Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Ivan Vučica

Druga domaća zadaća iz predmeta "Uvod u teoriju računarstva"

Zadatak broj 2048

Zagreb, lipanj 2008.

Druga domaća zadaća iz predmeta "Uvod u teoriju računarstva"

Student: Ivan Vučica

Matični broj studenta: 0036431411

Zadatak broj 2048: Napisati program za simulaciju rada Turingovog stroja koji provjerava da li je zapisani dekadski broj palindrom, odnosno da li je simetričan (primjer simetričnih brojeva: 034565430, 789987).

Uvod

Turingov stroj jednostavan je matematički model s mogućnostima sličnima onih računala općenite primjene. Teoretski, uz dovoljno dugačku ulazno-izlaznu traku i dovoljan broj stanja, moguće je izgraditi Turingov stroj koji će moći izvršiti bilo koji algoritam.

Turingov stroj ima skup stanja, trenutno stanje, skup mogućih znakova trake, traku ili više njih ograničene ili neograničene duljine, pokazivač na trenutni znak pojedine trake odnosno glavu, te funkciju prijelaza ovisnu o trenutnom stanju i o znaku na koji trenutno pojedina glava pokazuje.

Zadani zadatak tražio je određivanje simetričnosti ulaznih znakova iz skupa dekadskih znamenki. Ne traži se očuvanje ulaza, niti se specificira da je potrebno izlaz dati na samu traku. Nije specificiran broj traka. Ne specificira se na kojoj udaljenosti od početka trake mora počinjati sam ulazni niz.

Ulazni znakovi, odnosno dekadski zapisan broj, je simetričan iliti palindrom ako za svaki znak koji pročitamo s ulaza, na kraju možemo pronaći odgovarajući ulazni znak, te ako možemo nastaviti iterirati postupak sve dok ne ispraznimo sve znakove. Postupak se prekida stanjem greške ukoliko za pojedini znak s početka niza nismo uspjeli pronaći odgovarajući znak na kraju niza.

Ostvarenje

Rješenju se pristupilo u dva koraka: prvi je bio određivanje potrebnih stanja, ulaznih znakova te samih prijelaza; odnosno, prvi korak bio je općenito dizajniranje funkcije prijelaza.

Nakon dizajna funkcije prijelaza pristupilo se dizajnu simulatora, te konačno modifikaciji dobivenog relativno generičkog simulatora implementacijom zadane funkcije stanja.

Dizajn funkcije prijelaza

Prvi korak u ostvarenju funkcije prijelaza bilo je smišljanje osnovnog načina rada. Pri razvoju Turingovog stroja, pruža se nekoliko mogućnosti. Da li se treba koristiti više glava za čitanje jedne trake? Da li se treba koristiti više traka? Da li je možda dovoljna samo jedna traka? Kako ostvariti pamćenje pojedine ćelije prilikom kretanja po traci?

Za potrebe ovog studentskog projekta, izrađen je model Turingova stroja s jednom trakom.

Zatim, smišljen je osnovni način rada. Pretpostavimo da je na početku glava pozicionirana na desnoj strani, a da je prije samog ulaznog niza jedna praznina, B. Način rada bazira se na tome da pretpostavimo da broj jest palindrom, a zatim to pokušavamo opovrgnuti. Skica je slijedeća:

1. qS - Stroj primijeti da je glava trenutno na praznini B. Glava se pomiče za jedno mjesto udesno i odlučuje da je slijedeći korak qX određivanje koji znak je na trenutno prvom mjestu u nizu
2. qX – Određuje se koji je znak trenutno na prvom mjestu u nizu. Taj znak se briše jer više nije važan, a njegovu vrijednost zasad pamtimo u stanju. Sljedeći korak je ustvari jednak za deset stanja (q0-q9), no nosi dodatnu informaciju o tome koji znak je dosad bio na prvom mjestu. Također, ukoliko pročitamo prazninu, to znači da više nemamo brojeva za obradu, pa budući da nemamo više što obraditi, a nismo uspjeli dokazati da broj nije palindrom, završavamo s uspjehom.
3. q0-q9 – Sva ova stanja su ustvari međusobno identična osim što nose dodatnu informaciju podatka koji je pročitan s prvog mjesta u nizu (a koji sada više ne postoji). Koji god znak se

pročitao dok je stroj u tim stanjima, stanje ostaje jednako i znak ostaje jednak, samo se pomičemo u desno. To završava tek kad se pročita praznina. Tada se pomaknemo ulijevo i prijeđemo u odgovarajuće stanje q0c-q9c.

