Anti Laksonen

Vodič za takmičarsko programiranje

Učenje i poboljšavanje algoritama kroz takmičenje

Računarski fakultet

Vodič za takmičarsko programiranje

Isbn 978-89-7991-416-3

Autorizovan prevod sa engleskog jezika prvog izdanja knjige
Guide to Competitive Programming

Originally published by Springer

Prevod	Stanislav Košćal
Recenzija	Petar Prvulović
Lektura	Marija Sloboda
Gl. i odg. urednik	Jovana Ristić
Tehnički urednik	Vesna Petrinović
Slog i prelom	Aleksandar Popović
Izdavači	**CET Computer Equipment and Trade** Beograd, Skadarska 45 tel/fax: 011 3243-043, 3235-139, 3237-246 http://www.cet.rs
	Računarski fakultet Beograd, Knez Mihaila 6/VI tel. (011) 2627-613, 2633-321 www.raf.edu.rs
Za izdavača	Dragan Stojanović, direktor
Tiraž	1000
Štampa	„Pekograf", Beograd

Nastavno-naučno veće Računarskog fakulteta na 135. sednici održanoj 20.6.2019. godine donelo je odluku da knjiga „Vodič za takmičarsko programiranje" autora Antti Laaksonena bude štampana kao univerzitetski udžbenik.

Reč autora

Ova knjiga zamišljena je kao sveobuhvatan uvod u savremeno takmičarsko programiranje. Pretpostavlja se da su vam već poznate osnove programiranja, dok prethodno takmičarsko iskustvo, ili iskustvo u konstrukciji algoritama nisu neophodni. S obzirom na to da pokriva širok spektar tema različite težine, ovu knjigu mogu da koriste i početnici i iskusniji čitaoci.

Takmičenja programera postoje već prilično dugo. Međunarodno studentsko takmičenje iz programiranja (*International Collegiate Programming Contest*) osnovano je tokom sedamdesetih godina prošlog veka, dok je prva Međunarodna informatička Olimpijada (*International Olympiad in Informatics*) za učenike srednjih škola ustanovljena 1989. godine. Oba takmičenja sada predstavljaju priznate i značajne događaje u kojima učestvuje veliki broj učesnika iz svih delova sveta.

Danas je takmičarsko programiranje popularnije nego ikada ranije, a odlučujuću ulogu u njegovom popularisanju odigrao je Internet. Danas postoji aktivna onlajn zajednica programera takmičara i svake nedelje se organizuje veliki broj takmičenja. Istovremeno, povećava se i težina ovih takmičenja. Tehnike kojima su do pre samo nekoliko godina vladali samo najbolji učesnici, danas predstavljaju standarde koji su poznati velikom broju učesnika.

Koreni takmičarskog programiranja sežu u naučne studije algoritama. Međutim, dok naučnik iz oblasti računarstva treba da obrazloži funkcionisanje svog algoritma, programeri takmičari *implementiraju* svoje algoritme i predaju ih na korišćenje takmičarskom sistemu. Nakon toga algoritam se proverava kroz seriju testova i ukoliko ih sve prođe, on se prihvata. Ovo je ključni aspekt takmičarskog programiranja, zato što obezbeđuje način za automatsko dobijanje nepobitnog dokaza da algoritam funkcioniše. Štaviše, takmičarsko programiranje pokazalo se kao izuzetan način za učenje algoritama zato što ono ohrabruje dizajn algoritama koji zaista funkcionišu, za razliku od skiciranja ideja koje možda funkcionišu, a možda i ne.

Druga prednost takmičarskog programiranja jeste to što takmičarski problemi zahtevaju *razmišljanje*. Konkretno, ne postoje nikakve olakšice u rešavanju problema.

Ovo zapravo predstavlja ozbiljan problem na mnogim kursevima na kojima se uče algoritmi. Na njima možete da dobijete zanimljiv problem koji treba rešiti, ali u poslednjoj rečenici stoji nagoveštaj „Problem ćete rešiti izmenom Dijkstrinog algoritma". Kada ovo pročitate, biće vam jasno na koji način neki problem treba da rešite, tako da ćete, samim tim, manje razmišljati. U takmičarskom programiranju ovo se nikada ne događa. Ovde ćete dobiti skup raspoloživih alatki, a na vama je da se *sami* setite kako da ih iskoristite.

