Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Specifikacija projektnog zadatka

Studenti:	Dimitrije Gašić	SV31-2021
	Maša Ivanov	SV54-2021
Predmet:	Paralelno programiranje	
Tema projekta:	Generator izraza za igru Moj broj	

Sadržaj

Analiza problema

Cilj igre Moj broj je pronalaženje izraza od ponuđenih brojeva čija vrednost je najbliža zadatom rešenju. Tipična igra sadrži 4 jednocifrena, 2 dvocifrena i jedno trocifreno rešenje. U ovom projektu smo se fokusirali na deo kompjuterskog rešavanja ovog problema tako da smo izbacili mogućnost ljudskih igrača. Program za više ulaza poziva algoritam koji traži najbolji izraz za svaki od njih.

Pretraga najboljeg izraza se vrši generisanjem svih mogućih izraza pomoću kombinacija ulaznih brojeva (bez ponavljanja). Najbolji izraz je onaj čija je vrednost najbliža rezultatu i koji koristi najmanje ponuđenih brojeva. Ulazni brojevi i rešenja moraju biti celi brojevi, a izrazi se formiraju samo pomoću prostih operacija i zagrada. Ponuđeni brojevi i rešenja se učitavaju iz ulazne datoteke.

Pored zahteva za pronalaženjem najboljeg izraza, algoritam treba da zadovolji i nefunkcionalni zahtev za razumno vreme izvršavanja što direktno proizilazi iz broja izraza koji se generišu kao i načina evaluacije izraza (da li se koristi eksterni modul itd).

Koncept rešenja

Generalno, brute-force pristup generisanja svih permutacija operacija, zagrada i brojeva bi bio spor ne samo zbog nedostatka optimizacija koje uklanjaju izraze koji nemaju smisla (sabiranje/oduzimanje nulom i množenje/deljenje jedinicom), već i zbog potrebe da se svaki generisani izraz evaluira pomoću eksterne metode za računanje vrednosti.

Kako bismo izbegli ove probleme, osmislili smo alternativno rešenje koje se bazira na konceptu postfiksne notacije. Ovim je omogućeno da se generisanje izraza obavlja dodavanjem brojeva i operacija na prethodni izraz. Kod postfiksne notacije operacija se piše nakon njenih operanada. Ovo takođe uklanja potrebu za algoritamskim računanjem generisanih rešenja zato što je to moguće uraditi i pre generisanja (izmena prethodne vrednosti pomoću novog broja koji se dodaje). Takođe u procesu generisanja više nije postojala potreba za zagradama što je proizvelo dodatno ubrzanje.

Algoritam započinje generisanje izraza biranjem jednog od ponuđenih brojeva i dodavanjem operacije i drugog broja na njega. Na početni izraz se u rekurziji dodaje sledeći iz skupa ponuđenih brojeva i sledeća operacija i tako dok ne “isprazni” skup ponuđenih brojeva (koji prosleđujemo rekurziji). Ukratko, svaki poziv rekurzije generiše 6 podslučajeva (rekurzija) koji predstavljaju izraze dobijene sabiranjem, množenjem, oduzimanjem i deljenjem. Povratna vrednost svake rekurzije predstavlja najbolje lokalno rešenje od svih generisanih u podrekurzijama.

Prilikom generisanja se izbegavaju i rešenja koja nisu validna (deljenje nulom i deljenje sa ostatkom), kao i ponovljena rešenja (korišćenje komutativnih operacija sa različitih strana). Kod oduzimanja i deljenja je bitno od čega oduzimamo i delimo dok kod sabiranja i množenja dovoljno je da dodamo broj samo na kraj izraza. Zbog toga svaki slučaj koji koristi oduzimanje i deljenje ima dva podslučaja za obe strane.

Radi boljeg prikazivanja dobijenih rezultata i njihovog upoređivanja, pored početnog slučaja od 6 brojeva dodali smo i ulaze od 5, 7 i 8 brojeva.

Programsko rešenje

Program se sastoji iz 2 modula ne uključujući onaj za pokretanje (***Source.cpp***). Modul ***Calculator.cpp*** služi potrebe prevođenja postfiksne notacije u infiksnu i pored pomoćnih metoda sadrži samo jednu koja se koristi eksterno. Pronalaženjem rešenja se bavi modul ***Generate.cpp*** i sadrži metode za serijsko i paralelno izvršavanje, kao i strukturu Expression koja modeluje rešenja.

