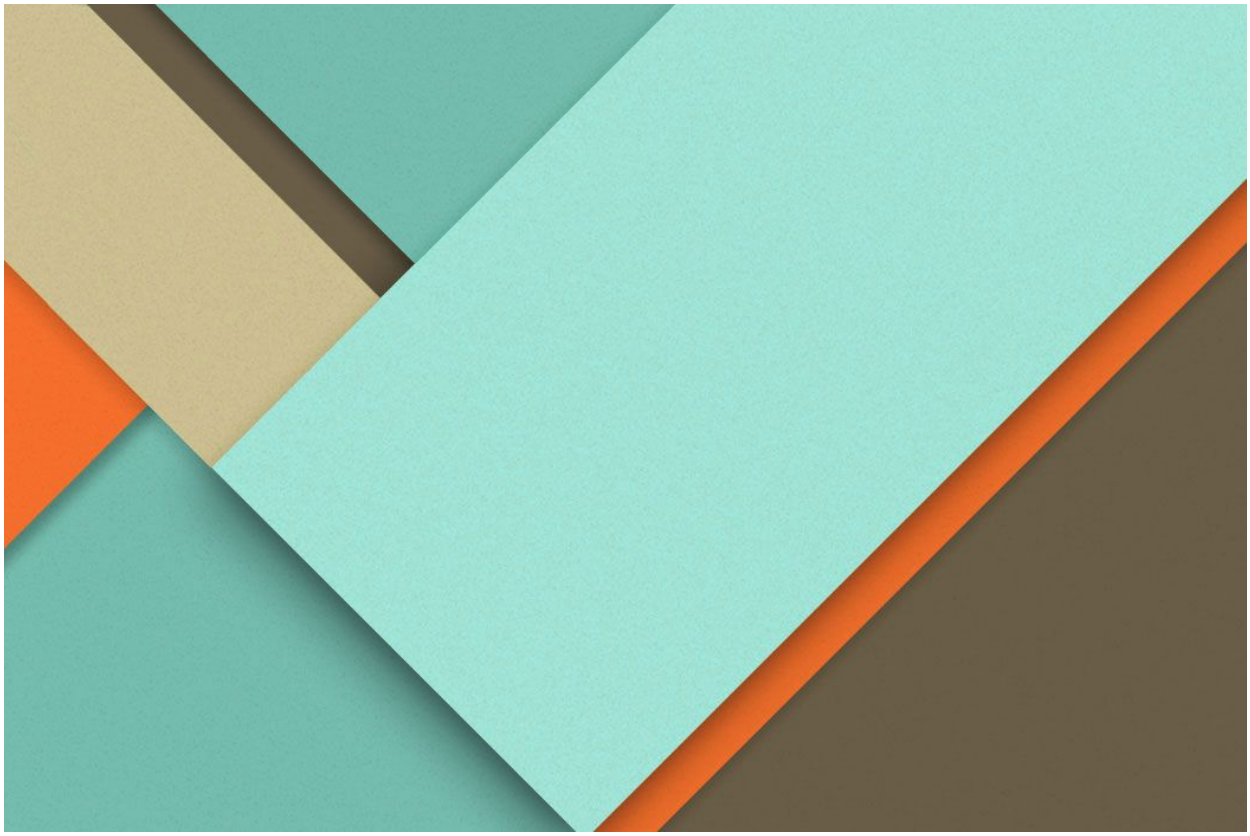


Distribuirani racunalac problema n kraljica

Milorad Stankovic
Racunarski fakultet

Pregled

Distribuirani sistem namenjen resavanju problema n kraljica. Dizajniran i implementiran da bude samoorganizujuc i skalabilan. Implementiran u programskom jeziku **java.**

Sistem je implementiran kao skup samoorganizujucih cvorova, koji izmedju sebe dele posao racunanja. Prisutan je jedan 'bootstrap' cvor koji se jedini ponasa drugacije od ostalih. Komunikacija sa njim je svedena na minimum, tj obratice mu se cvorovi samo kada pristupaju sistemu ili ga napustaju.

