Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Mihael Presečan

Druga domaća zadaća iz predmeta „Uvod u teoriju računarstva“

Zadatak broj 3063.

Zagreb, lipanj 2011.

Druga domaća zadaća iz predmeta „Uvod u teoriju računarstva“

Student: Mihael Presečan

Matični broj studenta: 0036458060

Zadatak broj 3063: $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$

Definirati (nije potrebno programski ostvariti) Turingov stroj proizvoljna oblika koji koristi navedeni izraz za određivanje vrijednosti broja e^x na beskonačno mnogo decimalnih mjesta. Stroj se nikada ne zaustavlja i na izlaznu traku zapisuje vrijednost broja e dobivenu u svakoj iteraciji računskog postupka.

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Sadržaj

1. Uvod

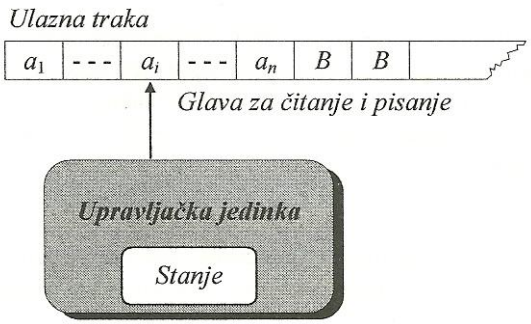
U ovom radu potrebno je konstruirati Turingov stroj koji će aproksimirati prirodan broj e korištenjem Taylorovog reda, na beskonačno iteracija. Ovakav zadatak zahtijeva da stroj niti u jednom trenutku ne stane, jer izraz za aproksimaciju u sebi podrazumijeva iteracije od 0 do beskonačnosti. Stoga će stroj u svakoj iteraciji sume Taylorovog reda stroj jednostavno ispisivati na traku približnu vrijednost u toj iteraciji.

U uvodnom poglavlju, dan je teoretski pregled Turingovog stroja te objašnjeni osnovni principi vezani uz zadatak. Zatim se opisuje algoritam koji računa aproksimaciju, te njegova implementacija Turingovim strojem. Na kraju se daje kratak zaključak.

1.1.Teoretski uvod u Turingov stroj

Turingov stroj (TS) je najopćenitiji poznati matematički model računanja. Bez obzira na njegovu jednostavnost TS ima iste mogućnosti računanja kao bilo koje digitalno računalo. Opisao ga je Alan Turing 1936. godine.

Osnovni model TS-a se sastoji od upravljačke jedinke, ulazne trake i glave za čitanje i pisanje. Upravljačka jedinka može poprimiti konačan broj stanja, ovisno o funkcijama prijelaza. Ulazna traka je omeđena s lijeve, a beskonačna s desne strane i podijeljena na ćelije u koje se zapisuje po jedan od konačnog broja znakova, opet ovisno o funkcijama prijelaza. Glava za čitanje i pisanje se miče od ćelije do ćelije, čita znakove u njima i, ovisno o odgovarajućoj funkciji prijelaza, zapisuje znak u pročitanu ćeliju i nakon toga se pomiče ulijevo ili udesno.



Postoje prošireni TS koje se mogu svesti na osnovni model i dokazati istovjetnost.
Turingov stroj zadaje se uređenom sedmorkom TS(Q,∑,Γ,δ,q0,B,F) gdje je:

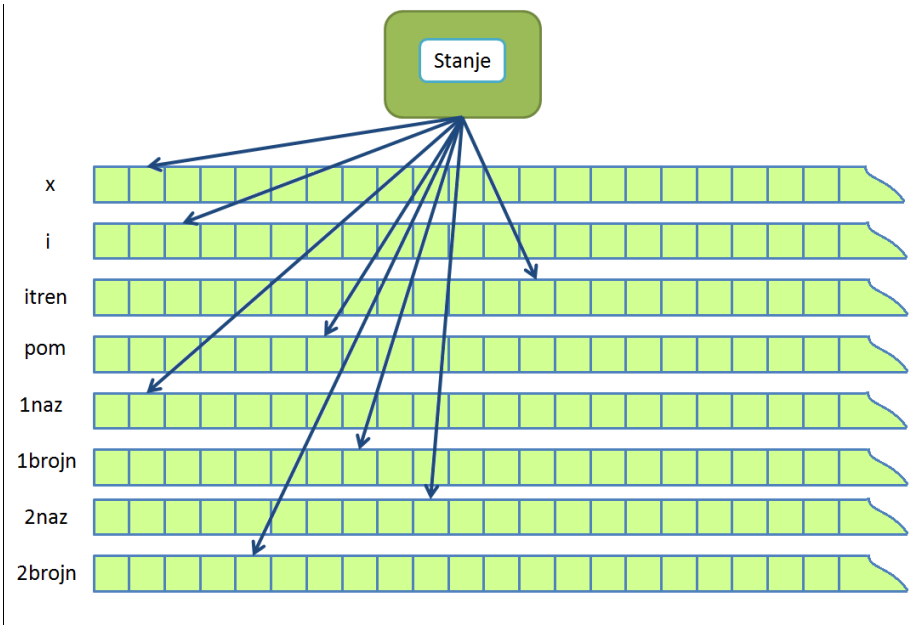
Q - konačan skup stanja;
∑ - konačan skup ulaznih znakova;
Γ - konačan skup znakova trake;
δ - funkcija prijelaza;

1.2. e^x aproksimacija Taylorovim redom

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots = 1 + \sum_{n=1}^{\infty} \frac{x^n}{n!}, \quad \forall x \in \mathbb{R}.$$

