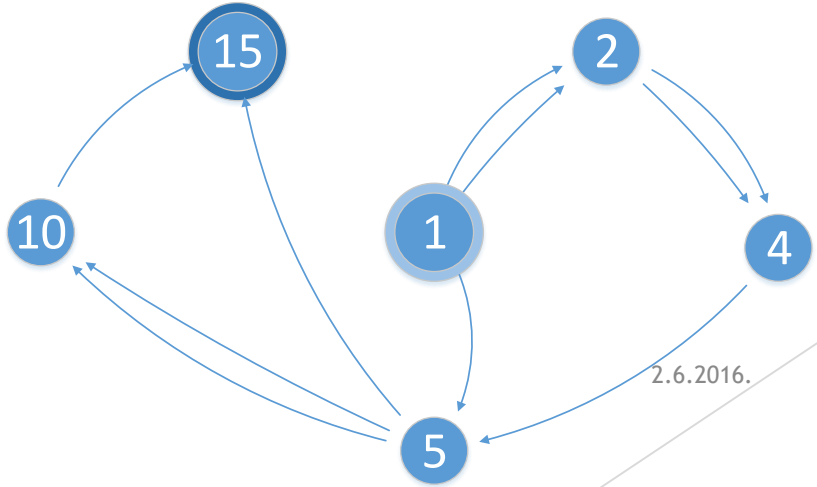
Pronalaženje kratkog adicijskog lanca

- NP-težak problem
- **Ne postoji** algoritam koji traži najmanji adicijski lanac
- Tri primjera traženja (bilo kakvog) adicijskog lanca:
  - Naivni pristup – Uzastopno se broj množi s početnim brojem (x^1) dok se ne dođe do zadanog broja
    $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 15$
  - Binarna metoda – Zadani broj se uzastopno dijeli s 2 ako je djeljiv, ako nije, oduzima se 1
    $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 14 \rightarrow 15$
  - Heurističke metode – Heuristikom se pokušava pronaći što kraći lanac. Naprimjer:
    $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 10 \rightarrow 15$

Pronalaženje kratkog adicijskog lanca







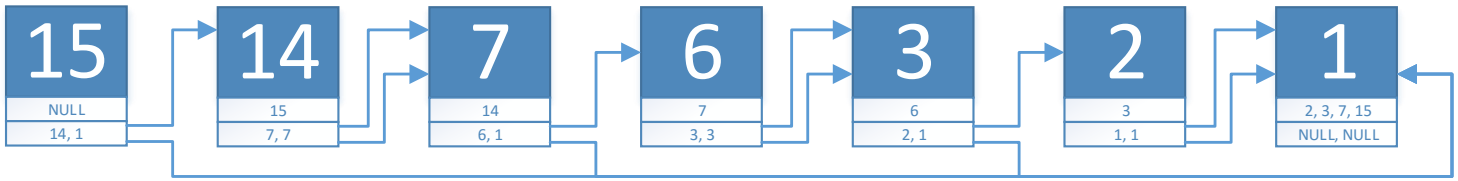
Genetski algoritam

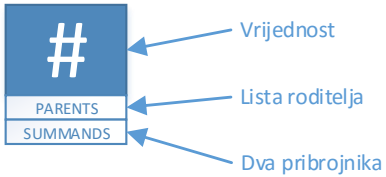
- Za heurističku metodu odabran je genetski algoritam
- Operator selekcije: k-turnirska selekcija
- Dva operatora mutacije
- Jedan operator križanja
- Uvjet zaustavljanja: 100.000 selekcija bez pronađene bolje jedinke
- Jedinka = adicijski lanac
- Inicijalna populacija: adicijski lanci dobiveni binarnom metodom, a zatim mutirani
- Funkcija dobrote: duljina lanca

Reprezentacija jedinke

- Reprezentacija jedinke: vrlo slična prethodnim grafičkim prikazima
- U osnovi usmjereni graf u kojem svaki čvor pokazuje na točno dva druga čvora, osim čvora s vrijednošću 1
- Dodatno, svaki čvor ima pokazivače na svoje „roditelje”, odnosno na čvorove koji pokazuju na njega (osim korijenskog čvora, čija je vrijednost traženi broj te nema roditelja)

