## مسئله‌ی 1. نوشتن عبارات منظم

پاسخ.

۱. برای حل این قسمت باید حالت‌بندی بکنیم. به این صورت که قسمت عددی نمایش علمی، یا بزرگ‌تر مساوی ۱ و کوچک‌تر از ۵ است و یا بزرگ‌تر مساوی ۵ و کوچک‌تر از ۱۰ است.
اگر حالت اول رخ دهد، توان باید عددی صحیح و کم‌تر از ۲ باشد. اگر حالت دوم رخ دهد، توان باید عدد صحیحی نامثبت باشد.
حال شروع به بدست آوردن regex برای حالت اول می‌کنیم. قسمت عددی می‌تواند مثبت یا منفی باشد و ممکن است ممیز هم داشته باشد، اما با توجه به محدوده حالت اول، قسمت صحیح آن برابر با ۱ تا ۴ است.
پس regex آن به صورت زیر بدست می‌آید:

```
[-+]?[1-4](\.[0-9]+)?
```

اگر علامت توان مثبت باشد، مقدار عددی آن برابر با ۰ یا ۱ است. اگر منفی باشد می‌تواند هرمقداری داشته باشد. پس با توجه به این موضوع regex توان به صورت زیر بدست می‌آید:

```
[eE](([-+]?[0-1])|(-[0-9]+))
```

پس در کل regex برای حالت اول بدون در نظر گرفتن space های اول و آخر آن، به صورت زیر بدست می‌آید:

```
([-+]?[1-4](\.[0-9]+)?([eE](([-+]?[0-1])|(-[0-9]+)))?
```

به طور مشابه regex را برای حالت دوم نیز بدست می‌آوریم. دقت شود که در این حالت، در قسمت عددی نمایش علمی، قسمت صحیح مقداری از ۵ تا ۹ است و توان نمی‌تواند مثبت باشد. پس regex می‌شود:

```
[-+]?[5-9](\.[0-9]+)?([eE](([-+]?0)|(-[0-9]+)))?)
```

حال برای بدست آوردن regex کلی کافی است تا بین این دو یک or قرار دهیم و البته space ها را هم در ابتدا و انتها چک کنیم (به علت طولانی شدن در دو خط نمایش داده شده است):

```
 *(([-+]?[1-4](\.[0-9]+)?([eE](([-+]?[0-1])|(-[0-9]+)))?)
|([-+]?[5-9](\.[0-9]+)?([eE](([-+]?0)|(-[0-9]+)))?))  *
```

۲. جواب به صورت زیر می‌شود. دقت شود که \1 به این معناست که باید به جای آن دقیقا همان عبارت group اول در regex تکرار شود که یعنی عبارت داخل پرانتز:

```
021([0-9])\1[0-9]{6}
```

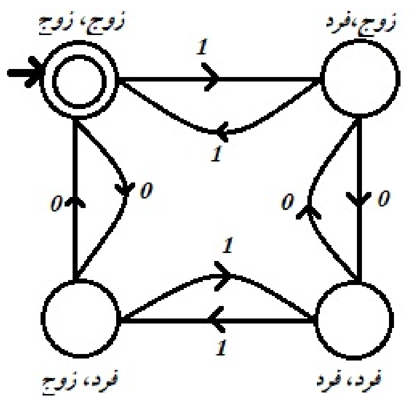
۳. جواب به صورت زیر می‌شود:

```
[a-z]+\.[0-9]{3}\.[A-Z]\.([0-9])\1
```

