Univerzitet u Sarajevu
Prirodno-matematički fakultet u Sarajevu

Seminarski rad

Tema: L i NL klase jezika

Student:
Penjić Safet

Sadržaj

I Uvod 3
1.1 Klasa P 3
Zadaci za vježbu 4
1.2 Klasa NP 4
Zadaci za vježbu 4
1.3 Klasa PSPACE 5
Zadaci za vježbu 5

II Klase L i NL 6
Zadaci za vježbu ..... .13

Literatura ..... .14

I Uvod

Čak i kad imamo problem koji možemo odlučiti (riješiti) i tako ga u stvari izračunati, može se desiti (što nije rijedak slučaj) da dato riješenje nije primjenjivo u praksi, zato što to izračunavanje zahtjeva nekontrolisanu količinu vremena ili nekontrolisanu količinu memorije.

U ovom seminarskom radu razmatraćemo kompleksnost izračunavanja problema u zahtjevima količine prostora (memorije), koji oni zahtijevaju. Preciznije, baziraćemo se na problemima koji na nekoj determinističkoj Turing mašini zahtijevaju prostor koji je jednak proizvodu neke konstante i logaritma dužine ulaza (kasnije ćemo precizno definisati ovu klasu koja se zove klasa L). Vrijeme i prostor su dvije najvažnije stvari koje razmatramo kada tražimo rješenje mnogih izračunljivih problema. Prostorna kompleksnost dijeli mnoge osobine sa vremenskom kompleksnošću, i daje način za dalju klasifikaciju problema u skladu sa njihovim izračunljivim teškoćama. Na početku ćemo uvesti samo neke osnovne stvari o klasama P, NP i PSPACE, koje će nam kasnije pomoći za potpuno shvatanje klasa L i NL. Za potpuno shvatanje svih termina u ovom radu, pretpostavićemo da je čitalac upoznat sa najvažnijim osobinama i radom Turing mašine.

1.1 Klasa P

Broj koraka koje algoritam koristi na nekom konkretnom ulazu može ovisiti od nekoliko parametara. Na primjer, ako je ulaz graf, broj koraka može ovisiti od broja vrhova, od broja ivica, i maksimalnog stepena grafa, ili neke kombinacije od ovih i sličnih faktora. Radi jednostavnosti možemo računati potrebno vrijeme algoritma čisto kao funkciju koja zavisi od dužine stringa koji predstavlja ulaz i ne razmatrajući nikakve druge parametre. U najgorem-slučaju analizi, oblike koje ćemo razmatramo ovdje, odnosi se na najduže potrebno vrijeme na svim ulazima određene dužine. U prosječnom-slučaju analizi, razmatramo prosjek svih potrebnih vremena na ulazu određene dužine.

Definicija 1.1: Neka je M deterministička Turing mašina koja staje na svim ulazima. **Potrebno vrijeme** ili **vremenska kompleksnost** od *M* je funkcija *f*: **N** ---> **N** gdje je *f(n)* maksimalni broj koraka koje M koristi na nekom ulazu dužine n. Ako je *f(n)* potrebno vrijeme od M, kažemo da *M* koristi *f(n)* vremena i da je *M f(n)* vremenska Turing mašina. Uobičajeno, koristimo *n* da predstavimo dužinu ulaza. ♠

Definicija 1.2: Neka su *f* i *g* funkcije *f, g*: $\mathbf{N} \rightarrow \mathbf{R}^+$. Kažemo da je ***f(n) = O(g(n))*** ako postoji cijeli brojevi *n* i n_0 takvi da za svaki cijeli $n \geq n_0$ važi

$$f(n) \leq c\, g(n).$$

Kad je *f(n)=O(g(n))* kažemo da je *g(n)* **gornja granica** za *f(n)*, ili preciznije, da je *g(n)* **asimptotski gornja granica** za *f(n)*, da istaknemo kako smijemo zanemariti važnost konstante. ♠

Definicija 1.3: Neka je *t*: $N \rightarrow R^+$ funkcija. Definišemo **vremenski kompleksnu klasu**, **TIME(*t(n)*)** kao porodicu (kolekciju) svih jezika koje su odlučive sa *O(t(n))* vremenskom Turing mašinom. ♠

Definicija 1.4: **P** je klasa jezika koji su odčučivi u polinomijalnom vremenu na determinističkoj jednoj-traci Turing mašini. Drugim riječima,

$$P = \cup_k \text{TIME}(n^k).$$ ♠

Zadaci za vježbu

E1. Dat je jezik PATH = {<*G, s, t*> | G je orjentisan graf koji ima direktan put od s do t}. Dokazati da PATH pripada P klasi jezika.

E2. Dat je jezik RELPRIME = {<x, y> | x i y su relativno prosti}. Dokazati da RELPRIME pripada P klasi jezika.

1.2 Klasa NP

U nekim problemima možemo primjeniti "silu" i dobiti rješenje koje je u polinomijalnom vremenu. Međutim, u velikom broju problema (uključujući mnogo interesantnih i korisnih), čak i kad izbjegnemo "silu" nismo u mogučnosti naći rješenje problema koje radi u polinomijalnom vremenu.

Obično u ovakvim slučajevima, prvo tražimo rješenje koje ima polinomijalno vremenski provjeravač, tj ako znamo rješenje problema, tražimo algoritam koji može u polinomijalnom vremenu provjeriti ispravnost riješenja.

Definicija 1.5: **Provjeravač** za jezik *A* je algoritam *V*, gdje

A={ *w* | *V* prihvata <*w, c*> za neki string *c*}.♠

Mjerimo vrijeme provjerivača jedino u terminu dužine od *w*, tako da je polinomijalno vremenski provjeravač koristi polinomijalno vrijeme u odnosu na dužinu od *w*. Jezik *A* je polinomijalno provjerljiv ako ima polinomijalno vremenski provjeravač.

