PRVA LABARATORIJSKA VJEZBA IZ UTR-a 2012/2013

Znaci Vas zadatak je napraviti simulator nedeterministickog konacnog automata sa epsilon prijelaza. U ovom dokumentu prvo cu objasniti sto je to nedterministicki konacni automat, sto su epsilon prijelazi i kako on funkcionira. Naslovi lijepo imenuju ono o cemu se u odjelku govori tako da mozete preskociti ono sto vec znate, a potrudit cu se skratiti sto vise.

U nekim djelovima ce vam se mozda ciniti da pisem ocite stvari, ali pisem ih za svaki slucaj da bi svakome bilo jasno.

DETERMINISTICKI KONACNI AUTOMAT

Dakle konacni automat si mozete zamisliti kao neki stroj koji ima svoje stanje i ima nekakav ulaz gdje se dovode nekakvi ulazni podaci. E sad ovisno o tome sto vas automat dobije na ulaz napravit ce prijelaz u neko novo stanje koje moze biti i isto stanje u kojem vec i je. Automat je definirat sa nekim skupom stanja, znaci stanja koja postoje za taj automat, dalje definiran je mogucim ulazima, nazvat cemo ih ulazni znakovi. Znak moze biti ili doslovno jedan character (znak) ili cijeli niz znakova (String) ili bilo sto sto vi za vas automat kazete da ce biti, znaci mozete ga definirati bilo kako (doci cemo dotoga kako je on definiran u vasem zadatku). Osim tog skupa stanja i ulaznih znakova automat ima i definiranu tablicu prijelaza po kojoj automat zna u koje ce stanje preci. A odluku donosi na temelju stanja u kojem se nalazi i ulaznog znaka. Znaci automat ce kad na ulaz dobije neki znak pogledati u tablicu gdje mu pise prijelaz za stanje u kojem se on trenutno nalazi i ulazni znak koji je dosao te ce promjeniti svoje stanje u ono koje je zapisano u tablici.

TRENUTNO STANJE	ULAZNI ZNAK	NOVO STANJE
q0	a	q1
q0	b	q2

Evo ovo je primjer nekakve tablice znaci kada se automat nalazi u stanju q0 i na ulazu dobije znak a preci ce u stanje q1, a ako dobije znak b preci ce u stanje q2. U ovoj tablici druge kombinacije nisu definirane sto znaci da ih nema i da u njima automat ne radi. Vidjet cemo sto vi trebate u takvom slucaju napraviti u vasem labosu.

Jos jedna stvar, automat osim svega sto smo rekli ima definirano jos i pocetno stanje te

prihvatljiva stanja. Pocetno stanje je stanje od kojeg automat krece, znaci na pocetku rada kad ga “upalite” nalazi se u tom pocetnom stanju i ono ovdje moze biti samo jedno. Dalje kad vas automat vise nema nista na ulazu on je zavrsio s radom i stanje u kojem se on nalazi je njegovo zavrsno stanje. E sada postoje stanja koja su prihvatljiva i ona koja nisu, sto to znaci. To znaci da to stanje u kojem automat zavrsi ako je ono prihvatljivo onda se niz ulaznih znakova koji su dosli na ulaz automata prihvaca, a ako zavrsno stanje nije prihvatljivo onda se taj niz ne prihvaca.

Koja ce stanja biti prihvatljiva, koje stanje je pocetno to znate jer vam to kazu, time je automat definiran i to smislja tvorac automata koji vam da sve potrebne podatke, a to su kao sto smo naveli:

skup svih stanja, pocetno(koje je jedno iz tog skupa), prihvatljiva (koja su takodjer iz skupa svih stanja), skup ulaznih znakova i tablica prijelaza.

NEDETERMINISTICKI KONACNI AUTOMAT

Evo nas dalje. Ako ste shvatili koncept konacnog automata iz proslog “poglavlja” ovo vam nece biti problem.

Dakle sve je isto kao kod obicnog (deterministickog) automata s jednom sitnom ali bitnom razlikom. Razlika je u prijelazenju u “novo” stanje. Znaci nas nedeterministicki automat je neodlucan sto znaci da se on ne treba odluciti za jedno stanje tj. moze u **jednom** prijelazu preci u vise stanja i tako se u **istom** trenutku nalaziti u vise stanja **odjednom**. Ovo mozda iz prve zvuci zbunjujuce, ali ustvari je vrlo jednostavno, vrijede ista pravila samo sto sada kada radimo prijelaz moramo provjeriti prijelaze iz **svih** trenutnih automatovih stanja za procitani ulazni znak i preci u **sva** stanja u koja se prelazi **svim** kombinacijama trenutnih stanja i ulaznog znaka. Ako za neko od trenutnih stanja nije definiran prijelaz (nema ga u tablici) onda za to stanje prijelaz niti ne radimo nego ga “preskocimo”. Znaci radimo prijelaze samo za ona stanja za koje je prijelaz definiran (postoji u tablici stanja). Naravno ako prijelazi postoje za vise trenutnih stanja napravit cemo ih sve i **sva** stanja u koja smo presli cinit ce **novi** skup trenutnih stanja u kojima ce se nas automat nalaziti.