Grafički prikaz jedinke

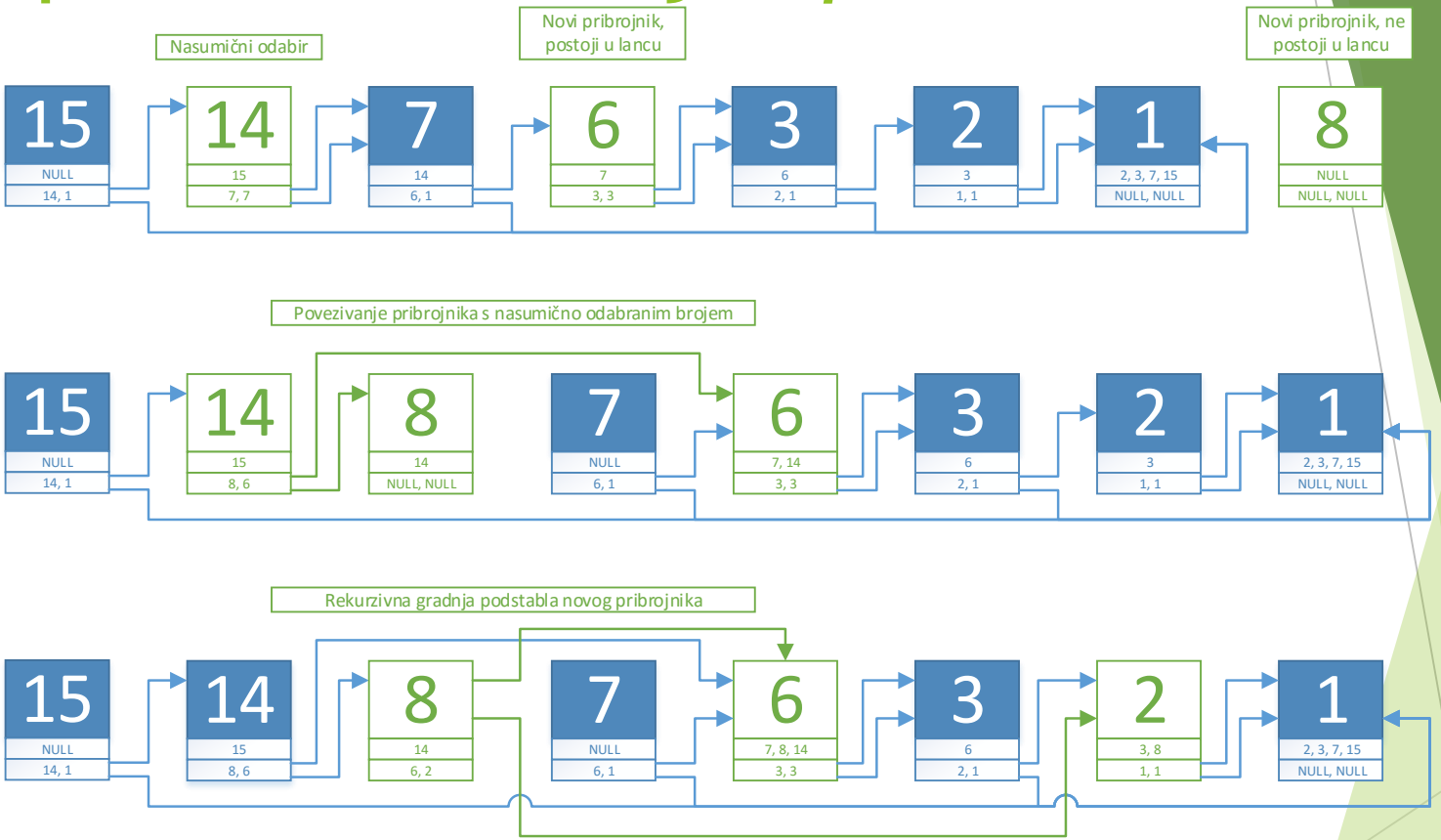




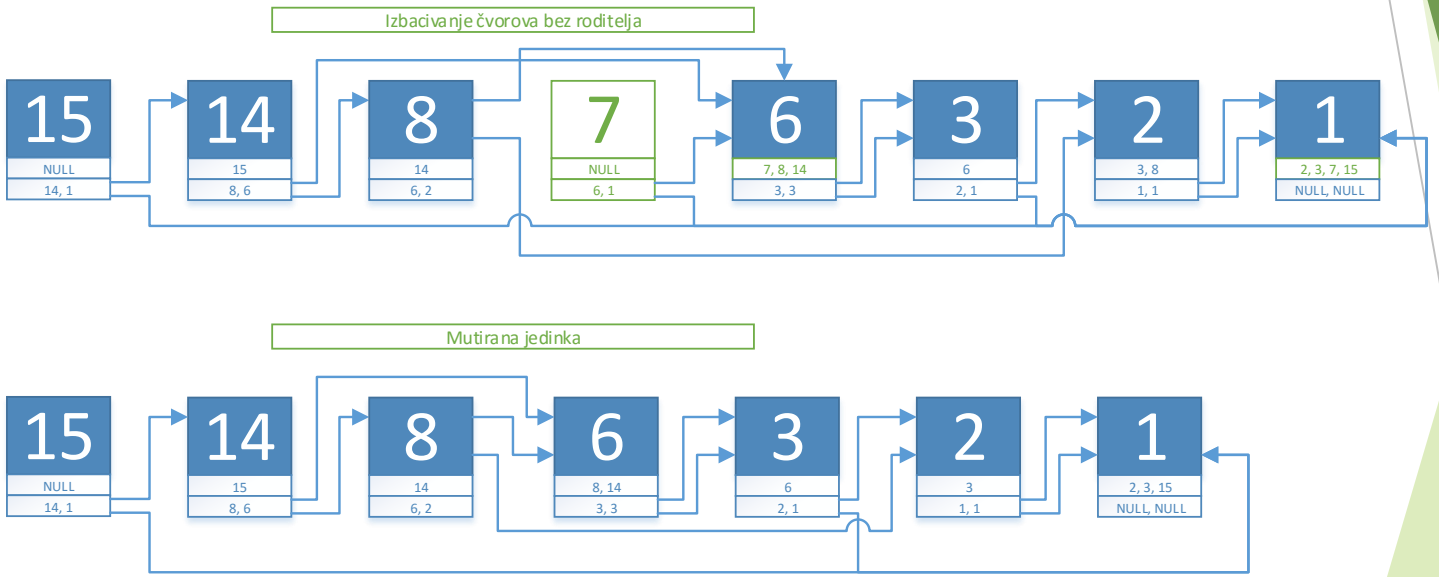
Operator mutacije *SplitNode*

- Bira se nasumični broj unutar lanca
- Traži se drugi manji broj od njega – taj broj je novo prvo dijete nasumičnom odabiru
- Drugo novo dijete je razlika nasumičnog broja i prvog djeteta
- Prvo dijete uvijek postoji u lancu, a drugo ne nužno
- Ukoliko drugo dijete ne postoji, rekurzivno se gradi njegovo podstablo
- Naposljetku se brišu svi čvorovi bez roditelja