Provjeravač koristi dodatne informacije, u Definiciji 1.5 predstavljenih sa simbolom *c*, da provjeri da je string *w* član od *A*. Ovu informaciju zovemo certifikat ili dokaz za članstvo u A. Primjetimo da, za polinomijalni provjeravač, certifikat ima polinomijalnu dužinu (u odnosu na dužinu w) iz prostog razloga što je to u stvari sve čemu provjeravač može pristupiti u svojim vremenskim granicama.

Definicija 1.6 **NP** je klasa jezika koje imaju polinomijalno vremenski provjeravač. ♠

Ako uvedemo klasu NTIME tada klasu NP možemo tumačiti i drugačije.

Teorema 1.7 Jezik je u NP ako i samo ako je odlučiv pomoću neke nedeterminističke polinomijalno vremenske Turing mašine. ♠

Definicija 1.8 **NTIME(*t(n)*)** = {L | L je jezik odlučiv pomoću O(t(n)) vremenske nedeterminističke Turing mašine}.♠

Posljedica 1.9 $NP = \cup_k NTIME(n^k)$. ♠

Zadaci za vježbu

F1. Dat je jezik CLIQUE = {<*G, k*> | G je neorjentisan graf sa k-klikom}. Dokazati da CLIQUE pripada NP klasi jezika.

F2. Dat je jezik SUBSET-SUM = {<*S, t*> | $S=\{x_1, ..., x_k\}$ i za neki $\{y_1, ..., y_k\} \subseteq \{x_1, ..., x_k\}$ imamo $\Sigma y_i = t$ }. Dokazati da SUBSET-SUM pripada NP klasi jezika.

1.3 Klasa PSPACE

Kao što smo uradili sa vremenskom kompleksnošću, trebamo izabrati model za izračunavanje količine prostora koji koristi neki algoritam. Nastavićemo sa modelom Turing mašine iz istog razloga koji smo koristili da mjerimo vrijeme. Turing mašine su matematički jednostavne i dovoljno bliske realnom kompjuteru da dadne značajne rezultate.

Definicija 1.10: Neka je *M* deterministička Turing mašina koja staje na svim ulazima. Prostorna kompleksnost od *M* je funkcija *f*: $\mathbf{N} \rightarrow \mathbf{R}^+$, gdje je *f(n)* maksimalni broj ćelija trake koje *M* skenira na nekom ulazu dužine n. Ako je *f(n)* prostorna kompleksnost od *M*, također kažemo da *M* koristi *f(n)* prostora.

Ako je *M* nedeterministička Turing mašina gdje sve branše staju na svim ulazima, definišemo prostornu kompleksnost *f(n)* kao maksimalni broj ćelija trake koje *M* skenira na nekoj branši svog izračunavanja za neki ulaz dužine *n*. ♠

I mi ćemo tipično uvesti prostornu kompleksnost Turing mašine koristeći asimptotičku notaciju.

Definicija 1.11: Neka je *f*: $\mathbf{N} \rightarrow \mathbf{R}^+$ funkcija. Prostorne kompleksne klase SPACE(f(n)) i NSPACE(f(n)), definisane su kao:

SPACE(f(n)) = {L | L je jezik odlučiv sa O(f(n)) prostornom determinističkom Turing mašinom}.

NSPACE(f(n)) = {L | L je jezik odlučiv sa O(f(n)) prostornom nedeterminističkom Turing mašinom}.♠

Sad bez problema možemo definisati klasu PSPACE za prostornu kompleksnost.

Definicija 1.12: **PSPACE** je klasa jezika koji su odlučivi sa polinomijalno prostornom determinističkom Turing mašinom. Drugim riječima,

$$\text{PSPACE} = \cup_k \text{SPACE}(n^k).$$ ♠

Definišemo NPSPACE kao nedeterministička verzija PSPACE, (odlučiv sa polinomijalno prostornom nedeterminističkom TM). Napomenućemo da je PSPACE = NPSPACE.

Zadaci za vježbu

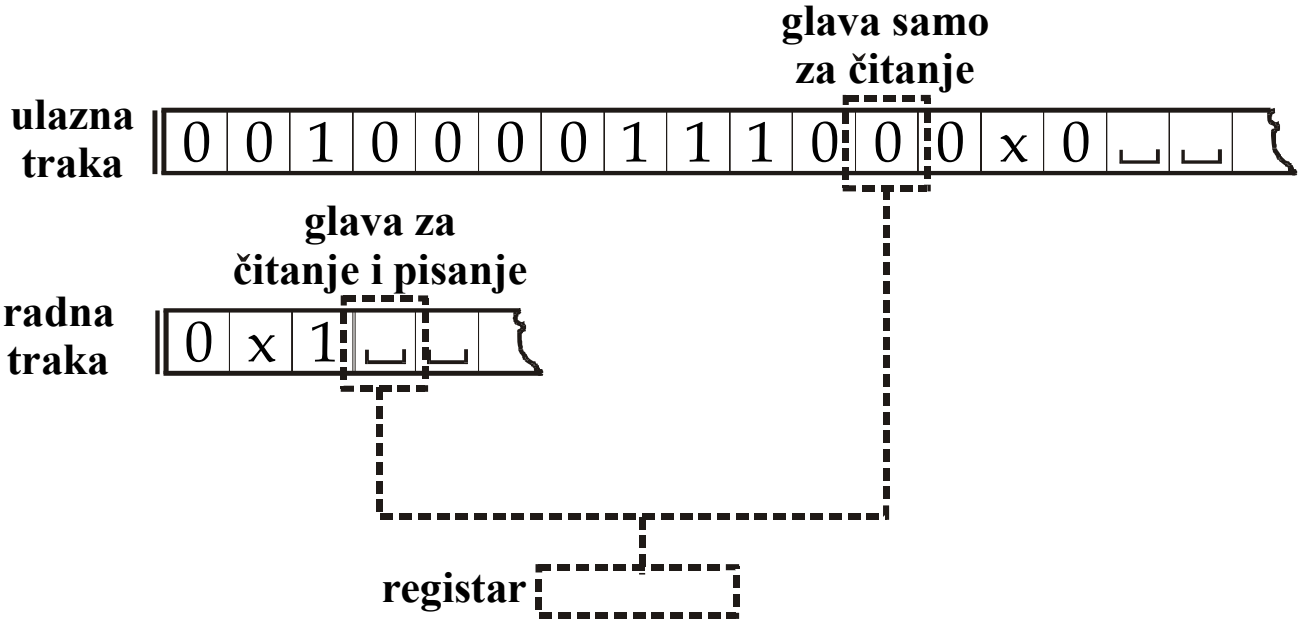
G1. Dat je jezik SAT = {<*f*> | f je zadovoljavajuća Boolean formula}. Dokazati da SAT pripada PSPACE klasi jezika.