U zadatku je potrebno aproksimirati prirodan broj e sumiranjem Taylorovog reda. Ono što izaziva problem u ovom zadatku jest da Turingov stroj može odraditi samo konačan broj iteracija sumiranja zbog brojeva s beskonačnim brojem znamenki. Na primjer, ako bi brojnik u izrazu bio višekratnih broja 2, a ne bio u isto vrijeme i višekratnik broja 3, tada dijeljenje u nekim iteracijama ne bi bilo izvedivo. Na primjer, 2 sa 6 dobiva se broj sa beskonačnim brojem znamenki 0.3333333... To u načelu vrijedi za sve brojeve, jer, idući od 0 do beskonačno, u nazivniku bi se sigurno pojavio broj koji sadrži prosti faktor (različit od 2 ili 5) koji bi izazvao beskonačni zapis. Rješenje ovog problema jest da se prirodni broj e u ovakvoj računici ostavlja kao razlomak. Tada dijeljenje neće uzrokovati beskonačno izračunavanje kvocijenta. Zato se za potrebe rješavanje ovog zadatka uvode razlomci.

TS koji će aproksimirati Taylorov red zamišljen je sa 8 traka i 8 glava za čitanje. Na svaku traku ide po jedna glava. Svaka traka predstavlja jedan zaseban broj. Tako se na traku x sprema vrijednost broja x, na traku i je pohranjena vrijednost iteracije i, traka itren sadrži vrijednost koja se u svakoj poditeraciji mijenja, dok traka pom služi kako pomoćna traka. Trake 1naz, 1brojn, 2naz i 2brojn (prvi nazivnik, prvi brojnik, drugi nazivnik, drugi brojnik) predstavljaju razlomak koji se sumira pri svakoj iteraciji TS-a. Prvi razlomak služi kao prvi broj u koji se računa vrijednost. Dok je u drugom razlomku pohranjena vrijednost prijašnje iteracije koja će se sumirati sa prvim razlomkom u određenom dijelu algoritma.



Broj se zapisuje u notaciji 0^n gdje je n prirodni broj. Ovaj TS je deterministički TS. To znači da za određeno stanje i znak na traci postoji samo jedna funkcija prijelaza. Oblik TS-a sa 8 traka i glava za čitanje je najpovoljniji jer se njime ostvaruje implementacija sa najmanjim mogućim brojem prijelaza. Znakovi trake su samo 0 i B, što uvelike smanjuje broj mogućih slučajeva trake i broj prijelaza.

2. Ostvarenje

2.1.Definicija TS-a:

Ovaj TS je definiran uređenom sedmorkom TS(Q,∑,Γ,δ,Q1,B,F) gdje je:

Q - konačan skup stanja od Q1 do Q56;

∑ - konačan skup ulaznih znakova jest samo znak 0;

Γ - konačan skup znakova trake su znak 0 i B;

B - oznaka praznog znaka jest B

δ - funkcija prijelaza definirana je pojedinim fazama algoritma (149 funkcija prijelaza);

Q1 - početno stanje jest Q1

Inicijalno stanje na trakama je takvo da prve ćelije svake trake imaju upisan znak prazne ćelije. On služi kao graničnik u pomicanju glave na početak niza. Na traci X zapisana je vrijednost broja X, dok na trakama 2naz, 2brojn i pom zapisana je vrijednost 1. Razlog tome jest da što Taylorov red prva iteracija za i=0, u brojnik se piše 1 jer bilo koji broj na nulu jest jedan, dok 0! jest 1. Time je prva iteracija završila i u i traku se upisuje vrijednost 1 (jedan znak „0“).

2.2.Osnovna ideja:

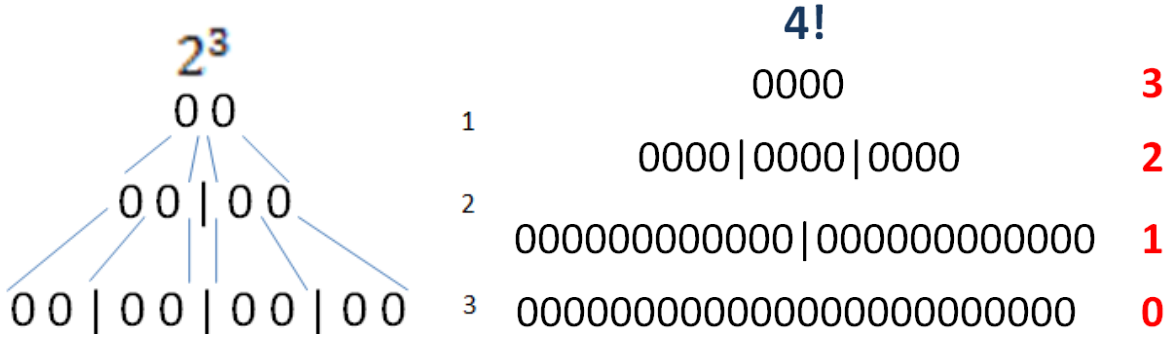
Algoritam aproksimacije Taylorovog reda sveden je u četiri faze:

1. Faza eksponent
2. Faza faktorijel
3. Svođenje na zajednički nazivnik
4. Priprema za novu iteraciju

Za izradu ovog zadatka algoritam je opisan u četiri razine. Prva razina je najapstraktniji prikaz što pojedina faza algoritma radi. Druga razina je opisivanje algoritma na apstraktnoj razini pseudokodom koji trake obilježava kao varijable. Treća razina je pseudokod koji trakama pristupa kao elementima sa konkretnim znakovima. Tu su opisani pomaci glave i kopiranje i brisanje sadržaja traka. Četvrta razina je konkretan opis algoritma funkcijama prijelaza.