Operator mutacije *SplitNode*



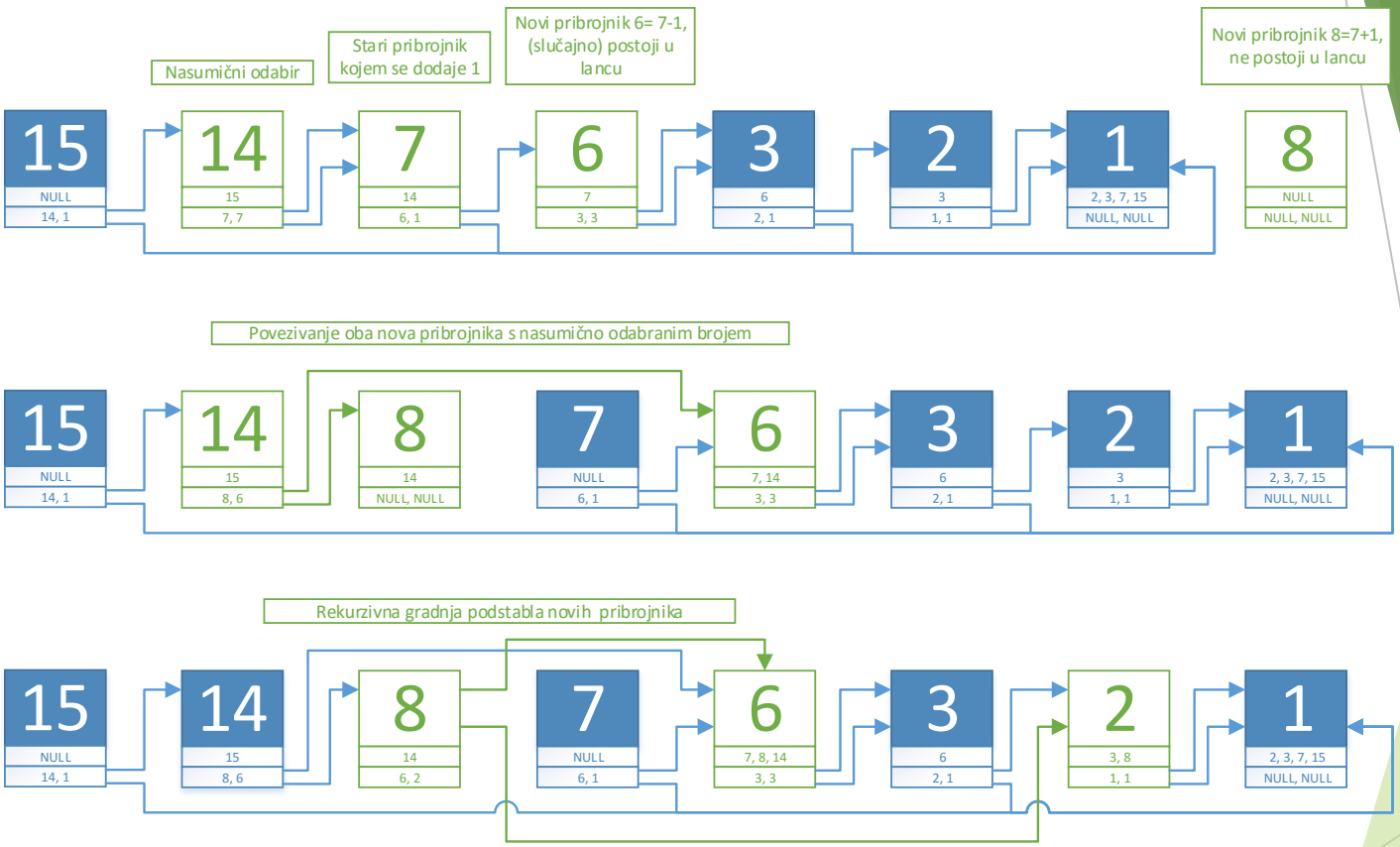
Operator mutacije *SplitNode*



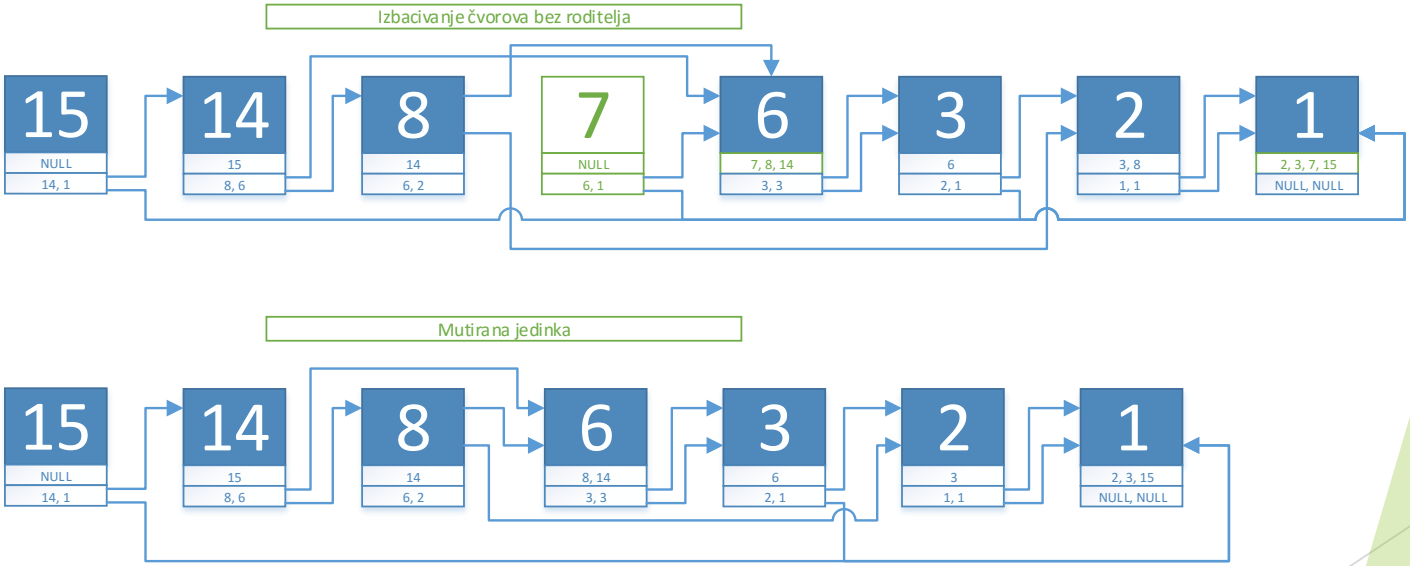
Operator mutacije *AddOne*

- Bira se nasumični broj unutar lanca
- Prvo novo dijete ima vrijednost starom većem pribrojniku uvećanom za jedan
- Drugo novo dijete ima vrijednost starom manjem pribrojniku umanjenom za jedan
- Ni prvo ni drugo novo dijete ne mora postojati u lancu
- Za svu djecu koja još ne postoje u lancu rekurzivno se gradi podstablo
- Naposljetku se brišu svi čvorovi bez roditelja