Sistem koristi originalan algoritam za odabir vidljivih cvorova, po uzoru na chord algoritam, koji ce biti detaljnije opisan kasnije u tekstu.

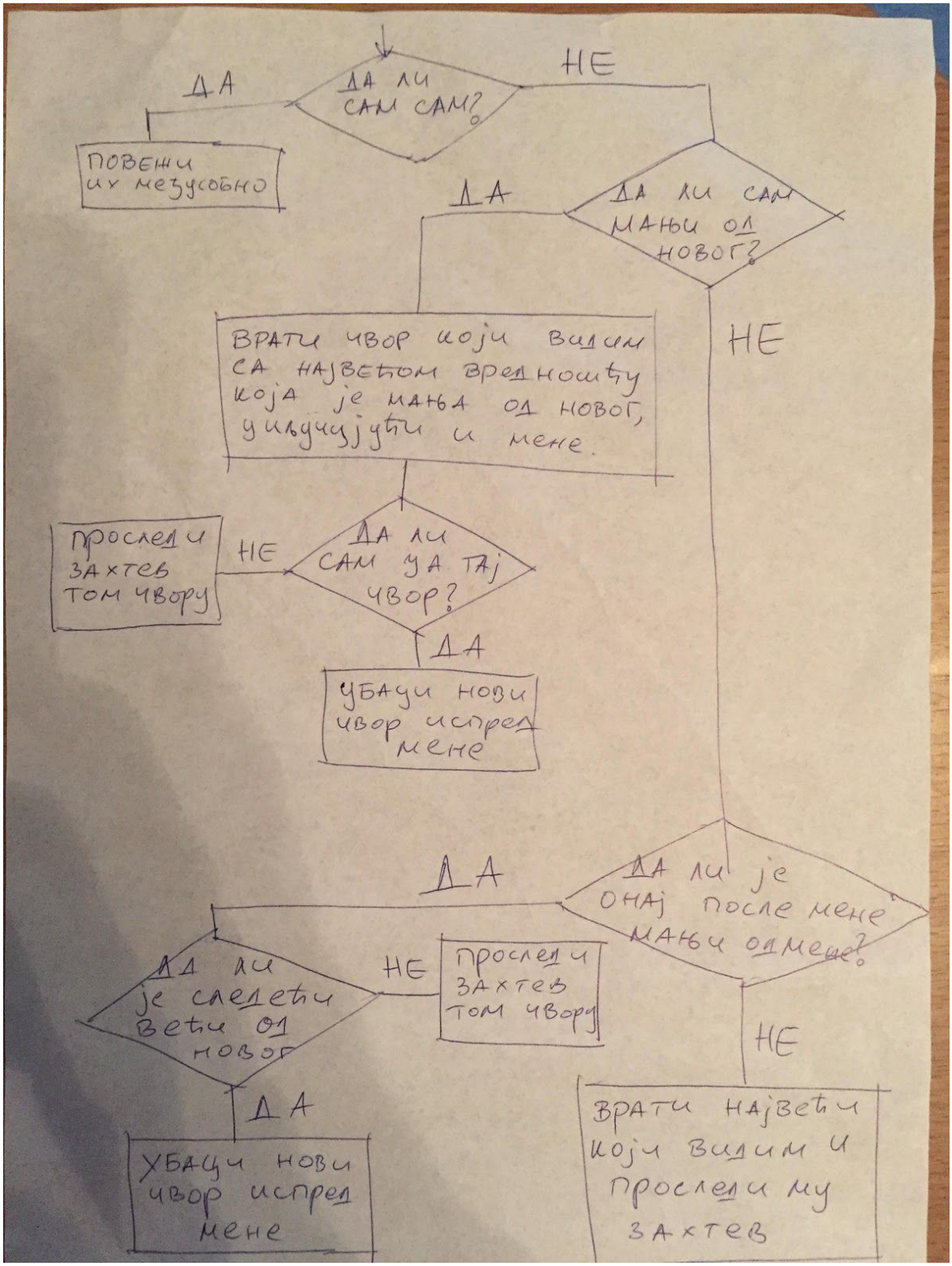
Opis bootstrap servera

Bootstrap cvor (koji jedini zapravo nije cvor) slusa na fiksnoj ip adresi/portu koju znaju svi cvorovi. Zaduzen je za pracenje svih aktivnih cvorova, kako bi kada se neki cvor prikljuci njemu mogao da prosledi neki nasumicni cvor, kome ce se on dalje obratiti da nadje svoje mesto u grafu. Dakle, bootstrap ima dve funkcije, jedna opsluzuje zahtev cvora koji se upravo pridruzuje mrezi, i vraca mu nasumican cvor koji vec radi. A druga je opsluzivanje zahteva za izlaz iz mreze, u kom slucaju brise zadati cvor iz spiska aktivnih, a cvor je nakon toga u obavezi da obavesti ostale cvorove o svom izlasku iz mreze.