1. razina:

U ovih nekoliko slika prikazan je način na koji se eksponira broj x i faktorizira broj i:



Brojevi sa strane prikazuju broj kako se pojedina iteracija algoritma mijenja (u ovom slučaju to je zapis napisan na traci itren)

Sumiranje dvaju broja svodi se ovom formulom:

$$\frac{a}{b} + \frac{c}{d} = \frac{a*d+c*b}{b+d}$$

2. razina:

Algoritam aproksimacije Taylorovog reda sveden je na apkstraktni pseudokod. Svaka traka opisuje jednu varijablu u kojoj se pohranjen neki cijeli broj.

```
SSP()
{
        while(true)
        {
                eksponent();
                faktorijel();
                svodjenjeNaZajednickiNazivnik();
                pripremaZaNovuIteraciju();
        }
}
```

- SSP() je funkcija koja beskonačno obavlja ovaj algoritam. Sastavljena je od više podalgoritama koji svaki vrši svoju funkciju. One su objašnjene u nastavku.

```
eksponent()
{
        1brojn = 0;
        itren = i;
        pom = x;
        while(itren > 0)
        {
                1brojn += pom;
                pom = 1brojn;
                itren--;
        }
}
```

- eksponent() je funkcija koja eksponira x^i pri čemu koristi pomoćne varijable itren i pom. Sa ***trake 1brojn*** uklanjaju se svi znakovi *0*, na ***traku itren*** zapisuje se jednak broj znakova *0* koliko ih ima na ***traci i***, a na **traku pom** se prepisuje onoliko znakova koliko ih ima na **traci x**. U svakoj iteraciji ovog dijela algoritma na ***traku 1brojn*** dodaje se onoliko znakova *0* koliko ih ima na ***traci pom***, a zatim se na ***traku pom*** prepisuje sadržaj sa ***trake 1brojn***. Na kraju svake iteracije uklanja se jedan znak sa ***trake itren***, dok se ne dođe do znaka prazne trake (što odgovara smanjenju varijable na 0). U tom trenutku ovaj dio algoritma završava.

```
faktorijel()
{
        itren = i;
        1naz=0;
        itren--;
        if(itren == 0) 1naz = 1;
        while(itren > 0)
        {
                1naz=0;
                foreach znak in itren
                {
                        1naz += pom;
                }
                pom = 1naz;
                itren--;
        }
}
```

- Funkcija faktorijel() računa faktorijelu broja i.
- Na početku se na ***traku itren*** prepisuju svi znakovi sa ***trake i***, a sa ***trake 1naz*** se brišu svi znakovi *0*. Sa ***trake itren*** se dodatno uklanja jedan znak (koji predstavlja množenje s 1 – pa nije potreban). Ako se na ***traci itren*** ne pronađe niti jedan znak *0*, tada se na ***traku 1naz*** dodaje jedan znak *0*.
- Unutarnja petlja algoritma (prikazana u kodu lijevo) dodaje onoliko znakova *0* na ***traku 1naz*** koliko ih ima na ***traci pom***. To se radi jednom za svaki postojeći znak na ***traci itren*** i predstavlja množenje uzastopnim zbrajanjem. Nakon što je množenje obavljeno, međurezultat se sprema na ***traci pom***, prepisivanjem svih znakova *0* sa ***trake 1naz***. Nakon toga se ukljanja jedan znak *0* sa ***itren trake*** te se algoritam ponavlja dok god na ***itren traci*** postoje znakovi *0*. Time je ostvareno uzastopno množenje s opadajućim brojem i – što odgovara računanju faktorijele.

```
svodjenjeNaZajednickiNazivnik()
{
        itren = 0;
        foreach znak in 1naz
        {
                itren += 2brojn;
        }
        pom = 0;
        foreach znak in 1brojn
        {
                pom += 2naz;
        }
        1brojn = 0; *
        foreach znak in 2naz
        {
                1brojn += 1naz;
        }
        2naz = 1brojn;
        2brojn = itren + pom;
}
```

- svodjenjeNaZajednickiNazivik () je funkcija koja računa $\frac{a}{b}+\frac{c}{d}=\frac{a*d+c*b}{b*d}$ dok u domeni traka (varijabli) funkcija izgleda ovako:

$$\frac{1brojn}{1naz}+\frac{2brojn}{2naz}=\frac{1brojn*2naz+2brojn*1naz}{1naz*2naz}$$

Množenje se i u ovom dijelu algoritma izvodi uzastopnim zbrajanjem, jednako kako je objašnjeno u fazi algoritma za računanje faktorijele. U ovom dijelu algoritma za to se koriste ***trake 1brojn***, ***2naz***, ***2brojn*** te ***1naz***. Nakon što je množenje izvršeno, rezultati brojnika se zbrajaju dodavanjem znakova *0* iz ***traka itren*** i ***pom***, ***traci 2brojn***.

*1brojn predstavlja sada samo pomoćnu varijablu a ne prvi brojnik. U ovom koraku prvi brojnik je pomnožen sa drugim nazivnikom i spremljen u pomoćnu varijablu pom.

```
pripremaZaNovuIteraciju()
{
        pom = 0;
        itren = 0;
        1brojn = 0;
        1naz = 0;
        i++;
}
```

- pripremaZaNovuIteraciju() je funkcija koja priprema sadžaj traka za novu interaciju. Uklanja sve znakove *0* osim graničnika za ***trake pom***, ***itren***, ***1brojn*** te ***1naz***. Nakon toga dodaje znak 0 na kraj ***trake i***, što odgovara promjeni indeksa sume aproksimacije.

