­

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه اول درس نظریه زبان‌ها و ماشین‌ها

نگارش:

امیرحسین سرور - 9731028

استاد درس:

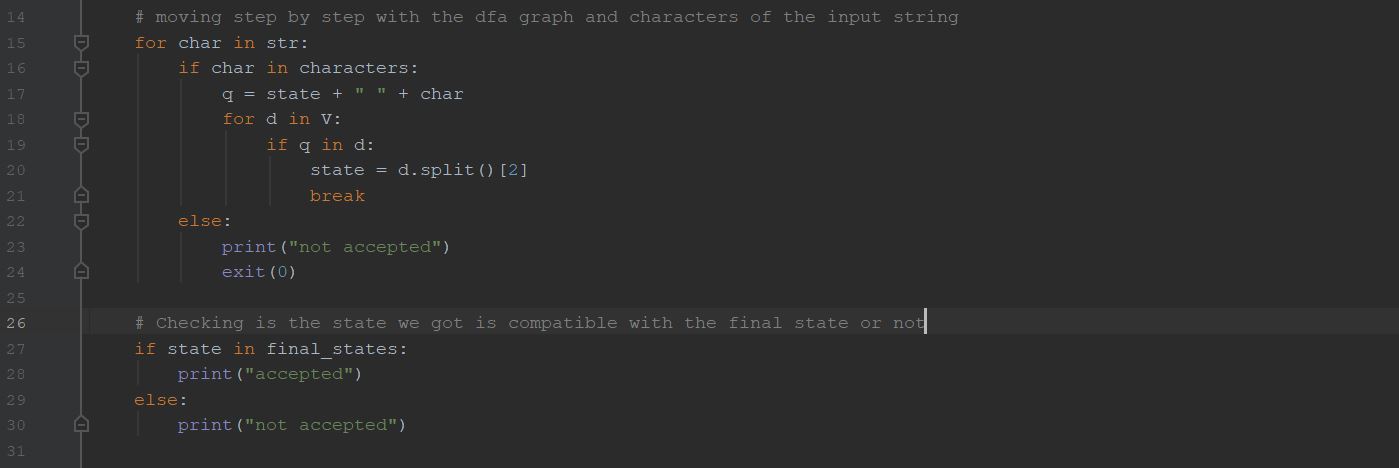
دکتر محمدرضا میبدی

اردیبهشت 1399

* سوال اول :

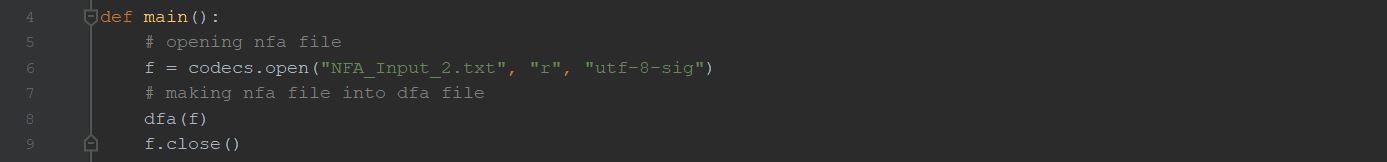


در ابتدا رشته از کاربر گرفته می‌شود. سپس فایل DFA را باز کرده و اطلاعات مربوط به گراف DFA در برنامه ذخیره می‌شود.