4. q0c-q9c – Stroj ima valjani prijelaz jedino ako je znak ispod glave jednak onom ulaznom. Tada pobrišemo znak ispod glave i nastavljamo provjeru ispočetka; pomoću stanja qS odnosno koraka 1. u ovom pregledu, pomičemo glavu do prve praznine ulijevo, koja je sada točno na mjestu gdje je nekad stajao prvi ulazni znak. Također, ako pročitamo prazninu, to znači da nemamo više podataka za obradu i završavamo obradu s uspjehom jer nismo uspjeli dokazati da broj nije palindrom.

Navedeni opis može se lijepo prikazati tablicom prijelaza:

	0	1	2	3	4	5	6	7	8	9	B
qS	qS, 0, L	qS, 1, L	qS, 2, L	qS, 3, L	qS, 4, L	qS, 5, L	qS, 6, L	qS, 7, L	qS, 8, L	qS, 9, L	qX, B, R
qX	q0, B, R	q1, B, R	q2, B, R	q3, B, R	q4, B, R	q5, B, R	q6, B, R	q7, B, R	q8, B, R	q9, B, R	qU, B, R
q0	q0, 0, R	q0, 1, R	q0, 2, R	q0, 3, R	q0, 4, R	q0, 5, R	q0, 6, R	q0, 7, R	q0, 8, R	q0, 9, R	q0c, B, L
q1	q1, 0, R	q1, 1, R	q1, 2, R	q1, 3, R	q1, 4, R	q1, 5, R	q1, 6, R	q1, 7, R	q1, 8, R	q1, 9, R	q1c, B, L
q2	q2, 0, R	q2, 1, R	q2, 2, R	q2, 3, R	q2, 4, R	q2, 5, R	q2, 6, R	q2, 7, R	q2, 8, R	q2, 9, R	q2c, B, L
q3	q3, 0, R	q3, 1, R	q3, 2, R	q3, 3, R	q3, 4, R	q3, 5, R	q3, 6, R	q3, 7, R	q3, 8, R	q3, 9, R	q3c, B, L
q4	q4, 0, R	q4, 1, R	q4, 2, R	q4, 3, R	q4, 4, R	q4, 5, R	q4, 6, R	q4, 7, R	q4, 8, R	q4, 9, R	q4c, B, L
q5	q5, 0, R	q5, 1, R	q5, 2, R	q5, 3, R	q5, 4, R	q5, 5, R	q5, 6, R	q5, 7, R	q5, 8, R	q5, 9, R	q5c, B, L
q6	q6, 0, R	q6, 1, R	q6, 2, R	q6, 3, R	q6, 4, R	q6, 5, R	q6, 6, R	q6, 7, R	q6, 8, R	q6, 9, R	q6c, B, L
q7	q7, 0, R	q7, 1, R	q7, 2, R	q7, 3, R	q7, 4, R	q7, 5, R	q7, 6, R	q7, 7, R	q7, 8, R	q7, 9, R	q7c, B, L
q8	q8, 0, R	q8, 1, R	q8, 2, R	q8, 3, R	q8, 4, R	q8, 5, R	q8, 6, R	q8, 7, R	q8, 8, R	q8, 9, R	q8c, B, L
q9	q9, 0, R	q9, 1, R	q9, 2, R	q9, 3, R	q9, 4, R	q9, 5, R	q9, 6, R	q9, 7, R	q9, 8, R	q9, 9, R	q9c, B, L
q0c	qS, B, L	-	-	-	-	-	-	-	-	-	qU, B, R
q1c	-	qS, B, L	-	-	-	-	-	-	-	-	qU, B, R
q2c	-	-	qS, B, L	-	-	-	-	-	-	-	qU, B, R
q3c	-	-	-	qS, B, L	-	-	-	-	-	-	qU, B, R
q4c	-	-	-	-	qS, B, L	-	-	-	-	-	qU, B, R
q5c	-	-	-	-	-	qS, B, L	-	-	-	-	qU, B, R
q6c	-	-	-	-	-	-	qS, B, L	-	-	-	qU, B, R
q7c	-	-	-	-	-	-	-	qS, B, L	-	-	qU, B, R
q8c	-	-	-	-	-	-	-	-	qS, B, L	-	qU, B, R
q9c	-	-	-	-	-	-	-	-	-	qS, B, L	qU, B, R
qU	USPJEH										