Operator mutacije *AddOne*



Operator mutacije *AddOne*



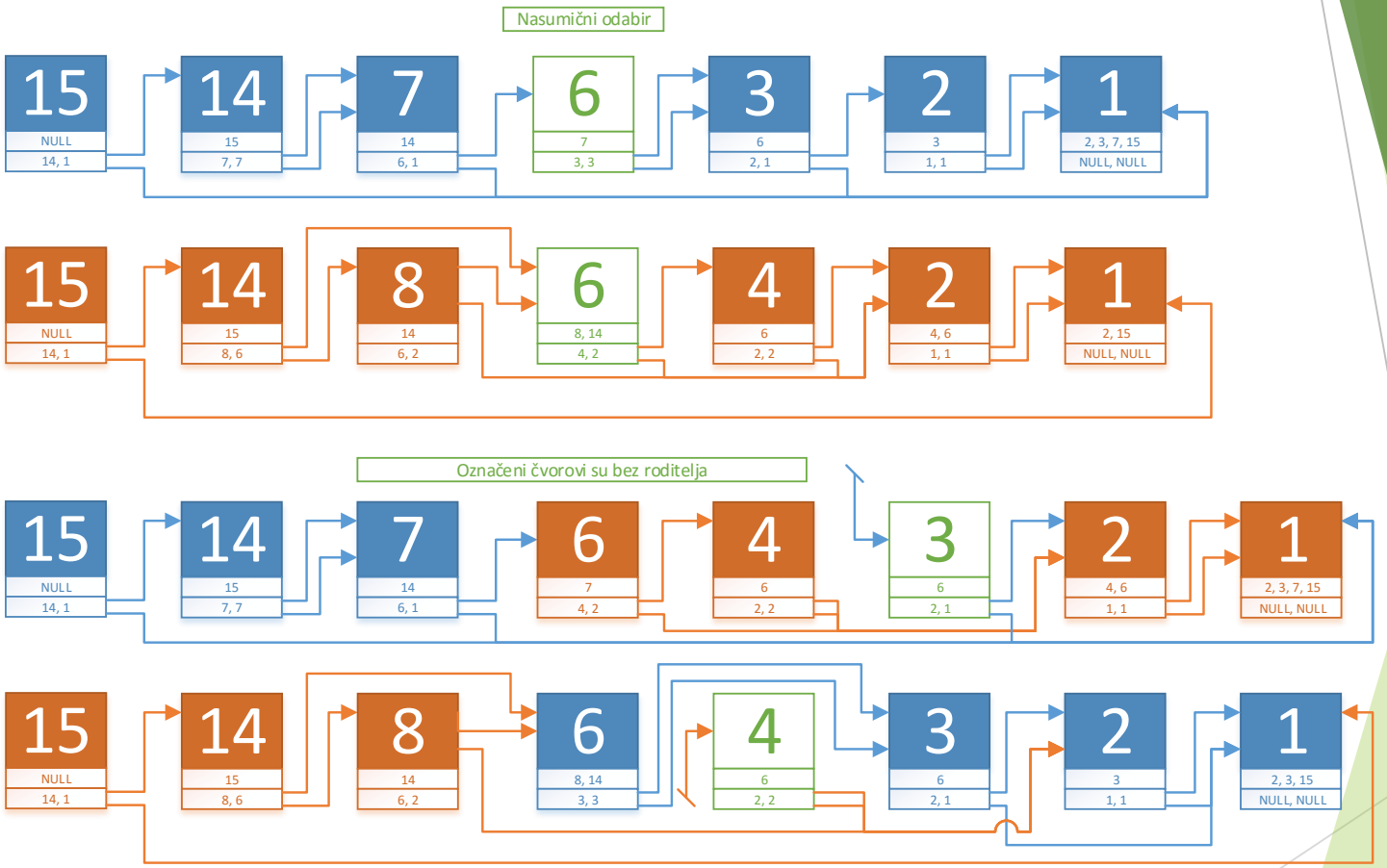
Operator križanja

- Može se interpretirati kao križanje s jednom točkom
- Bira se nasumični broj koji je sadržan u oba roditelja
- Stvara se dvoje djece:
  - Jedno dijete sadrži sve članove veće od nasumičnog odabira iz prvog lanca i sve manje ili jednake iz drugog lanca
  - Drugo dijete sadrži sve članove manje ili jednake nasumičnom odabiru iz prvog lanca i sve veće iz drugog lanca

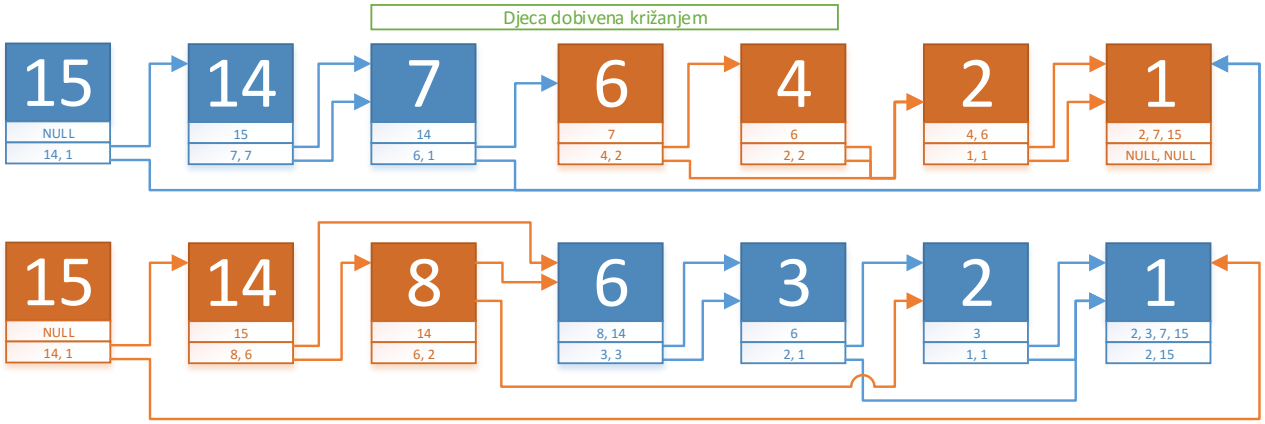
Operator križanja

- Zbog prirode reprezentacije jedinke, najjednostavnije je djecu prvo generirati kao kopije roditelja
- Nakon toga svakom djetetu se dodaje, preko eventualno postojećih članova, donji dio drugog roditelja
- Pri tom dodavanju, dodaju se i veze novo dodanih članova prema njihovim pribrojnicima (ali ne i roditeljima)
- Brišu se svi čvorovi bez roditelja

