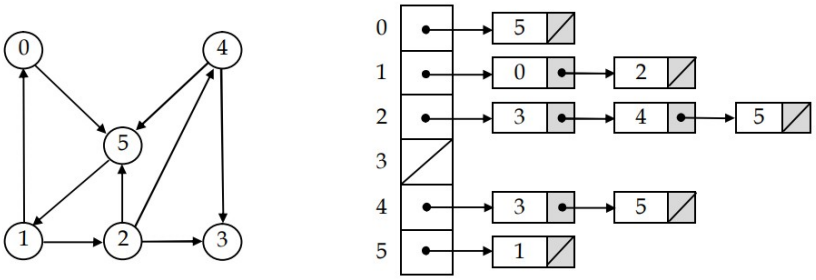
Predmet: Automati i formalni jezici
Godina studija: II
Semestar: IV
Akademska godina: 2017/2018.
Zadaća 3

Zadaća 3

Rok za predaju zadaće je srijeda, 30.05.2018 u 23:59.

Potrebno je implementirati klasu pod nazivom `EpsilonNka` u programskom jeziku *C++* koja predstavlja nedeterministički konačni automat sa *ε*-prelazima (*ε-NKA*). Glavna svrha klase `EpsilonNka` jeste da omogući kreiranje *ε-NKA* na osnovu zadanog regularnog izraza i provjeru da li je određena riječ prihvaćena automatom ili ne. Automat je potrebno implementirati kao usmjereni graf predstavljen preko **liste susjedstva**. Primjer ovakvog načina predstavljanja grafa je dat na slici 1:



Slika 1

Slika prikazuje generalni način predstavljanja grafa putem liste susjedstva. S obzirom da konačni automat za svaku granu(prelaz) dodatno sadrži i tačno jedan simbol za koji se vrši taj prelaz, svaki čvor liste će pored rednog broja čvora koji označava u koji se čvor prelazi sadržavati još i simbol za koji se taj prelaz ostvaruje. Shodno tome, vaš zadatak jeste da definirate jednu strukturu unutar klase `EpsilonNka` (možete je nazvati `Cvor`) koja će sadržavati dva atributa: jedan atribut tipa `int` koji predstavlja broj čvora(odnosno u našem slučaju stanja) u koji se prelazi i drugi atribut tipa `char` koji predstavlja simbol za koji se vrši taj prelaz. Pri tome ćemo *ε*-prelaze označavati malim slovom '*e*'. Prilikom implementacije listi susjedstva potrebno je korisititi klasu `std::list`. Ukoliko želite, umjesto definiranja vlastite strukture `Cvor` možete koristiti klasu `std::pair<int, char>` iz standardne biblioteke.

Sada možemo definirati izgled klase `EpsilonNka`. Klasa treba sadržavati sljedeće privatne atribute:

- `vector<list<Cvor>* > delta` ili `vector<list<pair<int, char> >* > delta` – Atribut delta koji je vektor pokazivača na listu čvorova. Ovaj vektor predstavlja stanja automata(stanja predstavljamo cijelim brojevima redom počevši od nule – dakle indeksi vektora delta predstavljaju oznake stanja). Svaki element vektora treba da pokazuje na listu čvorova sa kojima je taj čvor povezan odnosno listu stanja u koje to stanje ima prelaze, pri čemu su čvorovi u toj listi definirani na način objašnjen ranije.

  Primjer: Neka stanje 0 ima prelaz u stanje 1 za simbol a i epsilon prelaz u stanje 2. Tada će nulti element vektora delta pokazivati na listu koja sadrži dva čvora: <1, 'a'> i <2, 'e'>.

- `std::set<char> alfabet` – Atribut alfabet je skup simbola alfabeta jezika kojeg automat prihvata.

- `int pocetnoStanje` – Redni broj početnog stanja automata odnosno indeksa vektora delta koji predstavlja početno stanje

- `int finalnoStanje` – Redni broj finalnog stanja automata odnosno indeksa vektora delta koji predstavlja finalno stanje. Iako automat po pravilu može imati više finalnih stanja, za kreiranje samog automata će se koristiti postupak opisan u predavanju prema kojem na kraju uvijek dobijamo samo jedno finalno stanje pa ćemo radi jednostavnosti u klasi omogućiti samo jedno finalno stanje koje će se pamtiti u ovom atributu.

Klasa `EpsilonNka` treba sadržavati sljedeće javne metode:

- `EpsilonNka(const string &regex)` – Konstruktor koji kao parametar prima string koji predstavlja regularni izraz automata. Ovaj konstruktor treba da kreira automat koji prihvaća jezik opisan ovim regularnim izrazom prema postupku opisanom u predavanjima. Radi lakše implementacije, regularni izraz će biti zapisan u **postfiksnoj** notaciji. Prema ovoj notaciji operator se piše nakon svojih operanada(ili operanda ukoliko se radi o unarnom operatoru). Nekoliko primjera ove notacije je dato u nastavku(koristit ćemo simbol '.' za operaciju množenja odnosno nadovezivanja regularnih izraza):

a+b -> ab+

a+ba -> aba.+

(a+b)a -> ab+a.

(a+b) -> ab+**

a+ba -> a*ba.+*

(aa + bab)b -> aa*.ba.b.+b.*

Dakle, za postfiksnu notaciju nisu potrebne zagrade jer se redoslijed operacija sam definira u zavisnosti od pozicije i redoslijeda operatora. Radi jednostavnosti, pretpostavit ćemo da je regularni izraz uvijek u ispravnom formatu i da sadrži samo dozvoljene simbole(mala slova engleskog alfabeta *a-z* osim slova *e* koje je rezervisano za *ε* - prelaze, *+*, . i ***) tako da to nije potrebno posebno provjeravati.

- `bool prihvataRijec(const string &rijec)` – Ova metoda prima jedan parametar koji je također tipa `string` i koji predstavlja određenu riječ. Metoda treba da vrati `true` ukoliko automat prihvata ovu riječ, u suprotnom potrebno je vratiti `false`.

Potrebno je osigurati da nema curenja memorije(ili sličnih memorijskih problema tipa pristup ilegalnom pokazivaču i sl.). Također, dozvoljeno je i poželjno dodavanje **privatnih** metoda po želji s obzirom da će vam to pomoći pri izradi logike zadatka. Preporuka je da za sve pojedinačne korake procesa kreiranja automata iz regularnog izraza kreirate posebne privatne metode te na taj način razdvojite pojedinačne zadatke. Također, dozvoljeno je korištenje različitih struktura podataka(iz standardnih biblioteka ili vaše implementacije) ukoliko smatrate da vam one mogu pomoći pri izradi zadatka.

Potrebno je i napraviti `main` u kojem ćete istestirati vašu klasu sa bar 5 regularnih izraza i bar 3 riječi za svaki regularni izraz(sa pozitivnim i negativnim primjerima).