G2. Dat je jezik ALL_{NFA} = {<*A*> | A je NFA i L(A) = Σ^* }. Dokazati da ALL_{NFA} pripada NPSPACE klasi jezika.

II Klase L i NL

Klasama P, NP i PSPACE (koja je jednaka klasi NPSPACE) smo uveli linearna ograničenja u posmatranju vremenske i prostorne kompleksnosti tj. ograničenja za funkciju *f(n)* definisana u nekoj od spomenutih klasa, su bila najmanje n. Ovdje ćemo razmatrati manja, podlinearna prostorna ograničenja. U vremenskoj kompleksnosti, podlinearna ograničenja su neodgovarajuća za čitanje čitavog ulaza, tako da nema smisla isti model razmatrati u ovom slučaju. U podlinearnoj prostornoj kompleksnosti mašina može pročitati čitav ulaz ali nema dovoljno prostora da sačuva taj ulaz. Da razmotrimo ovu situaciju u punom značenju, moramo modifikovati naš model izračunavanja.

Predstavićemo Turing mašinu sa dvije trake: samo-čitaj ulaznom trakom i čitaj/piši radnom trakom. Na samo-čitaj traci ulazna glava može detektovati simbol ali ga ne može mijenjati. Omogućimo način da mašina može detektovati kad dođe do lijevog-dijela ili desnog-dijela kraja ulaza. Ulazna glava uvijek mora ostati na dijelu trake koja sadrži ulaz. Radna traka može čitati i pisati na uobičajen način. Jedino ćelije skenirane na radnoj traci se uračunavaju u prostornoj kompleksnosti ovog tipa Turing mašina.



Slika 2.1: Prostorno ograničeno izračunavanje. Jedino ćelije korištene na radnoj traci se broje u prostornoj kompleksnosti.

Možemo razmišljati o samo-čitaj ulaznoj traci kao o DVD-rom uređaju, uređaj koji koristimo za ulaz na mnogim ličnim računarima. Ako imamo stariji računar, DVD-rom sadrži više podataka nego što taj računar može sačuvati u glevnoj memoriji. Podlinearni prostorni algoritam dozvoljava računaru da manipuliše sa podacima bez pisanja svega toga u glavnu memoriju.

Za prostorna ograničenja koja su najmanje linearna, model TM sa dvije trake je ekvivalentan sa standardnim modelom TM sa jednom trakom. Za podlinearna prostorna ograničenja, koristimo samo model sa dvije trake.

Definicija 2.2 **L** je klasa jezika koji su odlučivi u logaritamskom prostoru na determinističkoj Turing mašini. Drugim riječima,

$$L = SPACE(\log n).$$

NL je klasa jezika koji su odlučivi u logaritamskom prostoru na nedeterminističkoj Turing mašini. Drugim riječima,

$$NL = NSPACE(\log n).\spadesuit$$

Fokusiramo se na log$n$ prostor umjesto, recimo $\sqrt{n}$ ili $\log^2 n$ prostora, iz razloga koji su u potpunosti slični onima zbog kojih uzimamo polinomijalno vrijeme i prostorno ograničenja. Logaritamski prostor je dovoljno velik da riješi dovoljno velik broj interensantnih izračunljivih problema, i ima zgodnu matematičku osobinu kao što je prilagodljivost, čak i kad promjenimo model mašine ili metod ulaznog enkodiranja. Pokazivači na ulazu mogu biti predstavljeni u logaritamskom prostoru, pa jedan od načina da predstavimo moć log prostornog algoritama je da razmatramo korisnost koja može biti od fiksiranog broja ulaznih pokazivača.

Zadatak 2.3

Dat je jezik A = $\{<0^k 1^k> \mid k \geq 0\}$. Dokazati da A pripada L klasi jezika.

Rješenje

Prije nego što uradimo zadatak u log prostoru, uradimo lakšu varijantu zadatka. Riješimo zadatak u linearnom prostoru. Sljedeća mašina, M_1, odlučuje jezik A u linearnom prostoru (i polinomijalnom vremenu, preciznije A $\in$ TIME(n log$n$)).