Serijski algoritam

Osnovni slučaj algoritma se izvršava na jednom procesoru i zbog toga serijska verzija ovog algoritma izvršava pozive rekurzija redosledno jednu za drugom bez potrebe za generisanjem niti i zadataka. Funkcije koje implementiraju serijski algoritam su sledeće:

Expression generateSerial(vector<int> round) - ulazna tačka algoritma; prima ponuđene brojeve i rešenje koje se traži. Ova funkcija prolazi kroz sve ponuđene brojeve i kreira početni izraz za svaki od njih koji prosleđuje rekurziji za generisanje izraza.

Expression serialRecursion(const vector<int>& factors, const Expression& current, int solution) - generiše nove izraze na osnovu prosleđenog (current) tako što prolazi kroz ponuđene brojeve (factors) i kombinuje ih sa operacijama. Broj novokreiranih izraza je 6 iz razloga opisanih u Konceptu rešenja. Kombinovanje starog izraza sa novim brojem i operacijom vrše metode u strukturi **Expression** koje za svaku operaciju generišu novi izraz i menjaju staru vrednost u skladu sa slučajem. Novogenerisani izrazi se potom prosleđuju rekurziji sve dok ne ponestane ponuđenih brojeva. Funkcija **checkSolution** se koristi za proveru svakog generisanog izraz i zamenu trenutno najboljeg u slučaju pronalaska boljeg. Na kraju funkcije se vraća lokalno najbolje rešenje.

Paralelni algoritam

Budući da većina modernih procesora ima više jezgara, unapređeni algoritam koristi biblioteku za paralelno programiranje ***tbb*** prilikom kreiranja niti i zadataka. Većina koda je slična serijskom algoritmu sa par bitnih izmena u ulaznoj funkciji i načinu pozivanja rekurzija za generisanje izraza.

Expression generateParallel(vector<int> round) - ulazna tačka paralelnog algoritma koja iterira kroz ponuđene brojeve korišćenjem funkcije **parallel_for** iz ***tbb*** modula pomoću strukture **BestExpression**. Za svaki broj poziva funkciju **parallelRecursion** i na kraju bira najbolji izraz od dobijenih.

Expression parallelRecursion(const vector<int>& factors, const Expression& current, int solution) - ima sličnu strukturu funkciji **serialRecursion** uz dodatak **task_group** iz modula ***tbb*** koji se koristi za pravljenje zadataka koji izvršavaju unutrašnje pozive rekurzija. Prilikom iteracije kroz ponuđene brojeve za svaki se paralelno izvršava 6 poziva rekurzija (za svaki slučaj operacije) i na kraju se svaki od dobijenih izraza proverava u funkciji **checkSolution** koja eventualno ažurira trenutno najbolje. Zbog bolje optimizacije je uveden globalni parametar **CUTOFF** koji definiše na kom nivou stabla rekurzije počinje da se poziva serijska rekurzija. Empirijskim putem je pronađeno da je optimalna vrednost za cutoff oko 4 ali detaljnija analiza sledi u poglavlju Ispitivanja.

Ispitivanja

Specifikacije računara na kom se pokretao program su sledeće:

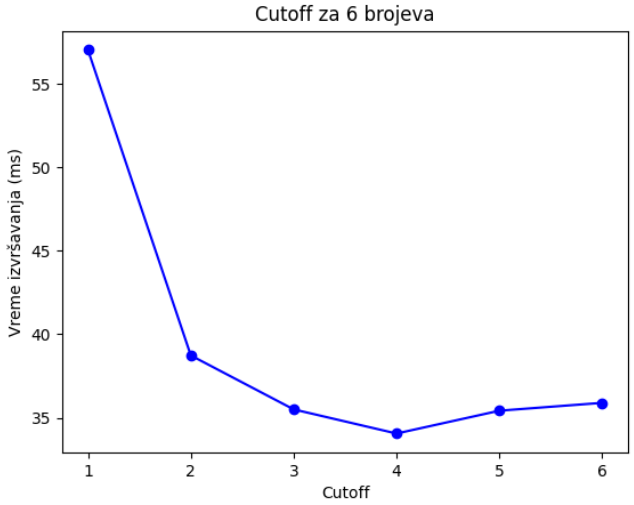
- Procesor: Intel Core(TM) i7-4850HQ (2.30GHz)
    - Broj jezgara: 4
- RAM: 16.0 GB
- Operativni sistem: Windows 10 Home