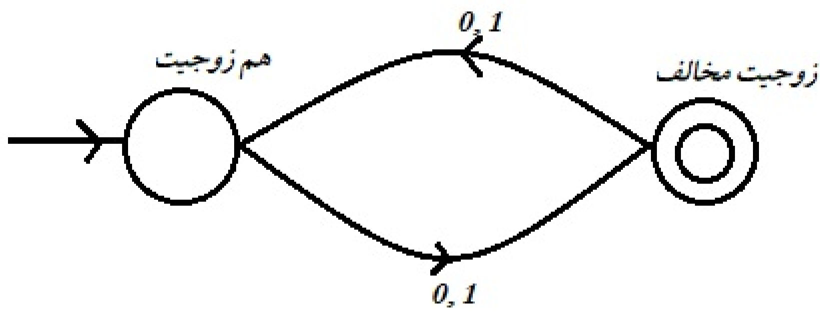
## مسئله‌ی 2.  طراحی DFA

پاسخ.

۱. در این DFA ما چهار راس داریم که هر راس نشان‌دهنده‌ی یک state است که زوجیت تعداد یک‌ها و صفرها را مشخص می‌کند. در هر راس با توجه به یالی که می‌آید زوجیت یکی از حرف‌ها تغییر کرده و به راس متناظر با وضعیت جدید یال می‌گذاریم. در نهایت حالتی که تعداد صفرها و یک‌ها زوج است حالت مورد قبول (accept) است و حالت شروع هم همین راس است چرا که در ابتدا رشته صفر تا ۰ و صفر تا ۱ دارد که تعداد هر دو زوج است.

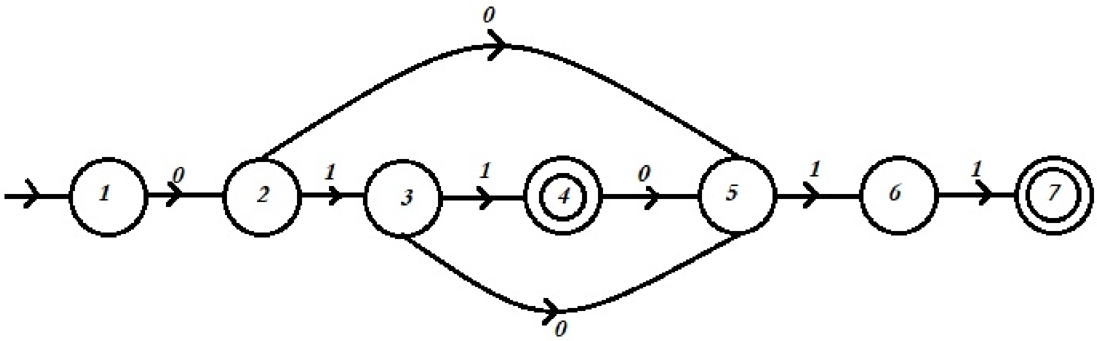



۲. برای این سوال دو وضعیت در نظر گرفته و به ازای هر وضعیت یک راس می‌گذاریم. حالت اول این است که زوجیت تعداد صفر و یک‌ها برابر باشند و حالت بعدی این است که زوجیت آن‌ها متفاوت باشد. در هرکدام از این وضعیت‌ها که باشیم هر حرفی بیاید به وضعیت مقابل می‌رویم چرا که اگر زوجیت صفر و یک‌ها یکسان باشد با اضافه‌شدن یک حرف جدید زوجیت یکی از آن‌ها تغییر می‌کند و دیگر یکسان نخواهد بود و اگر زوجیت آن‌ها متفاوت باشد با اضافه‌شدن یک حرف زوجیت آن حرف تغییر کرده و در نتیجه هم‌زوجیت خواهند شد. حالت accept هم برای زمانی‌است که زوجیت تعداد صفر و یک‌ها متفاوت باشد. همچنین حالت شروع حالتی است که زوجیت صفر و یک‌ها برابر است چون در ابتدا از هرکدام صفرتا داریم.