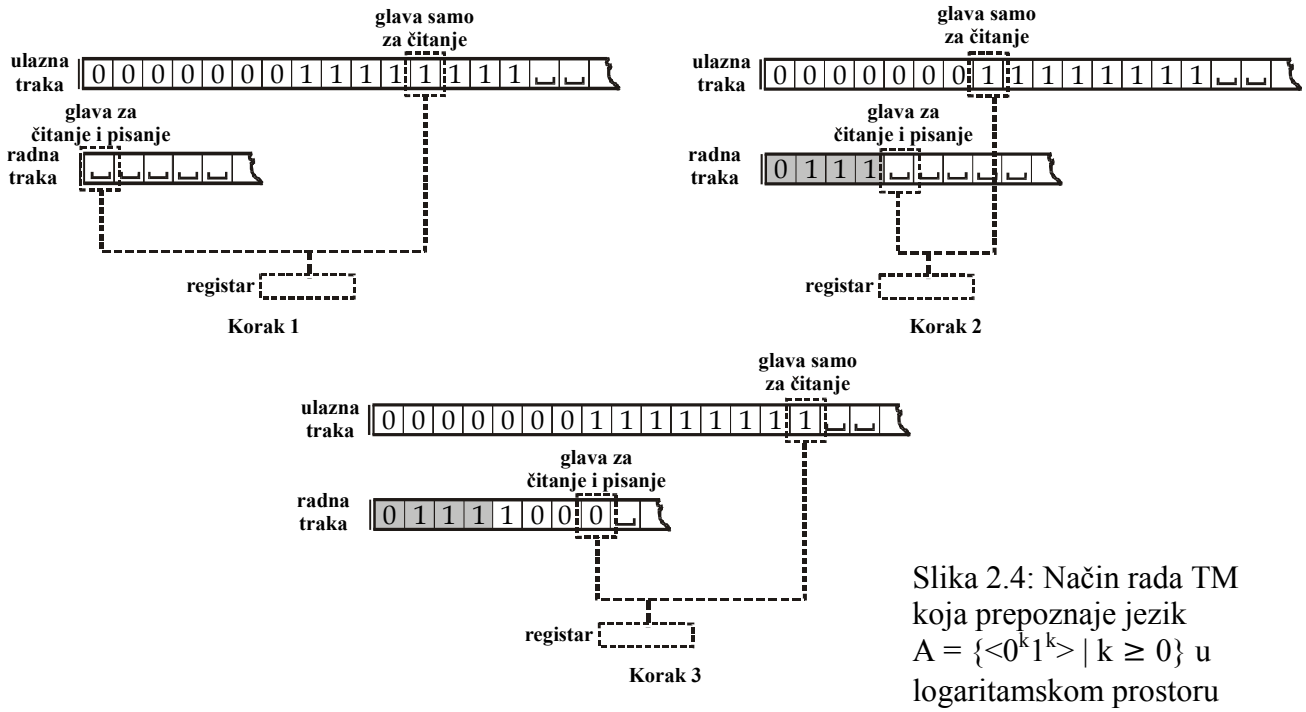
M_1= "Na ulazu w, gdje je w neki string:

1. Skeniraj traku i *odbaci* ako nađeš 0 desno od 1.
2. Ponavljaj sve dok neka 0-la i neka 1-ca ostanu na traci
   3. Skeniraj traku, i provjeri da li je ukupan broj 0-la i 1-ca koje su ostale na traci paran ili neparan. Ako je neparan, *odbaci*.
   4. Ponovo skeniraj traku, križaj svaku drugu nulu počevši od prve, pa onda križaj svaku drugu 1-cu počevši od prve 1-ce.
5. Ako nema 0-la i nema 1-ca na traci, *prihvati*. U suprotnom, *odbaci*."

Logaritamsko prostorna TM za A ne može križati 0-le i 1-ce koje nađe na ulaznoj traci zato što je ta traka samo za čitanje. Umjesto toga mašina broji broj 0-la i, odvojeno, broj 1-ca u binarnom obliku na radnoj traci. Jedini prostor koji zahtjeva je da zabilježi stanje dva brojača.

Algoritam koji dolučuje jezik A je sljedeći:

1. Provjeri da li se 1-ca pojavljuje negdje iza 0-le. Ako je odgovor pozitivan *odbaci*. (Ovo ne zahtjeva radnog prostora)
2. Prebroj broj 0-la i broj 1-ca.
3. Ako su brojači isti *prihvati*, u suprotnom *odbaci*.



Slika 2.4: Način rada TM koja prepoznaje jezik A = $\{<0^k 1^k> \mid k \geq 0\}$ u logaritamskom prostoru

U binarnom obliku, svaki brojač koristi samo logaritamski prostor, i zbog toga, algoritam koristi O(log *n*) prostora. Tako da je A ∈ L, što je i trebalo dokazati.

♣♣♣

Zadatak 2.5

U zadatku E1, na stranici 4, trebalo je pokazati da je jezik *PATH* = {<*G, s, t*> | G je orjentisan graf koji ima direktan put od s do t} član P klase jezika. Sad treba dokazati da *PATH* pripada NL klasi jezika.

Rješenje

Ne znamo može li *PATH* biti riješen u logaritaskom prostoru deterministički, ali znamo nedeterministički log prostorni algoritam za *PATH*. Sljedeća nedeterministička log prostorna Turing mašina odlučuje *PATH*.

M_2 = "Na ulazu <*G, s, t*>, gdje je G orjentisan graf sa vrhovima *s* i *t*:
1. Definiši brojač i inicijaliziraj ga na broj vrhova u grafu.
2. Definiši pokazivač da čuva "trenutni vrh" i inicijaliziraj ga na početni vrh s.
3. Dok je brojač različit od nule radi
    4. Ako je trenutni vrh jednak ciljanom vrhu *t*, *prihvati*.
    5. Nedeterministički izaberi vrh koji može biti dohvačen iz trenutnog vrha.
    6. Zabilježi pokazivač na novi vrh (tj. izabrani vrh) i smanji brojač.
7. *Odbaci*. "

Nedeterministička log prostorna Turing mašina odlučuje *PATH* operišući sa početnim vrhom s i nedeterministički pogađajući vrh na puta od s do t. Mašina bilježi jedino poziciju trenutnog vrha na svakom koraku na radnoj traci, ne cijelog puta (koji bi prevazilazio granice logaritamskog prostora). Mašina nedeterministički izabere sljedeći vrh između onih u koje pokazuje trenutni vrh. Tako da ponavlja ovu akciju dok ne dođe do vrha t i tad prihvati, ili dok ne prođe m koraka i odbaci, gdje je m broj vrhova u grafu. Tako da je *PATH* u NL-u, što je i trebalo dokazati.

♣♣♣

Zadatak 2.6

Pokazati da je $A_{DFA} \in L$. A_{DFA} = {<*A, w*> | *A* je DFA i *A* prihvata ulazni string *w* }.