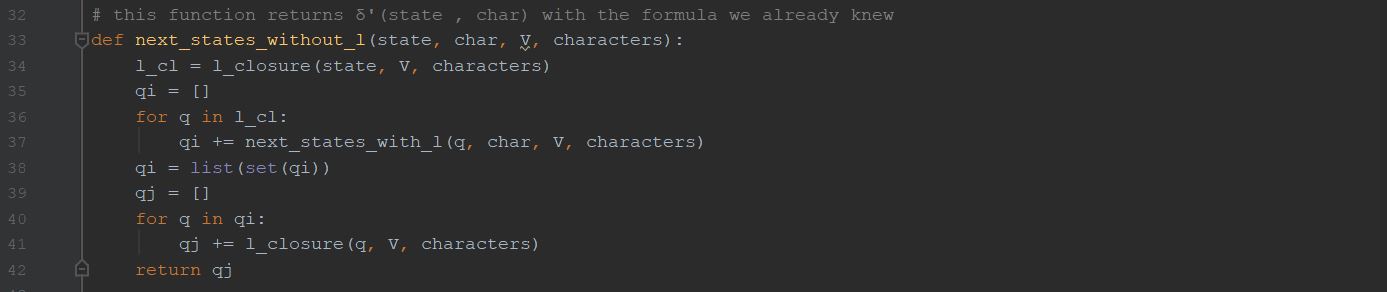
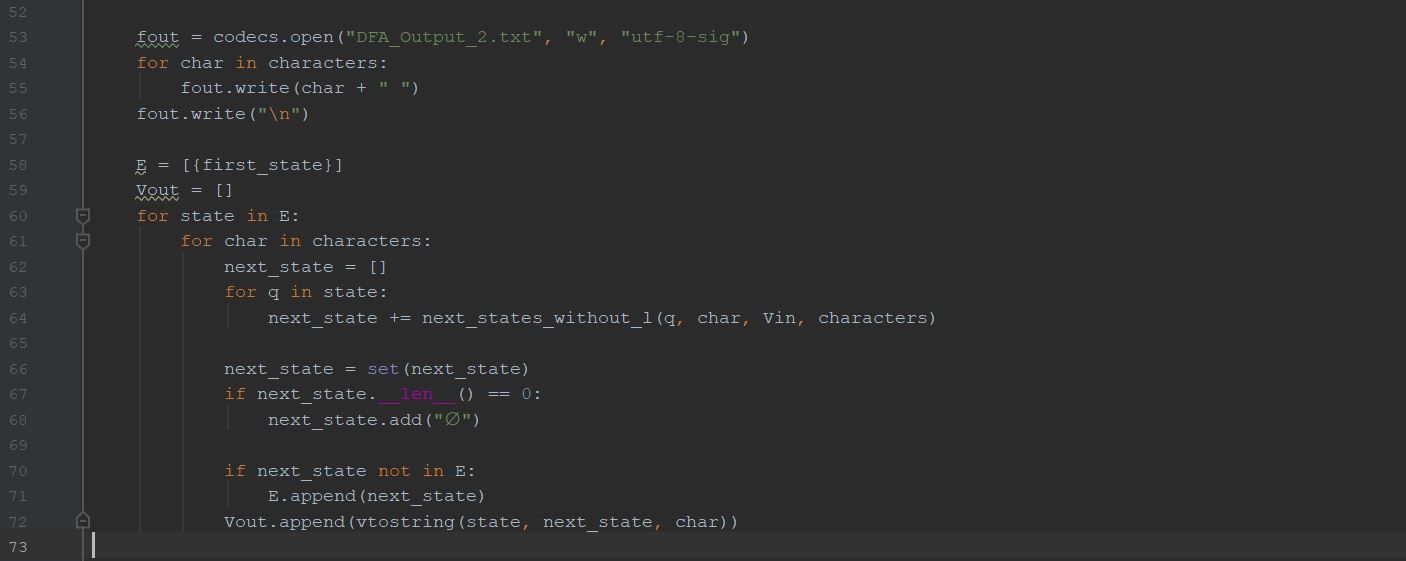
در مرحله بعد با اعمال for روی رشته وارد شده، کاراکترهای آن را به ترتیب می‌خوانیم و با توجه استیت فعلی و اطلاعاتی که از DFA داریم، یک رشته از استیت فعلی و کاراکتر خوانده شده تشکیل می‌دهیم و اگر این رشته جزو یکی از یال‌های DFA باشد، استیت فعلی را به قسمت سوم آن در فایل txt (یال) که در واقع استیت بعدی می‌باشد، تغییر می‌دهیم. همچنین چون در این سوال DFA داریم مطمئن هستیم یک یال با همچین مشخصاتی وجود دارد. این مراحل را تا زمانی که آخرین کاراکتر رشته موردنظر خوانده شود ادامه می‌دهیم و در آخر، اگر استیت فعلی جزو یکی از استیت های نهایی بود، عبارت accepted و در غیر اینصورت عبارت not accepted در خروجی چاپ می‌شود.