TRENUTNO STANJE	ULAZNI ZNAK	NOVO STANJE
q0	a	q1.q3
q0	b	q2
q0	a	q2

Evo primjera za tablicu stanja nedeterministickog automata. Dakle ako se nalazimo u stanju q0 i dodje nam na ulaz znak a preci cemo u **3 stanja**, (q1, q2, q3). U q1 i q3 sa prijelazom definiramim u prvom retku te u q2 sa prijelazom definiranim u trecem retku tablice.

Nedeterministiki automat takodjer ima “pocetno stanje” i prihvatljiva stanja. Ovi navodnici kod pocetnog stanja su tu zato sto nedeterministicki automat moze imati vise pocetnih stanja posto se u jednom trenutku moze nalaziti u vise stanja odjednom, tako moze i na pocetku rada.

Ako se pitate kako znamo prihvaca li se ulazni niz ili ne odgovor stize. Dakle kada automat zavrsi s radom naci ce se ili u jednom ili vise stanja i ako je **bilo koje** stanje od tih u kojima je automat zavrsio **prihvatljivo** stanje, tada se niz **prihvaca**.

NEDETERMINISTICKI AUTOMAT S EPSILON PRIJELAZIMA

Dakle prvo da kazem sto je to epsilon prijelaz. Epsilon prijelaz je prijelaz koji automat radi bez da procita znak s ulaza. Dakle automat iz svakog stanja za koje je definiran epsilon prijelaz automat moze prijeci u novo stanje bez da procita znak te znak citati nakon obavljenog epsilon prijelaza.

Epsilon prijelazi se ne rade bezveze, oni su takodjer definirani u tablici prijelaza i oznacavaju se nekim znakom koji tvoritelj automata izabere, a u vasoj labaratorijskoj vjezbi to je znak $ koji se nalazi u na mjestu gdje bi inace bio neki ulazni znak.

Nedeterministiki automat s epsilon prijelazima je dakle kao sto ime kaze najobicniji nedeterministcki automat koji moze imati definirane i epsilon prijelaze.

E sad kakve probleme to moze stvarati. Pa u svakom stanju u kojem se automat nalazi on moze i napraviti epsilon prijelaz bez da cita znak, pa sa novim stanjima citati znak ili moze prije epsilon prijelaza iz starih stanja citati znak i raditi za njega definirane prijelaze. Takodjer automat moze nakon nekog epsilon prijelaza ponovo raditi epsilon prijelaze koliko god da ih posoji. Automat takodjer moze epsilon prijelazom ici iz stanja q0 u q1 te imati definiran epsilon prijelaz iz stanja q1 u q0, pa iz q0 opet epsilon prijelaz u q1 i tako do beskonacnosti.

NATUKNICE I PONUDJENA RJESENJA ZA NEKE OD NAVEDENIH PROBLEMA

Dakle pocnimo od pocetka iako su neke stvari objasnjene u pdfu za labos.

Dakle vas simulator ce na ulaz dobiti datoteku s definicijom nekog automata bilo kojeg i on treba simulirati njegov rad kao sto je opisano u ranijem poglavlju da taj automat radi.

Vas ce program prvo trebat procitat tu datoteku i pospremit si to nekako da to moze kasnije koristiti.

1) u prvom redku se nalaze ulazni nizovi koje cete dovost na ulaz automata opisanog u ostatku datoteke. Za prijelaz napravljen bilokojim ulaznim znakom trebate ispisati stanja u kojem se automat nalazi. I to svaki nis u svom redu ispisujete. Ulazni nizovi su razdjeljeni znakom | . A znakovi ulazih nizova su medjusobno odvojeni znakom zarez.

2) u drugom retku su nazivi svih stanja odvojeni zarezom, ime jednog stanja sastoji se od malih slova engleske abecede i dekadskih znamenki. Znaci npr (123, a, sismis, 5aoiu65) su sasvim legalni nazivi stanja, a vi ih mozete lijepo spremiti u nekakvu listu stringova di je u stringu zapisano stanje.

3) u trecem retku nalaze se ulazni znakovi takodjer odvojeni zarezom i jedan ulazni znak takodjer je neki string koji se sastoji od malih slova engleske abecede i/ili dekadskih znamenki. Tu isto mozete to pospremiti u nekakvu listu stringova gdje je svaki element liste string koji predstavlja ulazni znak.