۳. در این DFA نباید دور جهت‌دار داشته باشیم چراکه درصورت وجود دور جهت‌دار، تعداد رشته‌هایی که DFA می‌پذیرد نامتناهی خواهد بود. چون می‌توان از راس آغاز شروع کرد و به یکی از رئوس دور رسید و سپس در دور به هر تعداد که می‌خواهیم بچرخیم و سپس به سمت راس accept برویم (اگر هیچکدام از رئوس دور به رئوس accept راه نداشته باشند، بود و نبود آن‌ها فرقی نمی‌کند و می‌توان آن‌ها را از DFA حذف کرد). بنابراین اگر DFA ما بخواهد رشته‌ی ۰۱۱۰۱۱ را بپذیرد و دور هم نداشته باشیم حداقل باید ۷ راس

داشته باشد (راس شروع بعلاوه‌ی ۶ راس دیگر که با آمدن هر حرف از این رشته به راس بعدی می‌رویم.) چون فرض کنید یک پیشوند از این رشته تا به اینجا خوانده شده و در یک راس از DFA هستیم. با خواندن حرف بعدی این رشته باید به یک راس جدید برویم (اگر به رئوسی که تا اینجا دیده‌ایم برویم دور خواهیم داشت). پس با شروع از راس آغاز و خواندن این رشته ۶ راس جدید خواهیم دید و در مجموع حداقل ۷ راس خواهیم داشت. حال مثالی ارائه می‌کنیم که ۷ راسی است و همه‌ی رشته‌های گفته شده را می‌پذیرد و رشته‌ی دیگری را نمی‌پذیرد:




در این مثال مشخص است که هر ۴ رشته‌ی گفته شده به یک راس accept می‌رسند. هیچ رشته‌ی دیگری هم در این DFA پذیرفته نمی‌شود چراکه مسیرهایی که از راس شروع به راس ۷ می‌رسند ۳ مسیر (۱،۲،۳،۴،۵،۶،۷ و ۱،۲،۵،۶،۷ و ۱،۲،۳،۵،۶،۷) و تنها مسیر از راس شروع به راس ۴، که دیگر راس accept است مسیر ۱،۲،۳،۴ است.

## مسئله‌ی 3. **NFA** معادل

پاسخ.
در هر قسمت اگر مقداری با pattern داده شده مطابقت داشت، آن را انتخاب کرده و در صورتی که حالت بعدی که برای آن در نظر میگیریم طول رشته بیشتری را شامل بشود، حالت در نظر گرفته را آپدیت میکنیم. همچنین میدانیم که عملگر + به معنای تعداد ۱ یا بیشتر از آن کارکتر است و عملگر | به معنای کافی بودن وجود یکی از کارکترها میباشد.

۱. رشته‌ی aaabccabbb
در این حالت aaab نسبت به حالات قبلی دارای بیشترین طول است پس باحالت سوم مطابقت داده میشود و جایگزین حالت قبلی میگردد. سپس دو تا کارکتر c داریم که با ۴ مطابقت داده میشوند و بعد نیز abbb را با حالت سوم تطبیق میدهیم. در نهایت رشته خروجی برابر ۲۴۴۲ خواهد بود.

۲. رشته‌ی cbbbbac
ابتدا به ازای کارکتر نخست ۴ را خروجی داده و سپس چهار کارکتر بعدی با حالت سوم مطابقت پیدا میکنند. این چهار کارکتر با حالت چهارم نیز مطابق هستند اما چون افزایش طولی نداریم، پس مقدار جدیدی به خود نمیگیرند. سپس دو کارکتر انتهایی هم به وضوح با حالت دوم و پنجم مطابق شده و خروجی نهایی برابر ۴۲۱۴ خواهد بود.

۳. رشته‌ی cbabc
در ابتدا و انتها که برای کارکتر c ۴ خروجی میدهیم و برای کارکترهای میانی نیز بیشترین طول برابر ۳ خواهد بود که برای این حالت باید ۲ را خروجی داد. پس در نهایت جواب برابر ۴۲۴ خواهد شد.

## مسئله‌ی 4. NFA

پاسخ.