Riješenje

Konstruisat ćemo log prostornu Turing mašinu M_3 koja odlučuje A_{DFA}. Kad M primi ulaz <*A, w*>, DFA i string *w*, M_3 simulira *A* na *w* nastojeći pratiti *A*-ovo trenutno stanje i trenutnu poziciju glave, obezbjeđujući nove informacije kad na njih dođe, za ovo dvoje. Potreban prostor za izvođenje ove simulacije je *O*(log *n*) zato što M_3 može zabilježiti svaku od ovih vrijednosti tako što će sačuvati pokazivač svog ulaza.

M_3 = "Na ulazu <*A, w*>, gdje je *A* DFA i *w* neki string:
1. Definiši prostor za trenutno stanje i inicijaliziraj ga na početno stanje od A.
2. Definiši prostor za trenutnu poziciju glave i inicijaliziraj je na početak ulazne trake.
3. Simuliraj *A* na ulazu *w* i svaki put, kad za neki simbol od w, DFA *A* dođe u novo stanje sačuvaj to stanje i sačuvaj novu poziciju glave ulazne trake."

♣♣♣

Zadatak 2.8

Dokazati da je jezik B = {w | w ∈ { (,) }* je string koji sadrži ispravno zatvorene zagrade}. Na primjer ((())) i (()()(())) pripadaju klasi L ali ((())) i)(ne pripadaju. Pokazati da je B ∈ L.

Rješenje

Dovoljno je da pokažemo da možemo prepoznati da je dati ulaz u obliku ispravno zatvorenih zagrada korištenjem samo jednog brojača. Stavimo brojač na nulu i čitamo ulaz sa lijeva na desno. Za svako "(" povečamo brojač za jedan. Za svako ")" smanjimo brojač za jedan. Ako brojač ikad dobije negativnu vrijednost, odbacimo ulaz (zato što smo zatvorili zagradu prije nego što smo je

otvorili). Ako brojač nije nula kad završimo čitanje, tad isto odbacujemo ulaz (zato što broj otvorenih i zatvorenih zagrada nije jednak). U suprotnom, prihvatamo ulaz.

♣♣♣

Zadatak 2.9

Pokazati da je NL klasa jezika zatvorena pod operacijama unija, komplement i zvijezda.

Rješenje

Neka su A_1 i A_2 jezici koji su odlučivi sa NL-mašinama N_1 i N_2. Konstruisat ćemo tri Turing mašine:

N_U koja će odlučivati $A_1 \cup A_2$;

$N_\circ$ koja će odlučivati $A_1 \circ A_2$; i

N_* koja će odlučivati A_1^*.

Svaka od ovih mašina prima ulaz *w*.

Nedeterminističke branše mašine N_U istovremeno simuliraju N_1 i simuliraju N_2. U bilo kojem slučaju, N_U prihvata ako bilo koja od simuliranih mašina prihvata.

Mašina $N_\circ$ nedeterministički izabere poziciju na ulazu, koji će string *w* podjeliti na dva podstringa. Jedino pokazivač te pozicije je sačuvan na radnoj traci - uopšte nam ne treba prostor da sačuvamo same podstringove. Poslije toga $N_\circ$ simulira N_1 na prvom podstringu, nedeterministički praveći branše koje su potrebne da simuliraju nedeterminizam od N_1. Ako neka od branši dospije u N_1 prihvatljivo stanje, tad $N_\circ$ simulira N_2 na drugom podstringu. Ako bilo koja od tih branša dođe do N_2 prihvatljivog stanja, $N_\circ$ prihvata.

Mašina N_* ima malo kompleksniji algoritam, pa ćemo opisati njegova stanja.

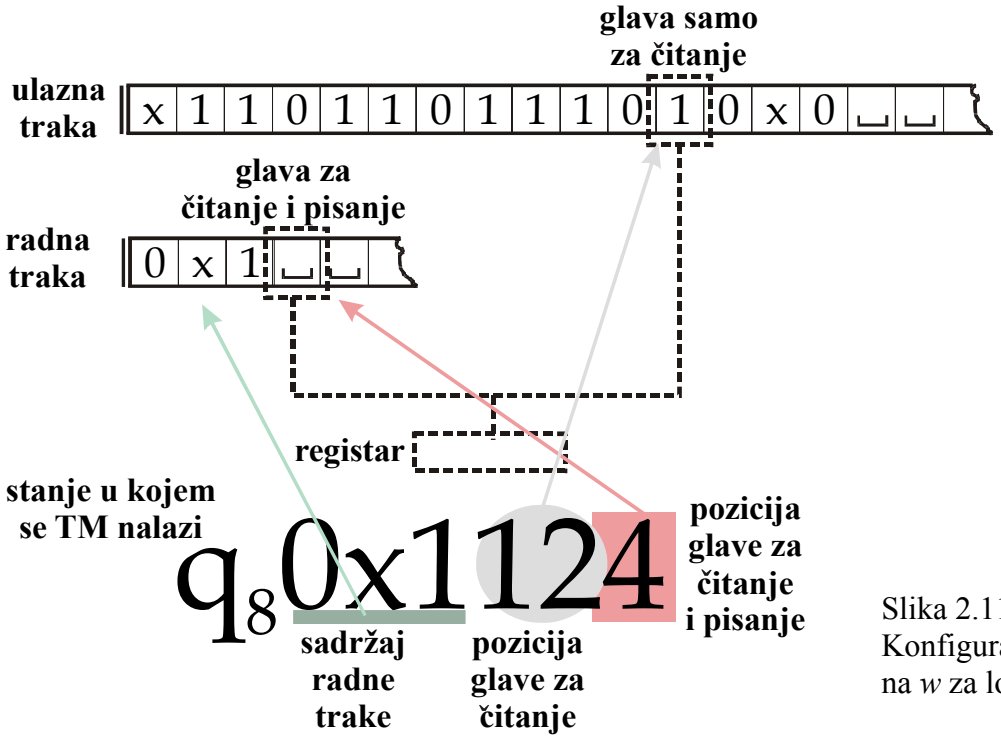
N_*="Na ulazu *w*:

1. Inicijaliziraj pozicije dva ulazna pokazivača p_1 i p_2 na 0, i za date pozicije odmah razmatraju prvi ulazni simbol.
2. Prihvati ako nema ulaznog simbola iza p_2.
3. Pomjeri p_2 naprijed na nedeterministički izabranu ulaznu poziciju.
4. Simuliramo N_1 na podstringu od *w*, čiji je sadržaj od pozicije p_1, pa sve do pozicije p_2, nedeterministički praveći branše koje su potrebne za simuliranje nedeterminizma od N_1.
5. Ako ova branša simulacije dođe do N_1 prihvatljivog stanja, kopiraj p_2 u p_1 i idi na korak broj 2."