4) u cetvrtom retku se nalaze navedena prihvatljiva stanja koja su odvojena zarezom. posto prihvatljiva stanja spadaju u skup svih stanja svako stanje se takodjer moze sastojati od malih slova engleske abecede te dekadskih znamenaka. Ovo takodjer mozete spremiti u listu stringova koja predstavlja listu prihvatljivih stanja.

5) u petom retku nalazit ce se **jedno** pocetno stanje koje mozete spremiti u nekakav string.

6) od 6tog retka pa nadalje ide tablica prijelaza koja nece bit ovak lijepo nacrtana kao moje nega ce imti zapis formatiran na sljedeci nacin:
<ime_stanja>,<ulazni_znak>-><skup_sljedecih_stanja_odvojeni_su_zarezom>
(primjer imate u pdfu gdje je zadatak)

Ok sad ovo isto nekako trebate pospremiti u memoriju sto ustvari nije ni toliko kompliciranо.

Znaci vi za neko stanje i neki ulazni znak zelite dobiti listu stanja u koja se tom kombinacijom prelazi. Pa takav koncept postoji i vecini je sigurno poznat. Rijec je o mapi, dictionariju, asocijativnom polju. (to su samo drugaciji nazivi za istu stvar).

Znaci onako kako ste u obicnom polju pristupali elementima pomocu indexa koji je bio neki broj.

Tako u ovakvoj strukturi podatako mozete pristupiti nekom elementu pomocu nekog kljuca, a taj kljuc moze prakticki biti bilo sto. I ovdje u ovom primjeru dobro bi bilo imati dakle tako nekakav rjecnik/mapu koja kao kljuceve ima stanje i ulazni znak i za taj par vraca listu stanja u koja se tom kombinacijom (tim parom) prijelazi.

E sad ovisi koji jezik koristite:

u pythonu npr mozete imati za kljuc touple koji se satoji od 2 elementa, ime stanja (string) i ulaznog znaka (string), a za vrijednost ima listu stringova (sljedecih stanja).

u c++u npr mozete imati za kljuc pair<string, string> i za vrijednost list<string>

i slicno je u ostalim jezicima, iznimka je c, ali ako radite u cu, vjerojatno i znate nesto vise pa vam ovaj tutorial nije potreban, ako pak planirate raditi u cu a i ovo vam se ne cini jednostavno, nemojte raditi u cu.

Ok sad smo otprilike rekli kak bi to izgledalo u memoriji (samo to trebate tako pospremiti sada)

I sada trebate pokrenuti vas automat. Znaci jednostavan ne skroz potpun pseudokod bi izgledao nekako ovako:

```
postavite pocetno stanje na pocetno stanje koje ste spremili u
string;

     petlja:
Naprvaiti sve definirane epsilon prijelaze za sva trenutna stanja i
dodati ih u listu trenutnih stanja.

Za zva trenutna stanja napraviti sve prijelaze za ulazni znak na
kojem se nalazimo. Ta stanja stavljamo u neku novu listu sljedecih
stanja.

pomaknemo se na sljedeci znak;
postavimo listu trenutnih stanja na stanja iz liste sljedecih stanja
i ispraznimo listu sljedecih;
```

Eto tako nekako, petlja se vrti dok god ima prijelaza. Prijelaz radite tako da pogledate na MOJA_MAPA[STANJE,ZNAK] i vidite sto cete dobiti. Ako tamo nema nista znaci da tamo niste nista stavli i ako ste radili dobro niste nista stavili jer niste imali nista za stavit jer tako pise u ulaznoj datoteci, a to sve skupa znaci da prijelaz nije definiran i nista idete dalje. A ako dobijete nazad listu sljdeckih stanja lijepo ih dodate u vasu listu di stavljate sva sljedeca stanja od svih prijelaza u jednom trenutku.

Jos trebate sve to skupa staviti u jednu petlju jer ova petlja je za izvodjenje jednog ulaznog niza, a vi imate

listu ulzanih nizova te trebate proci po svima njima i sve ih simulirati.

Nododat cu jos da u vasem labosu je zadano da ustvari uvijek prividno imate prijelaz, tj. ako prijelaza nema ve prelazite u prazno stanje koje je kod vas oznaceno sa #. Da Vas ne bunim, jedino sto to kod vas znaci je da ako nemate za radit prijelaza necete prekinut izvodjenje automata nego cete nastaviti dok ne prodjete sve znakove. A u ispisu cete ispisati da se ne nalazite u niti jednom stanju (tj da se nalazite u praznom skupu stanja) tako da ispisete znak ljestve (#).

Evo to je to, puno sam toga naprisao i nadam se da sam bio jasan. Ako nesto nije jasno slobodno pitajte :)