3. Razina

U ovom načinu prikaza prikazan je pseudokod podrobnijeg algoritma koji je usko vezan uz opis rada Turingovog stroja. Trake više ne predstavljaju varijable, već mjesto gdje se bilježi pojedini znak na traci. Kraj svake naredbe stoji referenca na funkciju prijelaza na koji se odnosi ovaj kod.

Tablica operatora

Znak/operator	Objašnjenje
=	Operator koji govori da za svaki znak s jedne trake prepisuje na drugu s time da se traka pomiče u smjeru koji određuje strelica pokraj
->,<-,←,→	Znak označava pomak glave
delete	Označava da za određeni znak na traci prepisuje znak prazne ćelije
#	Označava znak u funkciji prijelaza. Ovaj znak ima dvojno značenje. U lijevoj strani prijelaza označava kako nije važan sadržaj te ćelije u koju gleda ta glava. To nam u veliko smanjuje broj prijelaza jer to predstavlja univerzalan način na koji pokriva sve moguće slučajeve. Ovaj znak uveden je radi pojednostavljivanja funkcija prijelaza. U desnoj strani prijelaza ovaj znak označava da glava stoji na mjestu i na traku zapisuje stari znak, tj. sadržaj trake ostaje nepromijenjen.
!	U funkcijama prijelaza označava akciju pri kojoj se glava trake ne pomiče. Ono je istovjetno s osnovnim modelom TS-a koje se može izvesti tako da glava se makne u lijevu stranu i nakon toga ode u jedno među-stanje te se iz njega vrati u prvobitno stanje i glavu pomakne u desno, tj. onaj prvi znak. Na taj način se izvodi isti efekt. Ovaj znak je uveden radi pojednostavljenja algoritma.

U nastavku su dani pseudokôd koji opisuje algoritam koristeći konkretne prijelaze, dok su sami prijelazi dani nakon kôda.

```
eskponent()
{
      itren=i; (->)     (1.1)
      pom=x; (->)       (1.2)
      pom <-;           (1.3)
      itren <-;         (1.3)
      while(itren)
      {
            1brojn=pom; (->)(1.4.1)
            pom=delete; (<-)(1.4.2)
            1brojn <-;      (1.4.3)
            pom=1brojn; (->)(1.4.4)
            pom <-;         (1.4.5)
            itren--;
      }
      pom=delete; (->)(1.5)
}
```

```
faktorijel()
{
      itren=i; (->)     (2.1)
      i<-;              (2.2)
      itren<-;          (2.2)
      itren--;          (2.3)
      pom=i; (->)       (2.4)
      i<-;              (2.5)
      pom<-;            (2.5)
      while(itren)
      {
            1naz=delete;(<-)      (2.6.1)
            foreach znak in pom   (2.6.2)
            {
                  1naz=pom; (->)  (2.6.2)
                  pom<-;          (2.6.2)
            }
            pom->;                (2.6.3)
            pom=delete; (<-)      (2.6.3)
            1naz<-;               (2.6.4)
            pom=1naz; (->)        (2.6.4)
            pom<-;                (2.6.5)
            itren--;              (2.6.5)
            itren<-;              (2.6.5)
      }
}
```

```
svodjenjeNaZajednickiNazivnik()
{
      itren->;                  (3.1)
      pom->;                    (3.1)
      itren=delite (<-)         (3.1)
      pom=delite (<-)           (3.1)
      1brojn<-;                 (3.2)
      1naz<-;                   (3.2)
      2brojn<-;                 (3.2)
      2naz<-;                   (3.2)
      foreach znak in 1naz      (3.3)
      {
            itren=2brojn; (->)      (3.3.1)
            2brojn<-;               (3.3.1)
      }
      foreach znak in 1brojn    (3.4)
      {
            pom=2naz; (->)          (3.4.1)
            2naz<-;                 (3.4.1)
      }
      1brojn=delete (<-)        (3.5)
      1naz<-;                   (3.5)
      2naz<-;                   (3.5)
      foreach znak in 2naz      (3.6)
      {
            1brojn=1naz; (->)       (3.6.1)
            1naz<-;                 (3.6.1)
      }
      2naz=delete; (<-)         (3.7)
      1brojn<-;                 (3.7)
      2naz=1brojn; (->)         (3.7)
      2brojn=delete; (->)             (3.8.1)
      itren<-;                        (3.8.1)
      pom<-;                          (3.8.1)
      2brojn=itren; (->)              (3.8.2)
      2brojn=pom; (->)                (3.8.3)
}
```

```
pripremaZaNovuIteraciju()
{
      1brojn=delete; (<-)       (4.1)
      1naz=delete; (<-)         (4.1)
      pom=delete; (<-)          (4.1)
      itren=delete (<-)         (4.1)
      2brojn<-;                 (4.2)
      2naz<-;                   (4.2)
      i->;                      (4.3)
      i++;                      (4.3)
}
```

Algoritam eksponent

1.1 itren = i;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	Pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
1. δ	Q1	#	0	B	#	#	#	#	#	#	0,R	0,R	#	#	#	#	#	Q1
2. δ	Q1	#	B	B	#	#	#	#	#	#	B,L	B,L	#	#	#	#	#	Q2

1.2 pom = x;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	Pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
3. δ	Q2	0	#	#	B	#	#	#	#	0,R	#	#	0,R	#	#	#	#	Q2
4. δ	Q2	B	#	#	B	#	#	#	#	B,L	#	#	B,L	#	#	#	#	Q3