* سوال دوم :



بدنه اصلی که شامل تابع main است تنها از صدا زدن تابع dfa تشکیل شده است که وظیفه‌ی پیاده‌سازی الگوریتم تبدیل NFA بدون λ به DFA را دارد.

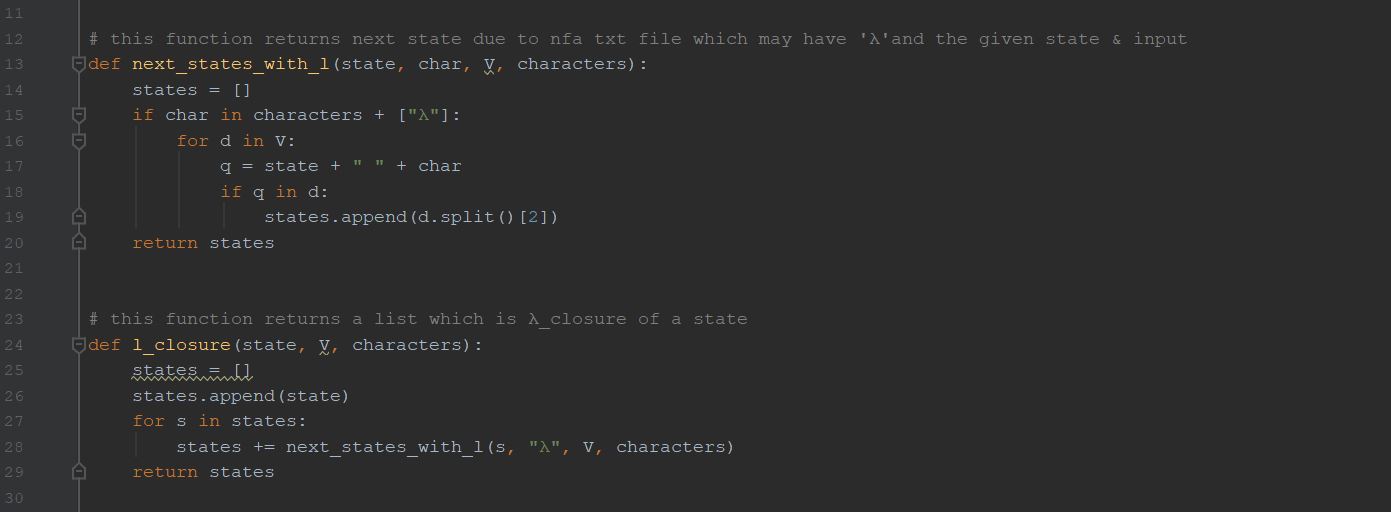
در تابع dfa ، بعد ازگرفتن و خواندن فایل NFA ، در فایل خروجی DFA ، ابتدا کاراکترهای استفاده شده نوشته می‌شوند. سپس آرایه از set ها تحت عنوان E برای نگه‌داری رئوس گراف تعریف می‌شود که اولین عنصر آن هم مشخص است. سپس روی این آرایه for زده تا رئوس بعدی و یال‌های مربوط نیز مشخص شوند. یال‌ها هم که در نهایت باید در فایل خروجی نوشته شوند، در آرایه‌ای تحت عنوان Vout نگه‌داری می‌شوند. در این for ، حالت بعدی هر استیت در DFA توسط اجتماع گیری از استیت‌های بعدی در گراف NFA بدون λ حالات موجود در استیت فعلی مشخص می‌شود. اگر حالت بعدی در رئوس فعلی موجود نبود، یک رأس جدید اضافه شده و اگر تهی باشد، به استیت ∅ هدایت می‌شود. در آخر یال جدید نیز به Vout اضافه می‌شود. قابل ذکر است منظور از NFA بدون λ ، تبدیل‌یافته NFA با λ به NFA بدون λ توسط توابعی است که در زیر توضیح خواهیم داد. در واقع ما در ابتدا NFA با λ را به NFA بدون λ تبدیل میکنیم.