Opis cvora

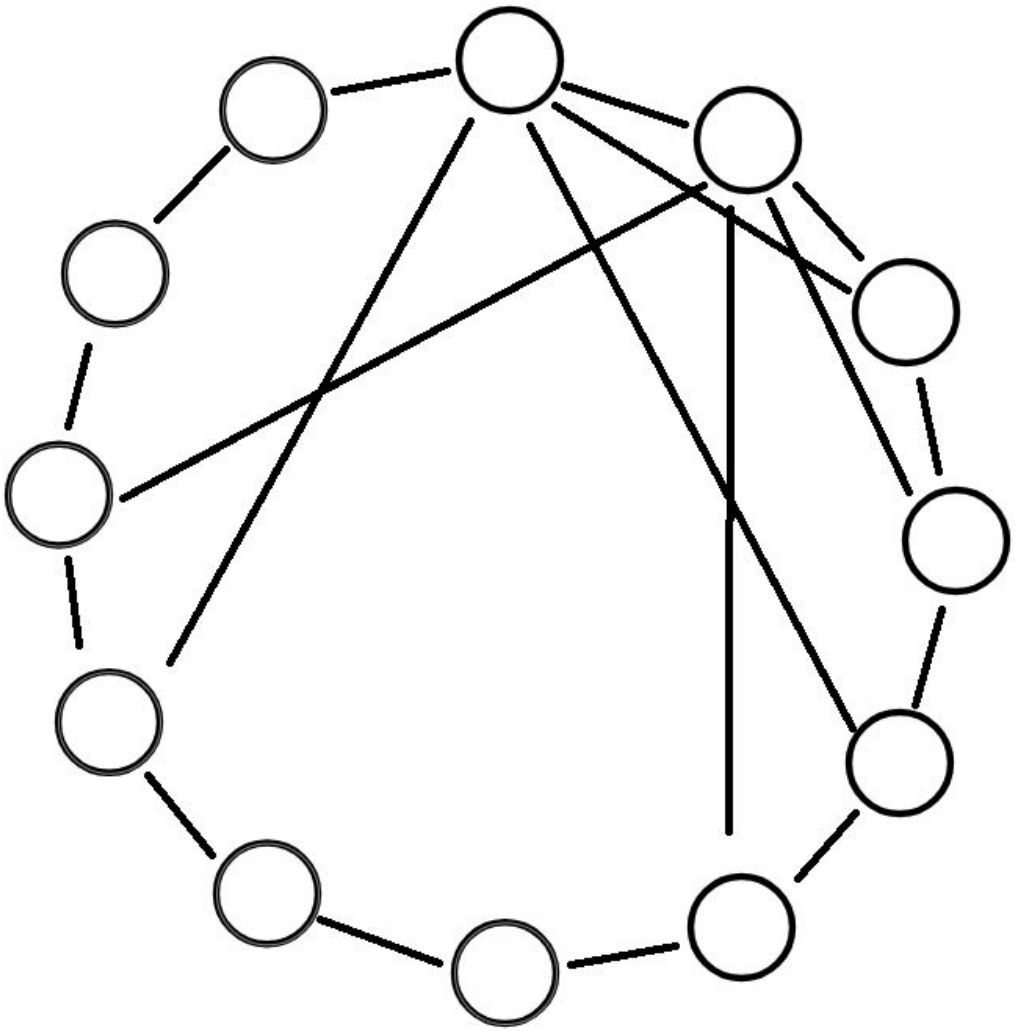
Svaki cvor u sistemu predstavlja jednu masinu. Cvor je u obavezi da pri pokretanju, iscita informacije iz konfiguracione datoteke i odmah nakon toga kontaktira bootstrap server (u konfiguracionoj datoteci su zapisane informacije o bootstrap serveru). Od bootstrapa dobije neki nasumican cvor koji je vec aktivan, i njemu se obrati sa zahtevom za reorganizaciju. Ako je cvor koji se ukljucuje jedini u sistemu, ne dobija novi cvor, vec kada se drugi cvor prikljuci sistemu oni uspostave komunikaciju. Kada cvor dobije od novog cvora zahtev za reorganizaciju, mora da obavesti ostale cvorove o tome, da pronadje adekvatno mesto za novi cvor i da namesti veze izmedju cvorova da zadovoljavaju zahteve algoritma.