1.3 pom←; itren←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
5. δ	Q3	#	#	0	0	#	#	#	#	#	#	0,L	0,L	#	#	#	#	Q3
6. δ	Q3	#	#	B	0	#	#	#	#	#	#	#	0,L	#	#	#	#	Q3
7. δ	Q3	#	#	0	B	#	#	#	#	#	#	0,L	#	#	#	#	#	Q3
8. δ	Q3	#	#	B	B	#	#	#	#	#	#	#	#	#	#	#	#	Q4

1.4 Algoritam funkcije eksponent broja X

1.4.1 1brojn = pom;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	Pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
9. δ	Q4	#	#	0	0	B	#	#	#	#	#	0,!	0,R	0,R	#	#	#	Q4
10. δ	Q4	#	#	0	B	B	#	#	#	#	#	0,!	#	#	#	#	#	Q5
11. δ	Q4	#	#	B	#	#	#	#	#	#	#	#	#	#	#	#	#	Q13

1.4.2 pom←delite;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
12. δ	Q5	#	#	0	B	#	#	#	#	#	#	#	B,L	#	#	#	#	Q6
13. δ	Q6	#	#	0	0	#	#	#	#	#	#	#	B,L	#	#	#	#	Q6
14. δ	Q6	#	#	0	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q7

1.4.3 1brojn←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	Itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
15. δ	Q7	#	#	0	#	B	#	#	#	#	#	#	#	B,L	#	#	#	Q8
16. δ	Q8	#	#	0	#	0	#	#	#	#	#	#	#	0,L	#	#	#	Q8
17. δ	Q8	#	#	0	#	B	#	#	#	#	#	#	#	B,R	#	#	#	Q9

1.4.4 pom = 1brojn;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
18. δ	Q9	#	#	0	B	0	#	#	#	#	#	#	0,R	0,R	#	#	#	Q9
19. δ	Q9	#	#	0	B	B	#	#	#	#	#	#	B,L	#	#	#	#	Q10

1.4.5 pom←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
20. δ	Q10	#	#	0	0	#	#	#	#	#	#	#	0,L	#	#	#	#	Q10
21. δ	Q10	#	#	0	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q11

1.4.6 itren--;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	Pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
22. δ	Q11	#	#	0	#	#	#	#	#	#	#	B,L	#	#	#	#	#	Q12

1.5 pom→delite;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
23. δ	Q12	#	#	#	0	#	#	#	#	#	#	#	0,R	#	#	#	#	Q12
24. δ	Q12	#	#	#	B	#	#	#	#	#	#	#	B,L	#	#	#	#	Q13
25. δ	Q13	#	#	#	0	#	#	#	#	#	#	#	B,L	#	#	#	#	Q13
26. δ	Q13	#	#	0	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q3
27. δ	Q13	#	#	B	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q14

Agoritam faktorijel

2.1 itren = i;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
28. δ	Q14	#	0	B	#	#	#	#	#	#	0,R	0,R	#	#	#	#	#	Q14
29. δ	Q14	#	B	B	#	#	#	#	#	#	B,L	B,R	#	#	#	#	#	Q15

2.2 i←; itren ←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
30. δ	Q15	#	0	0	#	#	#	#	#	#	#	0,L	0,L	#	#	#	#	Q15
31. δ	Q15	#	B	B	#	#	#	#	#	#	#	B,R	B,R	#	#	#	#	Q16

2.3 itren--;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
32. δ	Q16	#	#	0	#	#	#	#	#	#	#	B,R	#	#	#	#	#	Q17

2.4 pom = i;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
33. δ	Q17	#	0	#	B	#	#	#	#	#	0,R	#	0,R	#	#	#	#	Q17
34. δ	Q17	#	B	#	B	#	#	#	#	#	B,L	#	B,L	#	#	#	#	Q18

2.5 i←; pom ←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
35. δ	Q18	#	0	#	0	#	#	#	#	#	0,L	#	0,L	#	#	#	#	Q18
36. δ	Q18	#	B	#	B	#	#	#	#	#	B,R	#	B,R	#	#	#	#	Q19

2.6 algoritam faktoralizacije

2.6.1 1naz = delite;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
37. δ	Q19	#	#	0	#	#	B	#	#	#	#	#	#	#	#	#	#	Q21
38. δ	Q20	#	#	0	#	#	0	#	#	#	#	#	#	#	B,L	#	#	Q20
39. δ	Q20	#	#	0	#	#	B	#	#	#	#	#	#	#	B,R	#	#	Q21

2.6.2 foreach znak itren

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
40. δ	Q21	#	#	0	#	#	#	#	#	#	#	#	#	#	#	#	#	Q22
41. δ	Q21	#	#	B	#	#	#	#	#	#	#	#	#	#	#	#	#	Q24

2.6.2 foreach znak pom { 1naz = pom; pom←};

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	Itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
42. δ	Q22	#	#	0	0	#	B	#	#	#	#	#	0,R	#	0,R	#	#	Q22
43. δ	Q22	#	#	0	B	#	B	#	#	#	#	#	B,L	#	B,L	#	#	Q23
44. δ	Q23	#	#	0	0	#	#	#	#	#	#	#	0,L	#	#	#	#	Q23
45. δ	Q23	#	#	0	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q22
46. δ	Q22	#	#	B	#	#	#	#	#	#	#	B,L	#	#	#	#	#	Q24