Osim toga, rešavanje takmičarskih programerskih problema poboljšava programerska znanja kao i veštinu u otklanjanju problema. Uobičajeno je da se neko rešenje nagradi poenima samo ukoliko se ono pokaže kao uspešno u svim problemskim situacijama. Prema tome, uspešan programer takmičar je onaj koji je u stanju da implementira programe koji u sebi nemaju skrivenih mana. Ovo je veoma značajno u inženjeringu softvera, tako da nije slučajno što IT kompanije često angažuju programere koji su se dokazali u takmičarskom programiranju.

Kvalitetan programer takmičar ne postaje se preko noći, ali nema sumnje da ćete na tom dugom putu mnogo toga naučiti. Možete biti sasvim sigurni da ćete, čitajući ovu knjigu, rešavajući problem i učestvujući na takmičenjima, uspešno ovladati algoritmima.

Ukoliko imate bilo kakvih povratnih informacija, rado ću ih saslušati! U bilo kom trenutku možete mi se obratiti putem mejla: ahslaaks@cs.helsinki.fi.

Na kraju, želim da istaknem da sam izuzetno zahvalan velikom broju saradnika koji su mi poslali svoje komentare na rane verzije ove knjige, jer je time značajno unapređen njen kvalitet. Posebno želim da se zahvalim saradnicima koji su detaljno analizirali rukopis ove knjige. Njihova imena su: Miko Ervasti, Jane Junila, Jane Kokala, Tuka Korhonen, Patrik Estergard i Rope Salmi. Isto tako, želim da se zahvalim i Sajmonu Risu i Vejnu Vileru za izuzetnu saradnju prilikom izdavanja ove knjige.

Helsinki, Finska, Anti Laksonen
Oktobar 2017.

Sadržaj

Uvod

1

U ovom poglavlju naučićete šta je zapravo takmičarsko programiranje i upoznati se sa sadržajem knjige, kao i sa dodatnim resursima za učenje.

Na samom početku, u odeljku 1.1, navedeni su elementi takmičarskog programiranja, zatim neka od najpopularnijih programerskih takmičenja, kao i saveti za pripremu za ova takmičenja.

U odeljku 1.2 navedeni su ciljevi i teme ove knjige i ukratko je opisan sadržaj svakog poglavlja.

U odeljku 1.3 predstavićemo vam skup problema za vežbanje CSES (CSES Problem Set). Rešavanje zadataka i problema uz čitanje knjige predstavlja dobar način za ovladavanje takmičarskim programiranjem.

U odeljku 1.4 biće reči o drugim knjigama koje se bave takmičarskim programiranjem i dizajnom algoritama.

1.1 Šta je takmičarsko programiranje?

Takmičarsko programiranje objedinjuje dve teme: dizajn algoritama i njihovu implementaciju.

Dizajn algoritama Suštinu takmičarskog programiranja predstavlja pronalaženje efikasnih algoritama koji rešavaju dobro definisane računske probleme. Dizajn algoritama zahteva veštinu u rešavanju problema, kao i upotrebu i primenu matematičkih aparata. Vrlo često se do rešenja nekog problema dolazi kombinacijom poznatih metoda i novih pristupa.

Matematika ima veoma važnu ulogu u takmičarskom programiranju. Štaviše, moglo bi se reći da ne postoji jasna granica između dizajna algoritama i matematike. Ipak, ova knjiga je napisana tako da ne podrazumeva značajno prethodno matematičko predznanje. U dodatku koji se nalazi na kraju knjige prikazali smo neke matematičke koncepte koji

su primenjeni u ovoj knjizi, kao što su skupovi, logika i funkcije. Može vam poslužiti kao referentni podsetnik tokom čitanja ove knjige.

Implementacija algoritama U takmičarskom programiranju rešenja problema procenjuju se testiranjem implementiranog algoritma na skupu test primera. To znači da, nakon pronalaženja algoritma koji rešava problem, sledeći korak predstavlja njegova pravilna implementacija, što podrazumeva dobro poznavanje programiranja. Takmičarsko programiranje se značajno razlikuje od tradicionalnog razvoja softvera prema principima softverskog inženjerstva: programi su kratki (najčešće nisu duži od nekoliko stotina redova), treba ih napisati brzo i nije neophodno održavati ih nakon takmičenja.