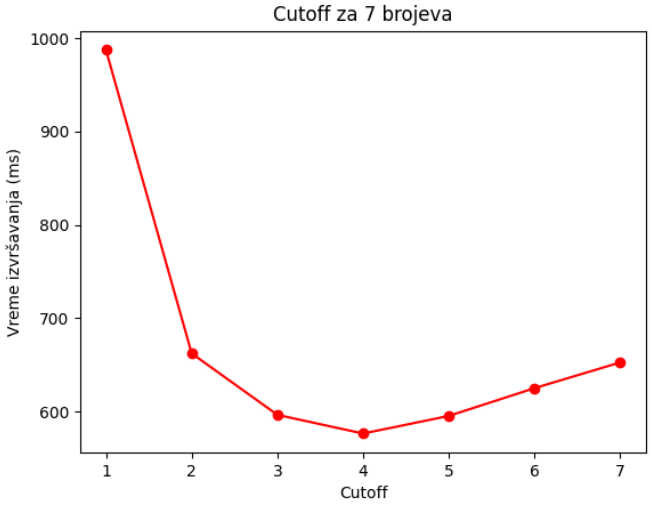
Tabela Cutoff-ova

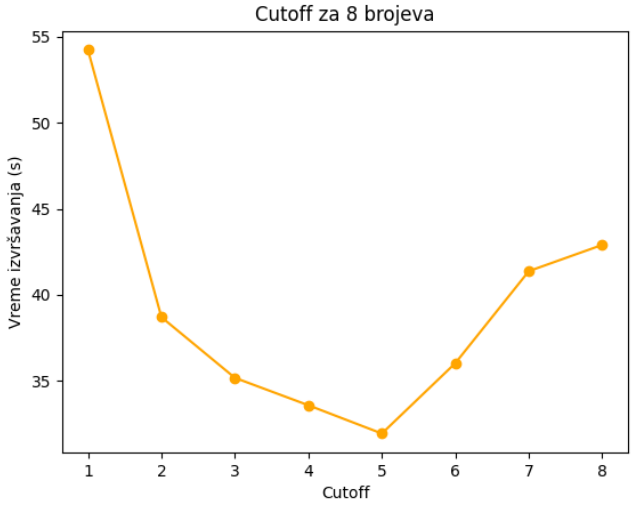
Cutoff	6 brojeva (ms)	7 brojeva (ms)	8 brojeva (s)
1	57.053	987.459	54.218
2	38.733	662.191	38.706
3	35.498	596.322	35.185
4	34.050	576.294	33.594
5	35.420	595.186	31.952
6	35.891	624.960	36.038
7	/	652.550	41.378
8	/	/	42.902