2.6.3 pom→; pom = delite;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
47. δ	Q24	#	#	0	0	#	#	#	#	#	#	#	0,R	#	#	#	#	Q24
48. δ	Q24	#	#	0	B	#	#	#	#	#	#	#	B,L	#	#	#	#	Q25
49. δ	Q25	#	#	0	0	#	#	#	#	#	#	#	B,L	#	#	#	#	Q25
50. δ	Q25	#	#	0	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q26

2.6.4 1naz←; pom = 1naz;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
51. δ	Q26	#	#	0	#	#	0	#	#	#	#	#	#	#	0,L	#	#	Q26
52. δ	Q26	#	#	0	#	#	B	#	#	#	#	#	#	#	B,R	#	#	Q27
53. δ	Q27	#	#	0	B	#	0	#	#	#	#	#	0,R	#	0,R	#	#	Q27
54. δ	Q72	#	#	0	B	#	B	#	#	#	#	#	B,L	#	B,L	#	#	Q28

2.6.5 pom←; itren--; itren←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
55. δ	Q28	#	#	0	0	#	#	#	#	#	#	#	0,L	#	#	#	#	Q28
56. δ	Q28	#	#	0	B	#	#	#	#	#	#	#	B,R	#	#	#	#	Q29
57. δ	Q29	#	#	0	#	#	#	#	#	#	#	B,L	#	#	#	#	#	Q30
58. δ	xQ30	#	#	B	#	#	#	#	#	#	#	#	#	#	#	#	#	Q32
59. δ	xQ30	#	#	0	#	#	#	#	#	#	#	#	#	#	#	#	#	Q31
60. δ	Q31	#	#	0	#	#	#	#	#	#	#	0,L	#	#	#	#	#	Q31
61. δ	xQ31	#	#	B	#	#	#	#	#	#	#	B,R	#	#	#	#	#	Q21

Algoritam za svođenje na zajednički nazivnik

3.1 itren & pom →; itren=delite; pom=delite;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
62. δ	Q32	#	#	0	0	#	#	#	#	#	#	0,R	0,R	#	#	#	#	Q32
63. δ	Q32	#	#	B	0	#	#	#	#	#	#	#	0,R	#	#	#	#	Q32
64. δ	Q32	#	#	0	B	#	#	#	#	#	#	0,R	#	#	#	#	#	Q32
65. δ	Q32	#	#	B	B	#	#	#	#	#	#	B,L	B,L	#	#	#	#	Q33
66. δ	Q33	#	#	0	0	#	#	#	#	#	#	B,L	B,L	#	#	#	#	Q33
67. δ	Q33	#	#	B	0	#	#	#	#	#	#	#	B,L	#	#	#	#	Q33
68. δ	Q33	#	#	0	B	#	#	#	#	#	#	B,L	#	#	#	#	#	Q33
69. δ	Q33	#	#	B	B	#	#	#	#	#	#	B,R	B,R	#	#	#	#	Q34

3.2 1brojn ←; 1naz←; 2brojn←; 2naz←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
70. δ	Q34	#	#	#	#	0	0	0	0	#	#	#	#	0,L	0,L	0,L	0,L	Q34
71. δ	Q34	#	#	#	#	0	0	0	B	#	#	#	#	0,L	0,L	0,L	#	Q34
72. δ	Q34	#	#	#	#	0	0	B	0	#	#	#	#	0,L	0,L	#	0,L	Q34
73. δ	Q34	#	#	#	#	0	0	B	B	#	#	#	#	0,L	0,L	#	#	Q34
74. δ	Q34	#	#	#	#	0	B	0	0	#	#	#	#	0,L	#	0,L	0,L	Q34
75. δ	Q34	#	#	#	#	0	B	0	B	#	#	#	#	0,L	#	0,L	#	Q34
76. δ	Q34	#	#	#	#	0	B	B	0	#	#	#	#	0,L	#	#	0,L	Q34
77. δ	Q34	#	#	#	#	0	B	B	B	#	#	#	#	0,L	#	#	#	Q34
78. δ	Q34	#	#	#	#	B	0	0	0	#	#	#	#	#	0,L	0,L	0,L	Q34
79. δ	Q34	#	#	#	#	B	0	0	B	#	#	#	#	#	0,L	0,L	#	Q34
80. δ	Q34	#	#	#	#	B	0	B	0	#	#	#	#	#	0,L	#	0,L	Q34
81. δ	Q34	#	#	#	#	B	0	B	B	#	#	#	#	#	0,L	#	#	Q34
82. δ	Q34	#	#	#	#	B	B	0	0	#	#	#	#	#	#	0,L	0,L	Q34
83. δ	Q34	#	#	#	#	B	B	0	B	#	#	#	#	#	#	0,L	#	Q34
84. δ	Q34	#	#	#	#	B	B	B	0	#	#	#	#	#	#	#	0,L	Q34
85. δ	Q34	#	#	#	#	B	B	B	B	#	#	#	#	B,R	B,R	B,R	B,R	Q35

3.3 množenje *2brojn* i *1naz* a rješenje u *itren*(itren = 2brojn * 1naz)

3.3.1 foreach znak 1naz { itren = 2brojn; 2brojn←;}

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
86. δ	Q34	#	#	B	#	#	0	0	#	#	#	0,R	#	#	#	0,R	#	Q34
87. δ	Q34	#	#	B	#	#	0	B	#	#	#	#	#	#	#	B,L	#	Q35
88. δ	Q35	#	#	#	#	#	0	0	#	#	#	#	#	#	#	0,L	#	Q35
89. δ	Q35	#	#	#	#	#	0	B	#	#	#	#	#	#	#	B,R	#	Q36
90. δ	Q36	#	#	#	#	#	0	#	#	#	#	0,R	#	#	#	#	#	Q34
91. δ	Q34	#	#	#	#	#	B	#	#	#	#	#	#	#	#	#	#	Q37