U ovom trenutku najpopularniji programski jezici koji se koriste na takmičenjima su C++, Python i Java. Primera radi, na takmičenju *Google Code Jam 2017*, od 3000 najboljih učesnika, 79% je koristilo C++, 16% Python i 8% Javu. Mnogi programeri smatraju da je C++ najbolji izbor kada je u pitanju takmičarsko programiranje. Prednosti ovog programskog jezika su velika efikasnost i njegova standardna biblioteka, koja sadrži veliku kolekciju implementiranih struktura i algoritama.

Svi primeri programa iz ove knjige napisani su u programskom jeziku C++, uz korišćenje struktura podataka i algoritama iz njegove standardne biblioteke. Programi su pisani prema standardu C++11, koji se može koristiti na većini aktuelnih takmičenja. Ukoliko niste vični programiranju u programskom jeziku C++, pravi je trenutak da počnete sa učenjem.

1.1.1 Takmičenja programera

Međunarodna informatička olimpijada (*International Olympiad in Informatics*, IOI) predstavlja godišnje programersko takmičenje učenika srednjih škola. Prema pravilima, svaka država na ovo takmičenje može da pošalje tim koji čine četiri studenta. Na taj način obično se skupi oko 300 takmičara iz osamdesetak država.

Međunarodna informatička olimpijada obično se sastoji od dva petočasovna takmičenja. Na oba takmičenja od učesnika se očekuje da reše tri ozbiljna programerska zadatka. Zadaci su podeljeni u segmente od kojih svaki donosi određene poene. Iako su članovi timova, takmičari se zapravo takmiče pojedinačno.

Učesnici na Međunarodnoj informatičkoj olimpijadi biraju se kroz nacionalna takmičenja. Pre same Olimpijade organizuju se mnoga regionalna takmičenja kao što su Baltička informatička olimpijada (BOI, *Baltic Olympiad in Informatics*), Srednjoevropska informatička olimpijada (*Central European Olympiad in Informatics*, CEOI), kao i Azijsko-pacifička informatička olimpijada (*Asia-Pacific Olympiad in Informatics*, APIO).

Međunarodno studentsko takmičenje iz programiranja (*International Collegiate Programming Contest*, ICPC) jeste godišnje takmičenje programera studenata. Svaki tim čine tri studenta ali, za razliku od IOI takmičenja, studenti rade kao tim, a svaki tim ima na raspolaganju samo jedan računar.

Međunarodno studentsko takmičenje iz programiranja ima nekoliko etapa, a najbolji timovi se na kraju pozivaju na Svetsko finale. Dok u prvoj fazi takmičenja može da

učestvuje i nekoliko desetina hiljada učesnika, u finalu ima mesta samo za mali broj[1] najboljih, tako da je sam ulazak u finale veliki uspeh.

Na svakom ICPC takmičenju timovi imaju 5 sati da reše desetak algoritamskih problema. Rešenje se prihvata samo ukoliko se efikasno reše svi test primeri. Tokom takmičenja učesnici mogu da vide rezultate drugih timova, ali tokom poslednjeg sata, semafor je zamrznut i nije moguće videti rezultate poslednje predatih verzija rešenja.

Onlajn takmičenja Postoje i mnogobrojna onlajn takmičenja koja su otvorena za sve. U ovom trenutku, najaktivniji sajt za takmičenje je Codeforces, na kome se organizuju takmičenja jednom nedeljno. Osim ovog sajta popularni su još i *AtCoder, CodeChef, CS Academy, HackerRank* i *Topcoder*.

Neke kompanije organizuju onlajn takmičenja sa pravim finalima uživo na nekom dogovorenom mestu. Primeri ovakvih takmičenja su Facebook Hacker Cup, Google Code Jam i Yandex.Algorithm. Naravno, velike kompanije koriste ovakva takmičenja i za regrutovanje novih kadrova: uspeh na takmičenjima je odličan način za demonstriranje programerskih veština.

1.1.2 Saveti za vežbanje

Učenje takmičarskog programiranja predstavlja ozbiljan posao. Međutim, postoji mnogo načina za vežbanje i oni se, između sebe, razlikuju po efikasnosti.