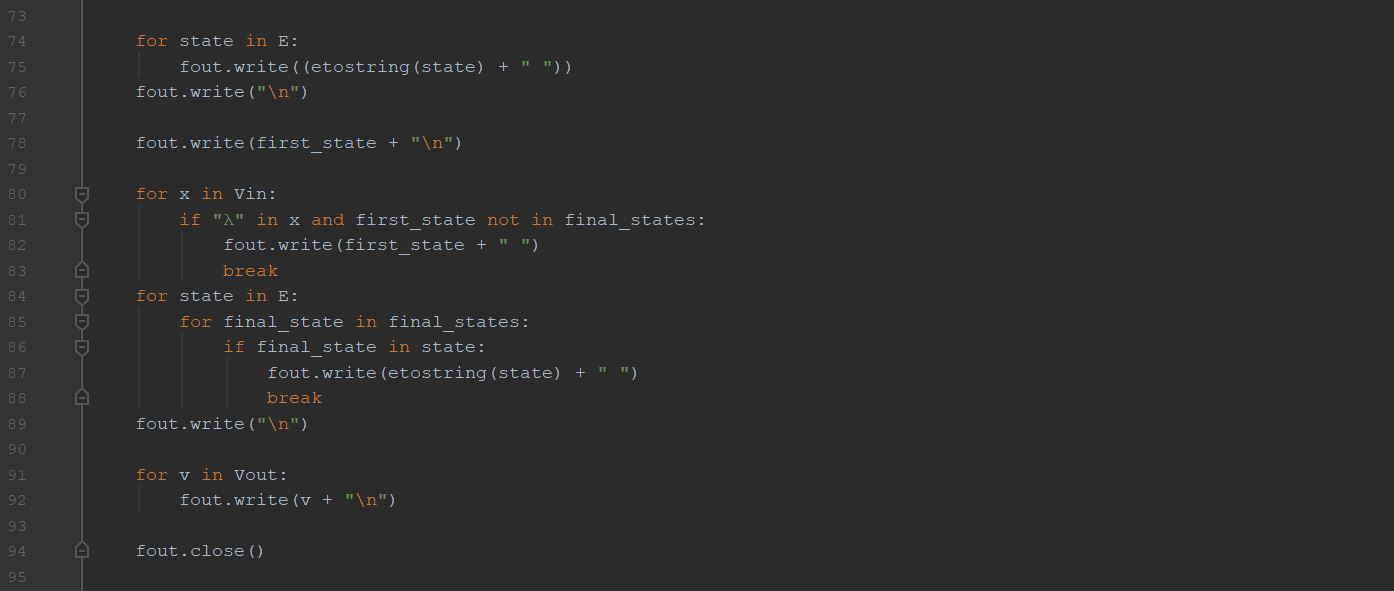
در تابع next\_states\_without\_l ، در واقع تابع 'δ برای بدست آوردن NFA بدون λ از طریق فرمول زیر پیاده‌سازی شده است:

در اینجا به دو تابع next\_states\_with\_l و l\_closure احتیاج داریم که زیر مشاهده می‌شود:

در تابع اول، مشخصاً استیت بعدی با توجه به کاراکتر ورودی و استیت فعلی، از روی NFA داده شده در فایل که ممکن است شامل λ هم باشد، مشخص می‌شود. در تابع دوم، در واقع \_closure λ یک حالت با محاسبه حالت‌هایی که تنها با λ می‌توان به آن‌ها رسید با استفاده از تابع اول و توسط یک آرایه برگردانده می‌شود.



در آخر نیز استیت های بدست آمده، حالت‌های نهایی و یال‌ها در فایل DFA خروجی نوشته می‌شود. حالت‌های نهایی DFA هم با چک کردن وجود حداقل یکی از حالات نهایی NFA در تمام حالات DFA بدست می‌آید و اگر NFA اولیه λ را هم بپذیرد، حالت اولیه نیز به عنوان یکی از حالات نهایی پذیرفته می‌شود.



پایان