♣♣♣

f(n) prostorno ograničena Turing mašina može koristiti najviše $2^{O(f(n))}$ vremena. Ova tvrdnja nije tačna za vrlo mala prostorna ograničenja. Na primjer, Turing mašina koja koristi $O(1)$ (tj konstanta) prostora može se završiti u n koraka. Da omogučimo ograničenje potrebnog vremena koje možemo primjeniti za svako prostorno ograničenje f(n) dajemo sljedeću definiciju.

Definicija 2.10 Ako je M Turing mašina koja ima odvojenu ulaznu traku samo za čitanje i w je ulaz, **kofiguracija od *M* na *w*** je podešenje koje se sastoji od trenutnog stanja mašine, sadržaja radne trake, i pozicije dvije glave na trakama.Ulaz w nije dio konfiguracije od *M* na *w*. ♠



Slika 2.11: Konfiguracija od *M* na *w* za log TM

Ako *M* koristi *f(n)* prostora i *w* je ulaz dužine n, broj konfiguracija od M na w je $n2^{O(f(n))}$. Da objasnimo ovaj rezultat, recimo da *M* ima *c* stanja i *g* simbola za traku. Broj stringova koji se mogu pojaviti na radnoj traci je $g^{f(n)}$. Ulazna glava može biti na jednoj od *n* pozicija i glava radne trake može biti u jednoj od *f(n)* pozicija. Tako da je ukupan broj konfiguracija od M na w, što je ujedno i gornja granica potrebnog vremena od *M* na *w*, je $cnf(n)g^{f(n)}$, ili $n2^{O(f(n))}$.

Navešćemo još jedan teorem za čiji dokaz nam treba poznavanje klase jezika NL-complete. coNL je klasa jezika *J* takvih da je $J \in$ NL.

Teorem 2.12 NL = coNL.

Zadatak 2.13

Dokazati da je 2-SAT = {< ϕ > | ϕ je zadovoljavajuća 2cnf-formula} u NL kalasi jezika. Skraćenica cnf je od konjunktivna normalana forma.

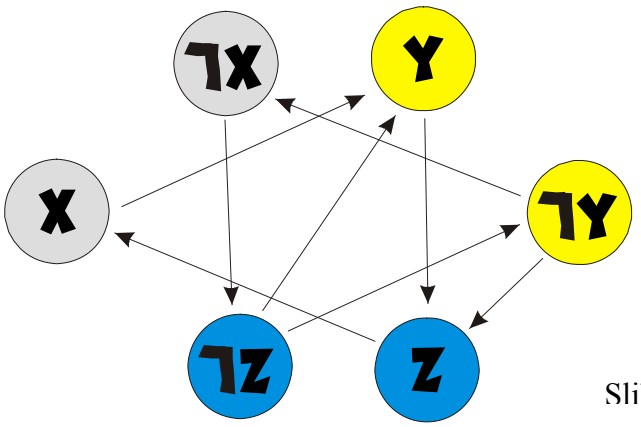
Rješenje

Ideja za riješenje ovog zadatka je da enkodiramo 2cnf-formulu u orjentisan graf G(ϕ). To ćemo postići tako što čemo vrhove od G(ϕ) predstaviti sa varijable od ϕ i njihovim negacijama. Ivice od G(ϕ) ćemo naštimati tako da postoji ivica (α, β) (čitaj iz α u β) ako i samo ako postoji klauzula (¬α, β) (ili (¬β, α)) u ϕ. Intuitivno, ivice možemo predstaviti pomoću logičke implikacije (⇒) zato što je:

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \equiv (\neg\alpha \vee \beta) \equiv (\beta \vee \neg\alpha).$$

G(ϕ) je simetričan graf, tj. (α ∨ β) je ivica ako i samo ako (¬α ∨ ¬β) je ivica. Primjetimo i to da putevi u G(ϕ) odgovaraju ispravnim implikacijama (tranzitivnošću od ⇒). Primjer konstrukcije:

$$(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$$



Slika 2.14

Navešćemo i dokazati jednu lemu koja će nam pomoći u rješavanju našeg zadatka.

Lema 2.15 ϕ je nezadovoljavajuća cnf formula ako i samo ako postoji varijabla x takva da postoji put iz x u $\neg x$ i iz $\neg x$ u x u grafu G(ϕ).

Dokaz:

Potreban uslov. Pretpostavimo da postoji varijabla x takva da postoji put iz x u $\neg x$ i iz $\neg x$ u x u grafu G(ϕ) i pretpostavimo da ϕ može biti zadovojena sa nekom dodjelom T. Ako pretpostavio da je T(x) = Tačno tada je T($\neg x$) = Netačno. Kako postoji put iz x u $\neg x$ to postoji i ivica (α, β) na ovom putu takva da je T(α) = Tačno i T(β) = Netačno. Ali (α, β) odgovara klauzuli ($\neg\alpha \vee \beta$) u ϕ, čega zaključujemo da ϕ nije zadovoljena. Došli smo do kontradikcije sa tvrđenjem da ϕ može biti zadovojena sa nekom dodjelom T. Ako pretpostavimo da je T(x) = Netačno, dokaz je sličan.