3.4 pom = 2brojn * 1naz

3.4.1 foreach znak 1brojn { pom = 2naz; 2naz←;}

δ	Stanje	Znakovi na pojedinim trakama	Akcije na pojedinim trakama	Sljedeće

		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	stanje
92. δ	Q37	#	#	#	B	0	#	#	0	#	#	#	0,R	#	#	#	0,R	Q37
93. δ	Q37	#	#	#	B	0	#	#	B	#	#	#	B,L	#	#	#	B,L	Q38
94. δ	Q38	#	#	#	#	0	#	#	0	#	#	#	#	#	#	#	0,L	Q38
95. δ	Q38	#	#	#	#	0	#	#	B	#	#	#	#	#	#	#	B,R	Q39
96. δ	Q39	#	#	#	#	0	#	#	#	#	#	#	#	0,R	#	#	#	Q40
97. δ	Q40	#	#	#	#	0	#	#	#	#	#	#	#	#	#	#	#	Q37
98. δ	Q40	#	#	#	#	B	#	#	#	#	#	#	#	B,L	#	#	#	Q41

3.5 1brojn = delite; 1naz←;2naz←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
99. δ	Q41	#	#	#	#	0	#	#	#	#	#	#	#	B,L	#	#	#	Q41
100. δ	Q41	#	#	#	#	B	#	#	#	#	#	#	#	B,R	#	#	#	Q42
101. δ	Q42	#	#	#	#	#	0	#	0	#	#	#	#	#	0,L	#	0,L	Q42
102. δ	Q42	#	#	#	#	#	B	#	0	#	#	#	#	#	#	#	0,L	Q42
103. δ	Q42	#	#	#	#	#	0	#	B	#	#	#	#	#	0,L	#	#	Q42
104. δ	Q42	#	#	#	#	#	B	#	B	#	#	#	#	#	B,R	#	B,R	Q43

3.6 1brojn = 2naz * 1naz;

3.6.1 foreach znak 2nazivnik { 1brojn = 1naz; 1naz←; }

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
105. δ	Q43	#	#	#	#	B	0	#	0	#	#	#	#	0,R	0,R	#	#	Q43
106. δ	Q43	#	#	#	#	B	B	#	0	#	#	#	#	B,L	B,L	#	#	Q44
107. δ	Q44	#	#	#	#	#	0	#	0	#	#	#	#	#	0,L	#	#	Q44
108. δ	Q44	#	#	#	#	#	B	#	0	#	#	#	#	#	B,R	#	#	Q45
109. δ	Q45	#	#	#	#	#	#	#	0	#	#	#	#	#	#	#	0,L	Q46
110. δ	Q46	#	#	#	#	#	#	#	0	#	#	#	#	#	#	#	#	Q43
111. δ	Q46	#	#	#	#	#	#	#	B	#	#	#	#	#	#	#	B,L	Q47

3.7 2naz = delite; 1brojn←; 2naz = 1brojn;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
112. δ	Q47	#	#	#	#	#	#	#	0	#	#	#	#	#	#	#	B,L	Q47
113. δ	Q47	#	#	#	#	#	#	#	B	#	#	#	#	#	#	#	B,R	Q48
114. δ	Q48	#	#	#	#	0	#	#	#	#	#	#	#	0,L	#	#	#	Q48
115. δ	Q48	#	#	#	#	B	#	#	#	#	#	#	#	B,R	#	#	#	Q49
116. δ	Q49	#	#	#	#	0	#	#	B	#	#	#	#	0,R	#	#	0,R	Q49
117. δ	Q49	#	#	#	#	B	#	#	B	#	#	#	#	B,L	#	#	B,L	Q50

3.8 2brojn = itren + pom;

3.8.1 2brojn = delite; itren←; pom←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	

118. δ	Q50	#	#	#	#	#	#	0	#	#	#	#	#	#	#	B,L	#	Q50
119. δ	Q50	#	#	#	#	#	#	B	#	#	#	#	#	#	#	B,R	#	Q51
120. δ	Q51	#	#	0	0	#	#	#	#	#	#	0,L	0,L	#	#	#	#	Q51
121. δ	Q51	#	#	0	B	#	#	#	#	#	#	0,L	#	#	#	#	#	Q51
122. δ	Q51	#	#	B	0	#	#	#	#	#	#	#	0,L	#	#	#	#	Q51
123. δ	Q51	#	#	B	B	#	#	#	#	#	#	B,R	B,R	#	#	#	#	Q52

3.8.2 2brojn = itren;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	Itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
124. δ	Q52	#	#	0	#	#	#	B	#	#	#	0,R	#	#	#	0,R	#	Q52
125. δ	Q52	#	#	B	#	#	#	B	#	#	#	#	#	#	#	B,L	#	Q53

3.8.3 2brojn = pom;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
126. δ	Q53	#	#	#	0	#	#	B	#	#	#	#	0,R	#	#	0,R	#	Q53
127. δ	Q53	#	#	#	B	#	#	B	#	#	#	#	B,L	#	#	B,L	#	Q54

Pripremanje za novu iteraciju

4.1 1brojn = delite; 1naz = delite; pom = delite; itren = delite:

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
128. δ	Q54	#	#	0	0	0	0	#	#	#	#	B,L	B,L	B,L	B,L	#	#	Q54
129. δ	Q54	#	#	0	0	0	B	#	#	#	#	B,L	B,L	B,L	#	#	#	Q54
130. δ	Q54	#	#	0	0	B	0	#	#	#	#	B,L	B,L	#	B,L	#	#	Q54
131. δ	Q54	#	#	0	0	B	B	#	#	#	#	B,L	B,L	#	#	#	#	Q54
132. δ	Q54	#	#	0	B	0	0	#	#	#	#	B,L	#	B,L	B,L	#	#	Q54
133. δ	Q54	#	#	0	B	0	B	#	#	#	#	B,L	#	B,L	#	#	#	Q54
132. δ	Q54	#	#	0	B	B	0	#	#	#	#	B,L	#	#	B,L	#	#	Q54
135. δ	Q54	#	#	0	B	B	B	#	#	#	#	B,L	#	#	#	#	#	Q54
136. δ	Q54	#	#	B	0	0	0	#	#	#	#	#	B,L	B,L	B,L	#	#	Q54
137. δ	Q54	#	#	B	0	0	B	#	#	#	#	#	B,L	B,L	#	#	#	Q54
138. δ	Q54	#	#	B	0	B	0	#	#	#	#	#	B,L	#	B,L	#	#	Q54
139. δ	Q54	#	#	B	0	B	B	#	#	#	#	#	B,L	#	#	#	#	Q54
140. δ	Q54	#	#	B	B	0	0	#	#	#	#	#	#	B,L	B,L	#	#	Q54
141. δ	Q54	#	#	B	B	0	B	#	#	#	#	#	#	B,L	#	#	#	Q54
142. δ	Q54	#	#	B	B	B	0	#	#	#	#	#	#	#	B,L	#	#	Q54
143. δ	Q54	#	#	B	B	B	B	#	#	#	#	B,R	B,R	B,R	B,R	#	#	Q55

4.2 2brojn←; 2naz←;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
144. δ	Q55	#	#	#	#	#	#	0	0	#	#	#	#	#	#	0,L	0,L	Q55
145. δ	Q55	#	#	#	#	#	#	0	B	#	#	#	#	#	#	0,L	#	Q55
146. δ	Q55	#	#	#	#	#	#	B	0	#	#	#	#	#	#	#	0,L	Q55
147. δ	Q55	#	#	#	#	#	#	B	B	#	#	#	#	#	#	B,R	B,R	Q56

4.3 i→; i++;

δ	Stanje	Znakovi na pojedinim trakama								Akcije na pojedinim trakama								Sljedeće stanje
		X	i	itren	pom	1brojn	1naz	2brojn	2naz	X	i	itren	pom	1brojn	1naz	2brojn	2naz	
148. δ	Q56	#	0	#	#	#	#	#	#	#	0,R	#	#	#	#	#	#	Q56
149. δ	Q56	#	B	#	#	#	#	#	#	#	0,!	#	#	#	#	#	#	Q1

3. Zaključak

U ovom seminarskom radu opisan je jedan od načina rješavanja matematičke aproksimacije e^x pomoću Taylorovog reda korištenjem Turingovog stroja.

Ovaj zadatak može se riješiti i na druge načine. Umjesto 0^n notacije prirodnog broja n, broj se može zapisati i u dekadskoj bazi. To bi značilo da bi broj slučajeva za 8 traka bilo: 10 znakova, jedan za svaku znamenku te znak prazne ćelije, što čini ukupno 10 mogućih znakova. Svaki znak je moguće kombinirati sa svih drugih 11 znakova. To ispada 11 * 11 = 121 kombinacija. S obzirom da se koristi 8 traka, to bi broj slučajeva koje bi trebali pokriti u funkcijama prijelaza jest broj 121^8 što daje jako veliki broj mogućih uvjeta. Ovakav način rada bio bi puno brži i imao bi manju prostornu složenost. Ali zato bi imao mnogo više funkcija prijelaza.

Drugi pristup ovome problemu bio bi ostvariti TS sa manje traka, ili čak samo jednom. U tom bi slučaju bilo potrebno uvesti ili nove znakove trake i graničnike koje bi davale informaciju gdje koji broj završava ili ostvariti nova složena stanja. Ovakav pristup uvelike bi uvećao broj stanja u kojima se automat može naći, jer za svaku radnju mora postojati zasebno stanje. Radi smanjenja velikog broj funkcija također se može problemu pristupiti korištenjem nedeterminističkog Turingovog stroja.

Još jedna od mogućnosti koja bi se u proširenju algoritma mogla uraditi jest pretvorba broja predstavljenog određenim brojem znakova *0* u dekadski broj radi čitkosti. Tako bi pri svakoj iteraciji algoritma i čovjek mogao pročitati vrijednost aproksimacije koja se računa.

Optimizacija se, kako je već spomenuto, može napraviti iz različitih aspekata algoritma. Rješenje predstavljeno u ovom radu ne predstavlja optimalno rješenje iz perspektive brzine izvođenja, ali je algoritam jednostavan za shvatiti, a TS ima relativno mali broj stanja te funkcija prijelaza, što ga čini jednostavnim za implementaciju. Različiti alternativni modeli TS-a i algoritama za rješavanje ovog nadilaze opseg ovog rada, pa su samo ukratko opisani.

Tijekom konstruiranja Turingovog stroja pokazalo se kako je TS interesantan matematički model za izračunljive operacije. Dakako da u vrijeme suvremenih računala, nema smisla ostavljati takvu vrstu izračuna za TS, s obzirom na to da se mnogo jednostavnije računaju suvremenim računalima, ali se pokazuje kako je Turingov stroj vrlo moćan alat za ovakva i slična matematička teoretska razmatranja.