Nacrt algoritma za samoorganizovanje:



Opis algoritma za racunanje vidljivih cvorova

Algoritam koji se koristi je nalik na chord algoritam, tj. cvorovi su zaduzeni da u svakom trenutku budu organizovani u prsten, sto se postize algoritmom prikazanim iznad. Kada su cvorovi rasporedjeni u prsten, kako bi sistem obezbedio brzu komunikaciju, koja ce teziti logaritamskoj zavisnosti, svaki cvor trazi najoptimalnije cvorove koje ce da 'vidi' u svakom zadatom trenutku. U ovom algoritmu se ti cvorovi izracunavaju na osnovu udaljenosti od trenutnog cvora u prstenu, i to, cvor ce da vidi sve cvorove koji su od njega udaljeni 2^n < ukupanbrojcvorova. Prikaz algoritma:



Kao sto se ovde vidi, posto ima ukupno 12 cvorova, svaki cvor ce da vidi cvorove koji su od njega udaljeni 2^0 = 1, 2^1 = 2, 2^2 = 4 i 2^3 = 8. Kada bi broj cvorova bio veci od 16, svaki cvor bi takodje video onog ko je od njega udaljen 2^4 = 16 itd.

Opis algoritma za racunanje problema n kraljica

Algoritam za racunanje n kraljica radi tako sto kada mu se dodeli posao, on prvo izracuna ukupan broj iteracija potrebnih kako bi se probala svaka moguca pozicija kraljica, sto je za neko n, gde je matrica nxn, n^n. Podeli posao na koliko ima cvorova, u opsege, i svaki cvor pokrece izracunavanje za svoj opseg. Koristi funkciju koja matematicki racuna pozicije kraljica na tabli na osnovu broja iteracije, tako da olaksava pracenje progresa i kradju poslova.

Kradja poslova

Kada neki cvor zavrsi sa poslom, pita ostale cvorove za koji ga jos nisu obavestili da su zavrsili koliko im je posla ostalo. Ako im je ostalo vise nego sto je opisano u njihovom config fajlu kao donja granica u %, on deli preostali posao na dva posla, skrati postojeci posao a sebi uzme novogenerisani posao. Ponavlja se dok se izracunavanje ne zavrsi u potpunosti.