Dovoljan uslov. Pretpostavimo da ne postoji varijabla x takva da postoji put iz x u $\neg x$ i iz $\neg x$ u x u grafu G(ϕ). Pokazačemo da tad ϕ može biti zadovojena sa nekom dodjelom. Ponavljaj sljedeću proceduru dok svaki vrh ne dobije tačnu vrijednost:

1. Izaberimo neki vrh α kojem nismo dodijelili vrijednost (a prema pretpostavci za taj vrh ne postoji put od α do $\neg\alpha$).
2. Dodjelite Tačnu vrijednost vrhu α, i svim vrhovima koji se nađu na putu od α, i
3. Dodijeli Netačnu vrijednost svim vrhovima koji se nađu na putu iz vrha $\neg\alpha$.

Primjetimo da x i $\neg x$ su uvijek dodjeljene vrijednosti u istoj rundi. Ne može se desiti nekakav sukob između navedenih argumenata, zato što bi u tom slučaju bilo, ako bi postojao put iz α u oba vrha β i $\neg\beta$, i tada bi, kako je G(ϕ) simetričan, postojali puteviiz $\neg\beta$ i β u $\neg\alpha$.(pa bi postojao put iz α u $\neg\alpha$, što je u kontradikciji sa pretpostavkom). Ako bi postojao put iz α u vrh kome je dodjeljena vrijednost NETAČNO u bilo kojoj ranijoj rundi, tada bi, po trečem pravilu procedure, vrhu α bila dodjeljena vrijednost (NETAČNO) u toj ranijoj rundi. Nakon što je svim vrhovima dodjeljena vrijednost, ne postoji ivica koja vodi iz TAČNO u NETAČNO, prema konstrukciji. Tako da je, svaka klauzula zadovoljavajuća i ϕ je zadovoljavajuća cnf formula, što je i trebalo dokazati.

Sad iz ove leme možemo lakše shvatiti da je 2-SAT $\in$ NL. Zaista ovaj jezik je u coNL zato što možemo prepoznati nezadovoljavajuću instancu u log prostoru, tako što ćemo pogađati varijablu x i put iz x u $\neg x$ i nazad.

Sljedeća nedeterministička log prostorna Turing mašina odlučuje $\overline{\text{2-SAT}}$:

M_6 = "Na ulazu <ϕ>, gdje je <ϕ> enkodiranje 2cnf-formule ϕ u graf G(ϕ) prema proceduri opisanoj na početku dokaza:

1. Definiši brojač i inicijaliziraj ga na broj vrhova u grafu.
2. Definiši pokazivač1 da čuva "početni vrh", i definiši pokazivač2 da čuva "trenutni vrh".
3. Nedeterministički izaberi vrh koji će postati početn vrh, i koji će postati trenutni vrh.
4. Dok je brojač različit od nule radi
    5. Nedeterministički izaberi vrh koji može biti dohvačen iz trenutnog vrha.
    6. Ako je novoizabrani vrh jednak negaciji početnog vrha tad radi
        7. Inicijaliziraj brojač na broj vrhova u grafu.

8. Inicijaliziraj početni vrh na novoizabrani vrh, i inicijaliziraj trenutni vrh na novoizabrani vrh.
9. Dok je brojač različit od nule radi
    10. Nedeterministički izaberi vrh koji može biti dohvaćen iz trenutnog vrha.
    11. Ako je novoizabrani vrh jednak negaciji početnog vrha, *prihvati*.
    12. Zabilježi pokazivač za trenutni vrh na novoizabrani vrh i smanji brojač.
13. Zabilježi pokazivač za trenutni vrh na noizabrani vrh i smanji brojač.
14. *Odbaci*."

Turing mašina M_6 nam govori da je $\overline{\text{2-SAT}} \in$ NL, iz čega slijedi da je 2-SAT $\in$ coNL. Prema Teoremi 2.12 zaključujemo da je 2-SAT $\in$ NL, što je i trebalo dokazati.

♣♣♣

Zadatak 2.16

Neorjentisan graf je **bipartitan** ako vrhove možemo podijaliti u dva skupa tako da sve ivice idu iz vrhova u jednom skupu u vrhove u drugom skupu. Pokazati da je graf bipartitan ako i samo ako ne sadrži konturu koja ima neparan broj vrhova. Neka je *BIPARTITE* = {<*G*> | *G* je bipartitan graf}. Pokazati da je *BIPARTITE* $\in$ NL.

Rješenje

Prvo ću pokazati da je graf bipartitan ako i samo ako ne sadrži neparnu konturu.

Pretpostavimo da je graf G bipartitan. Tada postoje dva skupa vrhova, označimo ih sa A i B, sa osobinom da sve ivice iz skupa A idu u skup B. U skupovima A i B nema ivica. Ako krenemo od nekog vrha *v* iz skupa A, taj vrh ili nije spojen ni sa jednom vrhom u grafu G ili je spojen sa nekim vrhom. Ako mu je stepen nula, za taj vrh graf G ne sadrži neparnu konturu. U slučaju da je spojen sa nekim vrhom *u*, taj vrh mora biti u skupu B. Ako je stepen od *u* manji od 2, graf ne sadrži neparnu konturu. Ako je stepen od *u* veći ili jednak 2, pored vrha *v*, u skupu A postoji još jedan vrh *x* koji ima zajedničku ivicu sa *u*. Vrh *x* ne može imati zajedničku ivicu sa *v*, zato što je graf G bipartitan. Ako je stepen od *x* manji od 2, graf G nema neparnu konturu. Ako je stepen od *x* veći ili jednak 2, pored vrha *u*, postoji još jedan vrh *y* u skupu B koji sa *x* ima zajedničku ivicu. Ako je stepen od y manji od 2, graf G ne sadrži neparnu konturu. Ako je stepen od *y* veći ili jednak 2, vrh *y* pored vrha *x* je povezan sa nekim vrhom u skupu A. Ako je to vrh *v*, graf ima konturu, ta kontura nije parna. Ako je to neki novi vrh koji je različit od v, proces nastavljamo. Zaključujemo da ni u jednom slučaju graf G ne sadrži neparnu konturu.