Operator križanja



Operator križanja

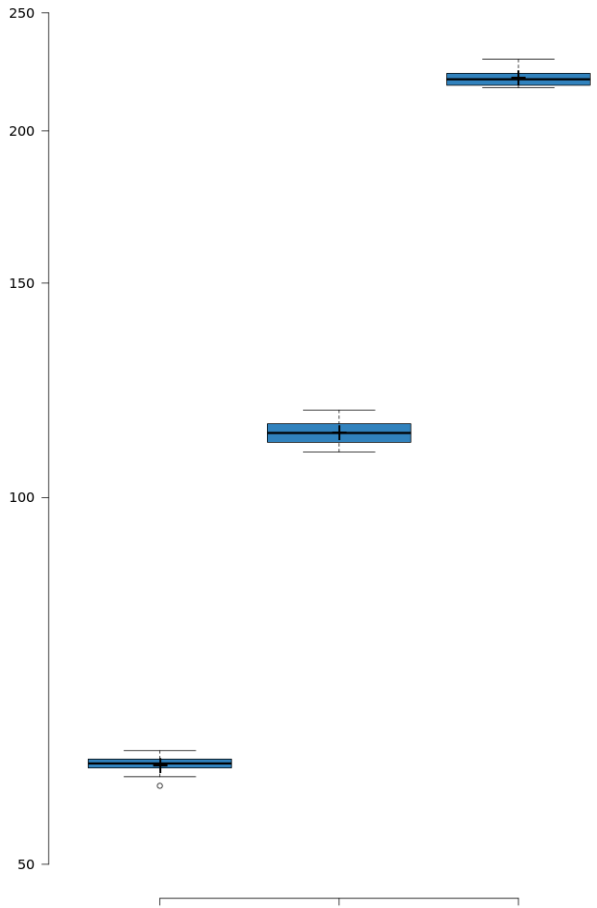


Pseudokod algoritma

```
1.  VELIČINA_POPULACIJE =    100
2.  BROJ_GENERACIJA = 10 000 000
3.  BROJ_STAGNACIJE =    100 000
4.  VELIČINA_TURNIRA = 4
5.  VJEROJATNOST_SPLIT_NODE = 0.75
6.  VJEROJATNOST_ADD_ONE = 0.25
7.
8.  inicijaliziraj_populaciju(populacija)
9.
10. za i=0 do BROJ_GENERACIJA:
11.     ako stagnira BROJ_STAGNACIJE:
12.         Izađi iz petlje
13.
14.     turnir = nasumično_odaberi(populacija, VELIČINA_TURNIRA)
15.     roditelji = dva_najbolja(turnir)
16.     djeca = krizaj(roditelji)
17.     najgori = dva_najgora(turnir)
18.
19.     ako random() < VJEROJATNOST_SPLIT_NODE:
20.         mutiraj(djeca[0], „SPLIT_NODE“)
21.
22.     ako random() < VJEROJATNOST_ADD_ONE:
23.         mutiraj(djeca[1], „ADD_ONE“)
24.
25.     populacija.obrisi(najgori)
26.     populacija.dodaj(djeca)
27.
28. najbolji_lanac = populacija.najbolji_lanac_ikad()
29. ispisi(najbolji_lanac)
```

Analiza

- Parametri algoritma su:
  - Vjerojatnost mutacije *AddOne*: 0.75
  - Vjerojatnost mutacije *SplitNode*: 0.75
  - Veličina populacije: 1000
  - Veličina turnira: 8
- Na slici je u logaritamskoj skali prikazana najmanja pronađena duljina adicijskog lanca za eksponente $2^{37}-3$, $2^{67}-3$ i $2^{127}-3$



Zaključak – općenito i implementacija

- Traženje kratkih adicijskih lanaca od velike su koristi za kriptografiju i druga područja
- Rad s velikim brojevima je teško ispitivati od strane čovjeka, u odnosu na računalo
- Programski jezik Java 8 pokazao se kao odličan izbor za implementaciju algoritma (`BigInteger`, `stream`, `HashMap`)
- Java VisualVM – odličan alat za dijagnostiku implementacije

Zaključak – Genetski algoritam

- Funkcija dobrote koja ovisi isključivo o duljini lanca nije (dovoljno) dobar izbor
- Reprezentacija jedinke kao graf ima i prednosti (jeftini popravak potomaka) i nedostatke (smanjivanje količine gena, potencijal jedinke vs. dobrota jedinke)
- Problematično praćenje evolucije zbog teške interpretabilnosti
- Poboljšanja u budućnosti: paralelizacija, bolji izbor parametara, *adekvatniji* izbor genetskih operatora za velike brojeve

Zahvaljujem na pažnji.

Sva pitanja su dobrodošla.