۱. فرض کنید که اتوماتای متناظر با عبارت منظم $R$ برابر با $G$ باشد. استیت آغازین این گراف را $G[start]$ و استیت اکسپت آن را $G[end]$ می‌نامیم. $m$ کپی از گراف $G$ ایجاد می‌کنیم و آنها را $G_1, G_2, ..., G_m$ می‌نامیم. می‌دانیم هر یک از این گراف‌ها، دقیقا یک استیت شروع دارند و یک استیت اکسپت. حال می‌خواهیم یک گراف $G'$ متشکل از این گراف‌ها بسازیم.

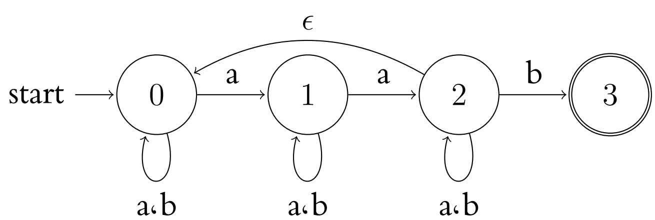
برای این کار، به ازای هر $i < m$ یک یال جهت دار با کاراکتر $\varepsilon$ از راس اکسپت گراف $G_i$ به راس شروع گراف $G_{i+1}$ می‌گذاریم. از آنجا که در $G'$ تنها باید یک استیت شروع داشته باشیم، استیت شروع، $G_1[start]$ خواهد بود(پس سایر استیت هایی که قبلا استیت شروع بودند را به یک استیت عادی تبدیل می‌کنیم). هم‌چنین در گراف $G'$، استیت‌های اکسپت برابر است با استیت‌های اکسپت گراف‌های $G_n, G_{n+1}, ..., G_m$. پس استیت‌های accept گراف‌های $G_1, G_2, ..., G_{n-1}$ را به استیت‌های عادی تبدیل می‌کنیم. ادعا می‌کنیم گراف ساخته شده، $NFA$ متناظر با $R\{n, m\}$ است. زیرا اگر یک رشته بخواهد که پذیرفته شود، در استیت اکسپت یکی از گراف‌های $G_i$ که $n \leqslant i \leqslant m$ اکسپت می‌شود. برای این اتفاق، مسیری که یک رشته طی می‌کند تا به این استیت اکسپت برسد، مسیری است که از $G_1, G_2, ..., G_{i-1}$ عبور کرده است. مسیری که در هر یک از این گراف ها طی شده، مسیری متناظر یک رشته در زبان $R$ است. زیرا از $G_{index}[start]$ شروع شده، تعدادی یال طی شده، به استیت $G_{index}[end]$ آن گراف رسیده، و بعد با یک یال $\varepsilon$ به استیت $G_{index+1}[start]$ رفته است. پس مسیری که در یک گراف طی شده، یک رشته است که در $G_{index} = G$ پذیرفته می‌شود. به طریق مشابه، می‌توان گفت مسیری که در $G_i$ طی شده مربوط به یک رشته در $G$ است.

پس رشته پذیرفته مربوط به $R\{i\}$ است. با توجه به این که $n \leqslant i \leqslant m$ ، رشته‌ی اکسپت شده داخل زبان $R\{n, m\}$ است. هم‌چنین واضح است که هر رشته مربوط به $R\{n, m\}$ نیز اکسپت می‌شود. برای مثال اگر $W \in R\{t\}$ که $n \leqslant t \leqslant m$ ، می‌دانیم $W \in RR...R$ که تعداد $R$ ها برابر $t$ است. پس کافی است در $G_i$ که $1 \leqslant i \leqslant t$ مسیری را طی کنیم که متناظر با رشته ای است که از $i$ امین $R$ انتخاب شده است.