Prilikom rešavanja problema uvek treba imati u vidu da broj rešenih problema nije toliko važan koliko je važan njihov *kvalitet*. Nekada je primamljivo izabrati probleme koji lepo izgledaju i lako se rešavaju, a preskočiti probleme koji izgledaju teško i nezanimljivo. Međutim, upravo fokusiranje na ove teške probleme može vam pomoći u popravljanju programerskih veština.

Važno je napomenuti da se većina takmičarskih problema može rešiti korišćenjem jednostavnih i kratkih algoritama, ali teži deo posla predstavlja osmišljavanje algoritma. Takmičarsko programiranje, prema tome, nije učenje složenih algoritama napamet, već učenje odgovarajućeg pristupa problemima, kako bi se oni rešili korišćenjem jednostavnih alata.

Konačno, neki programeri preziru implementiranje algoritama: njima je zabavan dizajn algoritma, a dosadna njegova implementacija. Međutim, sposobnost brzog i tačnog implementiranja algoritama je veoma značajna i ova veština se može usavršiti. Sa druge strane, pogrešan pristup je kada se na takmičenjima najviše vremena posveti pisanju koda i otklanjanju grešaka, umesto razmišljanju o načinima na koje se konkretan problem može rešiti.

1 Tačan broj mesta u finalu razlikuje se od godine do godine. Na primer, 2017. godine bilo je ukupno 133 mesta za finaliste.

1.2 O knjizi

S obzirom na to da nastavni plan *IOI Syllabus* [15] reguliše teme na Međunarodnoj Informatičkoj Olimpijadi, upravo ovaj plan je predstavljao polaznu tačku prilikom izbora tema koja će biti obrađene u ovoj knjizi. Međutim, ovde su obrađene i neke napredne teme koje su (od 2017. godine) izbačene sa IOI takmičenja, ali se pojavljuju na nekim drugim takmičenjima. Primeri ovakvih tema su maksimalni tokovi, nim teorija i nizovi sufiksa.

Iako su mnogi aspekti takmičarskog programiranja objašnjeni u standardnim udžbenicima o algoritmima, postoje i značajne razlike. Primera radi, u mnogim udžbenicima fokus je stavljen na implementiranje algoritama sortiranja i fundamentalnih struktura podataka od samog početka. Međutim ovo znanje nije previše relevantno u takmičarskom programiranju, zato što postoji mogućnost korišćenja funkcionalnosti standardne biblioteke. Osim toga, u ovoj knjizi obrađene su i teme koje su dobro poznate u takmičarskoj zajednici, ali se o njima retko govori u udžbenicima. Primer ovakve teme su segmentna stabla, koja se mogu koristiti za rešavanje velikog broja problema, čije bi rešavanje, inače, zahtevalo veoma komplikovan algoritam.

Jedan od ciljeva prilikom pisanja ove knjige bilo je i *dokumentovanje* tehnika takmičarskog programiranja o kojima se inače govori samo na onlajn forumima i blogovima. Kad god je to bilo moguće, pozvali smo se na naučne reference za metode koje su karakteristične za takmičarsko programiranje. Međutim, to u mnogim slučajevima i nije bilo moguće zato što za mnoge tehnike koje su sada deo folklora takmičarskog programiranja, nije moguće sa sigurnošću precizirati ko ih je i kada originalno otkrio.

U nastavku teksta možete da vidite kako izgleda struktura ove knjige:

- U poglavlju 2 najpre su prikazane karakteristike programskog jezika C++, a zatim rekurzivni algoritmi i manipulacija bitovima.
- U poglavlju 3 bavićemo se problemom efikasnosti algoritama, odnosno kreiranjem algoritama koji brzo mogu da obrade velike skupove podataka.
- U poglavlju 4 prikazali smo algoritme sortiranja i binarne pretrage, fokusirajući se na njihovu primenu u dizajnu algoritma.
- U poglavlju 5 prikazane su razne strukture podataka standardne C++ biblioteke, kao što su vektori, skupovi i mape.
- Poglavlje 6 predstavlja uvod u tehniku dizajna algoritma nazvanu dinamičko programiranje. Ovde su navedeni primeri problema koji se primenom ove tehnike mogu rešiti.
- U poglavlju 7 prikazani su osnovni grafovski algoritmi, kao što su pronalaženje najkraćih puteva i minimalnih razapinjućih stabala.
- U poglavlju 8 upoznaćete se sa nekim naprednim temama dizajna algoritama, kao što su bitovska paralelizacija i amortizovana analiza.
- U poglavlju 9 bavićemo se efikasnim procesiranjem upita nad nizovnim opsezima, kao što su, na primer, izračunavanje sume vrednosti i određivanje minimalnih vrednosti.
- U poglavlju 10 su prikazani specijalizovani algoritmi za rad sa stablima, uključujući i metode za obradu upita nad stablima.