Dizajn simulatora

Simulator je izgrađen tako da se stanja popišu na jedno mjesto u kodu (što nije nužno, no radi preglednosti je korisno). Zatim se u jednoj funkciji postavlja tablica stanja. Za svaku kombinaciju stanja i ulaznog znaka možemo imati određenu prijelaznu funkciju, dakle tri parametra: novi znak koji pišemo na trenutnu lokaciju, novo stanje i smjer pomaka glave za jedno mjesto. Ako kombinacija ne postoji, stroj prelazi u stanje greške.

Za naš slučaj, ovaj prelazak u stanje greške znači kraj simulacije uz poruku da nije otkriven palindrom.

Algoritam je vrlo jednostavan: repetitivno ponavljamo programsku funkciju koja obavlja prijelaz, dok god ona ne vrati logičku vrijednost laž, odnosno dok nam ne signalizira da je obrada gotova. Kad je obrada gotova, (otprilike) provjeravamo da li je zadnje stanje bilo stanje qU odnosno uspjeh. Za naš zadatak, ovisno o konačnom stanju ispisujemo poruku da li je broj palindrom ili nije.

Unutar programske funkcije za obavljanje prijelaza gledamo da li u tablici prijelaza postoji traženi element na osnovu trenutnog stanja i znaka na koji pokazuje ulazna glava. Ako postoji, izvršavamo "naredbu" koja je zapisana na toj ćeliji u tablici (tj., mijenjamo trenutno stanje, mijenjamo znak ispod glave i pomičemo glavu). Inače, postavljamo stanje greške i javljamo dovršetak. Zatim, provjerimo da li smo kojim slučajem u stanju uspjeha. Ako smo u tom stanju, postavljamo uspjeh i javljamo dovršetak.

Ukoliko se glava pokuša pomaknuti nadesno "izvan trake", traka (koja je simulirana stringom) se proširuje za jedan znak koji se postavlja na znak za prazninu B. Time se simulira beskonačnost trake udesno.

Ostale funkcije su pomoćne i koriste se uglavnom za jednostavno korisničko sučelje.

Simulator je razvijan na GNU/Linux operativnom sustavu i nije testiran na Windows operativnom sustavu.

Zaključak

Već i jednostavni algoritmi, poput traženog, na Turingovom stroju postaju srednje zahtjevni. Ograničenost ovog stroja ostavlja ga daleko izvan domene praktičnog, te kao jedan jednostavan matematički princip i model za rješavanje zadataka. Za sve druge uporabe, i amaterske i komercijalne i znanstvene, digitalno računalo je neusporedivo bolje rješenje; primjerice po 255 stanja u svakom od nekoliko milijardi okteta s kojima današnja kućna računala standardno raspolažu (pa ako uračunamo i sekundarnu memoriju, i nekoliko stotina milijardi okteta) pruža mnogo veću slobodu u razvoju algoritama i softvera za konačnog korisnika.

Turingov stroj je interesantan iz čisto povijesnih razloga, te kao zanimljiv misaoni eksperiment.