Pretpostavimo sad da graf G ne sadrži neparnu konturu. Dokaz da je graf bipartitan je vrlo sličan gornjem dokazu, mi ga nećemo navoditi, i ostavlja se čitaocu za vježbu.

Iskoristimo ovu činjenicu da pokažemo da je *BIPARTITE* $\in$ NL.

Konstruisat ćemo log prostornu nedeterminističku Turing mašinu M_7 koja odlučuje $\overline{\textit{BIPARTITE}}$., to jest kompliment od *BIPARTITE*. Iz gornje tvrdnje G nije bipartitan ako i samo ako sadrži neparnu konturu.

Definisaćemo brojač koji će sadržavati dužinu puta k. Nedeterministički čemo izabrati početni vrh i sačuvati ga. Za k $\leq$ n (n je broj vrhova u grafu G) pogađačemo sljedeći vrh *u* na putu. Ako se dogodi da je *u*=*v* i k je neparan prihvatamo s obzirom da graf mora sadržavati neparnu konturu.

M_7 = "Na ulazu <*G*>, gdje je G neki graf sa n vrhova:
1. Definiši prostor za brojač k i inicijalizirajmo ga na 1.
2. Definiši prostor za početni vrh i za vrh na putu.
3. Nedeterministički izaberimo početni vrh *v*.
4. Dok je k $\leq$ n radi sljedeće:
    5. Nedeterministički izaberi neki vrh *u*, koji se nalazi na putu iz prethodnog vrha.
    6. Ako su vrhovi *u* i *v* jednaki, i k je neparan broj, *prihvati*.
7. *Odbaci*."

Ovim smo pokazali da je *BIPARTITE* $\in$ coNL. Sad ćemo iskoristiti Teorem 2.10 koji tvrdi da je NL=coNL iz čega slijedi da je *BIPARTITE* $\in$ NL, što je i trebalo dokazati.

♣♣♣

Zadaci za vježbu

*__H1__ Igra **Nim** se igra sa nekoliko grupa štapića, gdje su grupe nezavisne jedna od druge. U jednom potezu igrač može ukloniti neki broj štapića, različit od nule, samo iz jedne grupe. Igrači se naizmjenično mjenjaju u potezima. Igrač koji uzme zadnji štapić gubi. Recimo da imamo poziciju u igri Nim sa k grupa koje sadrže $s_1, ..., s_k$ štapića. Poziciju zovemo **balansiranom** ako, kad je svaki od brojeva s_i napisan binarno i binarni brojevi su napisani kao redovi matrice čiji su elementi bitovi, svaka kolona od bitova sadrži paran broj 1-ca. Dokazati sljedeće dvije činjenice.

a. Počevši od nebalansirane pozicije, postoji jedinstven potez koji mijenja poziciju u balansiranu.

b. Počevši u balansiranoj poziciji, svaki potez koji odigramo mijenja poziciju u nebalansirajuću.

Neka je $NIM = \{< s_1, ..., s_k > |$ svaki s_i je u binarnoom obliku i Igrač I ima pobjedničku strategiju u igri Nim počevši od ove pozicije$\}$. Koristi navedene osobine o balansiranim pozicijama da pokažete da je NIM $\in$ L.

*__H2__ Neka je D jezik ispravno zatvorenih uglastih i običnih zagrada. Na primjer, ([()]()[]) je u D, ali ([)] nije. Pokazati da je D u L.

H3 Neka je $MULT = \{ a \# b \# c \mid a, b$ i c su binarni prirodni brojevi i $a \cdot b = c\}$. Pokazati da je $MULT \in$ L.

H4 Za svaki pozitivan cijeli broj x, neka je $x^{\mathbf{R}}$ cijeli broj ćija binarna reprezentacija je obrnuta od binarne reprezentacije x. (Pretpostavimo da nema vodeće 0-le u binarnoj reprezentaciji od x). Definišimo funkciju $\mathbf{R}^+: \mathbf{N} \rightarrow \mathbf{N}$ gdje je $\mathbf{R}^+(x) = x + x^{\mathbf{R}}$.

a. Neka je $E_2 = \{<x, y> \mid \mathbf{R}^+(x) = y \}$. Pokazati da je $E_2 \in$ L.

b. Neka je $E_3 = \{\{<x, y> \mid \mathbf{R}^+ (\mathbf{R}^+(x)) = y \}$. Pokazati da je $E_3 \in$ L.

*__H5__ Definišimo *UCYCLE*$=\{ < G > \mid$ G je neorjentisan graf koji sadrži jednostavnu konturu$\}$. Pokažite da je *UCYCLE* $\in$ L.

H6

a. Neka je *ADD*$= \{<x, y, z> \mid x, y, z > 0$ su binarni cijeli brojevi i $x+y=z\}$. Pokažite da je *ADD* $\in$ L.

b. Neka je *PAL-ADD* $= \{<x, y> \mid x, y > 0$ su binarni cijeli brojevi gdje $x + y$ je cijeli broj čija binarna reprezentacija je palindrom$\}$. (Primjetite da se pretpostavlja da binarna reprezentacija sume nema vodeću nulu. Palindrom je string koji je jednak svojoj inverziji.) Pokažite da je *PAL-ADD* $\in$ L.

Literatura

[MS] **Michael Sipser**, Introduction to the theory of computation, second edition; str. 303-334, *2006*
[MJ] **Matthew Johnson**, http://www.dur.ac.uk/matthew.johnson2/teaching/acc/; lecture 11-13, *2007*
[SA] **Sanjeev Arrora**, **Boaz Barak**, Computation Complexity: A Modrern Approach str.75-90, *2007*
[OG] **Oded Goldreich**, Computation Complexity: A Conceptual Perspective; str. 147-186, *2006*