- U poglavlju 11 biće reči o matematičkim temama koje su relevantne za takmičarsko programiranje.
- U poglavlju 12 prikazane su napredne grafovske tehnike, kao što su jako povezane komponente ili maksimalni protoci.
- U poglavlju 13 bavićemo se geometrijskim algoritmima i biće prikazane odgovarajuće tehnike za rešavanje geometrijskih problema.
- U poglavlju 14 biće reči o tehnikama u vezi sa niskama, kao što su string heširanje, Z-algoritam i korišćenje nizova sufiksa.
- U poglavlju 15 prikazane su još neke napredne tehnike, kao što su algoritmi kvadratnog korena ili optimizacija rešenja baziranih na dinamičkom programiranju.

1.3 Skup problema CSES

Skup CSES (*CSES Problem Set*) predstavlja skup problema koji se mogu koristiti za vežbanje takmičarskog programiranja. Problemi su poređani prema težini, a sve tehnike koje su neophodne za njihovo rešavanje obrađene su u ovoj knjizi. Ovaj skup problema možete preuzeti na sledećoj adresi:

https://cses.fi/problemset/

Pogledajmo sada kako se rešava prvi problem u ovom skupu problema po imenu Čudni algoritam (*Weird Algorithm*). Problem glasi ovako:

Razmotrite algoritam koji kao ulaz uzima pozitivni celi broj n. Ukoliko je u pitanju paran broj, algoritam ga deli sa dva, a ako je neparan, algoritam ga množi sa tri i dodaje vrednost jedan. Algoritam ponavlja ovaj postupak sve dok n ne bude 1. Primera radi, sekvenca za vrednost $n = 3$ izgleda ovako:

$$3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

Vaš zadatak je da simulirate izvršenje algoritma za zadatu vrednost parametra n.

Ulaz
Jedina ulazna linija sadrži celi broj n.
Izlaz
Štampanje reda koji sadrži sve vrednosti parametra n tokom izvršenja algoritma.
Ograničenja
- $1 \leq n \leq 10^6$

Primer
Ulaz:
```
3
```
Izlaz:
```
3 10 5 16 8 4 2 1
```

Ovaj problem povezan je sa čuvenim Kolacovim problemom (*Collatz conjecture*), prema kome se prethodni algoritam završava za svaku vrednost parametra n. Doduše, to niko do sada nije uspeo da dokaže. U ovom problemu znamo da inicijalna vrednost parametra n neće biti veća od jednog miliona, što ovaj problem čini mnogo lakšim za rešavanje.

U pitanju je problem jednostavne simulacije koji ne zahteva mnogo razmišljanja. Evo mogućeg rešenja problema u programskom jeziku C++:

```
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    while (true) {
        cout << n << " ";
        if (n == 1) break;
        if (n%2 == 0) n /= 2;
        else n = n*3+1;
    }
    cout << "\n";
}
```

Kôd najpre čita ulaznu vrednostn, zatim simulira algoritam i štampa vrednost parametra n nakon svakog koraka. Jednostavno je proveriti da li algoritam ispravno rešava slučaj u kome je vrednost parametra $n = 3$, koji je zadat u iskazu problema.

Sada je trenutak za predavanje koda u sistem CSES. Nakon toga, kôd će biti kompajliran i proveren korišćenjem sistema za testiranje. Nakon svake pojedinačne provere, CSES nam govori da li je naš kôd prošao proveru ili ne. Isto tako, možemo ispitati ulaz, očekivani izlaz, kao i izlaz koji je izbacio naš kôd.