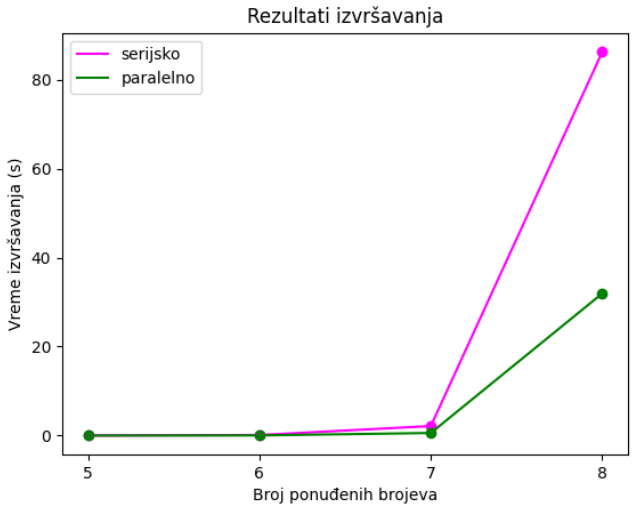
Tabela rezultata

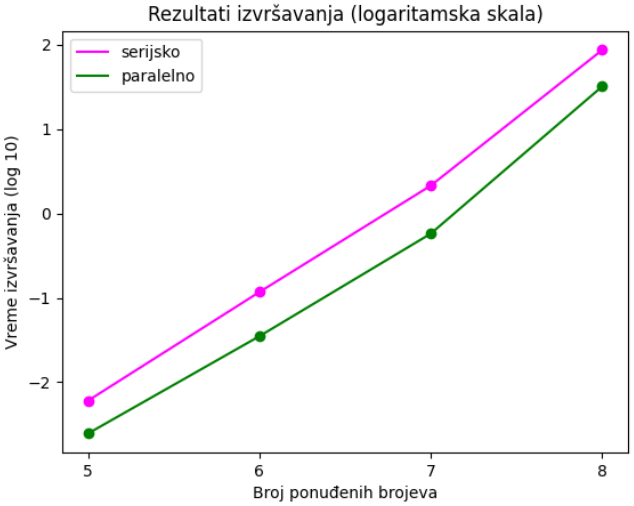
Ponuđeno brojeva	Prosečno generisanih izraza	Serijsko (s)	Paralelno (s)	Ubrzanje
5	18 186	0.00605	0.00247	2.44
6	292 821	0.11706	0.03510	3.33
7	5 389 768	2.13530	0.57629	3.71
8	175 279 198	86.24911	31.95233	2.70

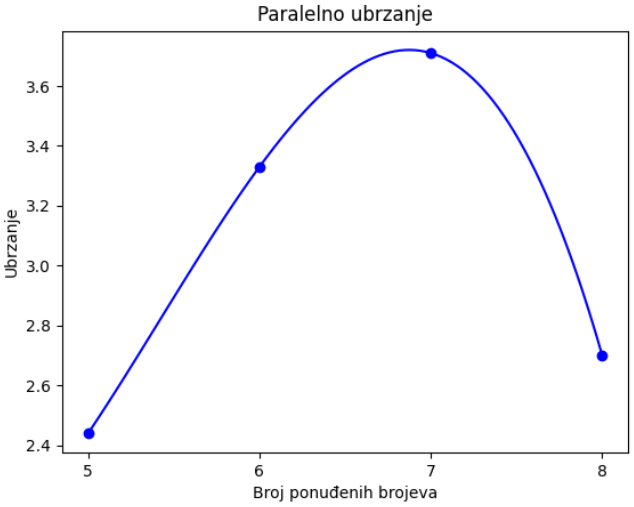

Cutoff za 6 brojeva
Vreme izvršavanja (ms)
55
50
45
40
35
1
2
3
4
5
6
Cutoff


Cutoff za 7 brojeva
Vreme izvršavanja (ms)
1000
900
800
700
600
1
2
3
4
5
6
7
Cutoff


Cutoff za 8 brojeva
Vreme izvršavanja (s)
55
50
45
40
35
1
2
3
4
5
6
7
8
Cutoff


Rezultati izvršavanja
serijsko
paralelno
Vreme izvršavanja (s)
0
20
40
60
80
5
6
7
8
Broj ponuđenih brojeva


Rezultati izvršavanja (logaritamska skala)
serijsko
paralelno
Vreme izvršavanja (log 10)
2
1
0
−1
−2
5
6
7
8
Broj ponuđenih brojeva


Paralelno ubrzanje
Ubrzanje
2.4
2.6
2.8
3.0
3.2
3.4
3.6
5
6
7
8
Broj ponuđenih brojeva

Analiza rezultata

Prilikom testiranja, program je pokretan sa različitim ulaznim parametrima. Jedan od njih je Cutoff koji predstavlja nivo do kog se algoritam izvršava serijski, a nakon njega paralelno. Sa tabele i grafika možemo videti da su najbolji rezultati za 6 i 7 ulaznih brojeva postignuti ukoliko je Cutoff 4 dok je za 8 brojeva najbolje vreme izvršavanja sa Cutoff-om 5.

Merenja vremena izvršavanja algoritma u zavisnosti od broja ulaznih brojeva su uključivala ulaze od 5, 6, 7 i 8 brojeva. Kao što može da se primeti na grafiku Rezultata izvršavanja skoro je nemoguće primetiti odnos serijskog i paralelnog izvršavanja zbog reda veličine. Da bismo prikazali taj odnos koristili smo logaritamsku skalu sa osnovom deset na 2. grafiku rezultata.

Paralelno ubrzanje smo dobili deljenjem serijskog vremena izvršenja paralelnim. Prosečno ubrzanje koje smo postigli je 3. Za ulaz od 5 brojeva je ubrzanje bilo dosta manje od onog za ulaz od 6 i 7 brojeva zbog manje isplativosti paralelizma (nema dovoljno rada). Za ulaz od 8 brojeva, ubrzanje takođe bilo manje od prosečnog zbog velikog broja generisanih zadataka i korišćenja memorijskih resursa.