Nakon testiranja našeg koda, sistem CSES izbacuje izveštaj sledeće sadržine:

test	verdict	time (s)
#1	ACCEPTED	0.06 / 1.00
#2	ACCEPTED	0.06 / 1.00
#3	ACCEPTED	0.07 / 1.00
#4	ACCEPTED	006 / 1.00
#5	ACCEPTED	0.06 / 1.00
#6	TIME LIMIT EXCEEDED	– / 1.00
#7	TIME LIMIT EXCEEDED	– / 1.00
#8	WRONG ANSWER	0.07 / 1.00
#9	TIME LIMIT EXCEEDED	– / 1.00
#10	ACCEPTED	0.06 / 1.00

Prema ovom izveštaju, naš kôd je prošao pojedine testove (ACCEPTED), u nekim testovima je bio prespor (TIME LIMIT EXCEEDED), dok je u drugim dobio neispravan izlaz (WRONG ANSWER). Ovo je baš iznenađujuće!

U prvom slučaju u kome je kôd bio neispravan, parametar *n* je imao vrednost 138367. Ukoliko bismo ga proverili lokalno korišćenjem ovog ulaza, ispostavilo bi se da je naš kôd zaista spor. Štaviše, on se nikada ne završava.

Razlog zbog koga je naš kôd zakazao jeste taj da promenljiva *n* tokom simulacije može dobiti veoma veliku vrednost. Konkretno, ova vrednost može postati veća od gornje granice promenljive tipa `int`. Da bismo ispravili ovaj problem, dovoljno je da naš kôd izmenimo tako da promenljivoj n dodelimo tip `long long`. Nakon ove izmene, dobićemo željeni rezultat:

test	verdict	time (s)
#1	ACCEPTED	0.05 / 1.00
#2	ACCEPTED	0.06 / 1.00
#3	ACCEPTED	0.07 / 1.00
#4	ACCEPTED	0.06 / 1.00
#5	ACCEPTED	0.06 / 1.00
#6	ACCEPTED	0.05 / 1.00
#7	ACCEPTED	0.06 / 1.00
#8	ACCEPTED	0.05 / 1.00
#9	ACCEPTED	0.07 / 1.00
#10	ACCEPTED	0.06 / 1.00

Kao što ste u ovom primeru mogli da vidite, čak i sasvim jednostavni algoritmi mogu u sebi sadržati sitne manjkavosti. U tom smislu, takmičarsko programiranje nas uči pisanju algoritama koji zaista funkcionišu.

1.4 Ostali resursi

Osim ove, postoje i neke druge knjige koje se bave temom takmičarskog programiranja. Knjiga *Programerski izazovi* [28] iz 2003. godine, autora Skiena i Revila, predstavlja pionirsko delo u ovoj oblasti. Nešto je aktuelnija knjiga *Takmičarsko programiranje* 3 [14], autora Halim i Halim. Obe ove knjige namenjene su čitaocima bez prethodnog iskustva u takmičarskom programiranju.

Tražite izazov? [7] je naprednija knjiga u kojoj postoji skup teških problema sa takmičenja poljskih programera. Najzanimljivija karakteristika ove knjige jeste to što u njoj

postoji detaljna analiza načina rešavanja problema. Ova knjiga namenjena je iskusnim programerima takmičarima.

Knjige o algoritmima takođe predstavljaju zanimljivo štivo za programere takmičare. Najsveobuhvatnija ovakva knjiga jeste *Uvod u algoritme* [6], koju su napisali Kormen, Lejerson, Rivest i Štajn, tim autora poznat i pod pseudonimom CLRS. Ova knjiga predstavlja odličan resurs ukoliko želite da proverite sve detalje u vezi sa nekim algoritmom, kao i da dokažete njegovu ispravnost.

Knjiga *Dizajn algoritma* [19], autora Klajnberga i Tardoa fokusira se na tehnikama dizajna algoritama i iscrpno analizira metod *podeli i vladaj*, pohlepne algoritme, dinamičko programiranje i algoritme maksimalnog protoka. Nešto je praktičnija knjiga *Uputstvo za dizajn algoritama* [27], autora Skiene, zato što u njoj postoji obiman skup računskih problema, a opisani su i načini za njihovo rešavanje.