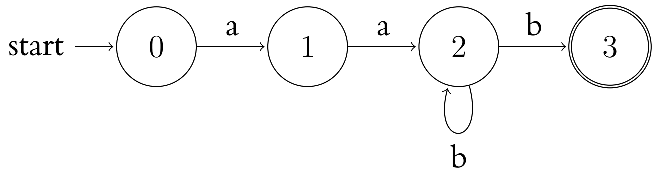
پس $G'$ همان $NFA$ است که به دنبال آن بودیم.

۲. در کل ۶ مسیر aabb را مشخص میکنند: (همواره در استیت صفر هستیم و از بعد آن استیت ها را به ترتیب ذکر کرده شده‌اند.)
۱۲۲۳
۱۱۰۰
۰۱۱۱
۰۱۱۰
۰۱۱۲
۰۱۲۲
که از بین آنها مسیر ۱۲۲۳ در استیت نهایی رشته را تمام میکند و لذا رشته قابل تشخیص است.

حالت کلی :



نمونه‌ی یک مسیر قابل قبول :

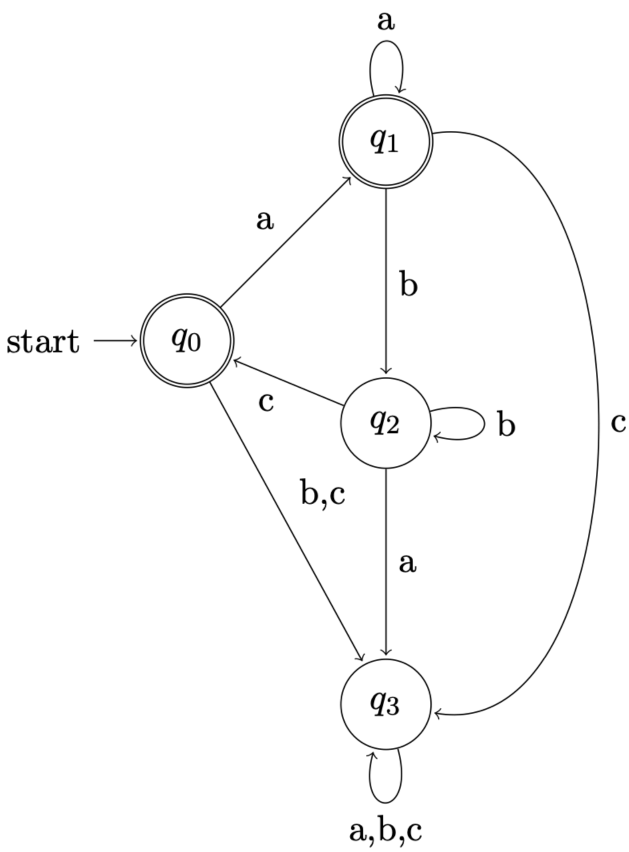



۳. ادعا می کنیم که بی نهایت مسیر وجود دارد . برای این کار بی نهایت مسیر مثال می زنیم که به استیت نهایی می رسد. ابتدا مسیر زیر را طی می کنیم: $0 \to 1 \to 0 \to 1 \to 2 \to 3$ تا اینجا $aabb$ ایجاد شده است . مسیر زیر را در نظر بگیرید به طوری که تمام یال های طی شده اپسیلون باشند: $3 \to 2 \to 1 \to 0 \to 3$ این مسیر دوباره ما را به استیتی که بودیم یعنی استیت 3 برمی گرداند و چون یال های اپسیلون را طی کردیم پس $aabb$ تغییر نمی کند. اسم این مسیر را مسیر خوب می گذاریم. حالا با استفاده از مسیری که در ابتدای پاسخ گفتیم به استیت 3 می رویم ، الان می توانیم به هر تعدادی از مسیر خوب استفاده کنیم و در نهایت دوباره در استیت 3 باشیم پس با آوردن بی نهایت مثال نشان دادیم که بی نهایت مسیر وجود دارد.

## مسئله‌ی 5. NFA به DFA

پاسخ.

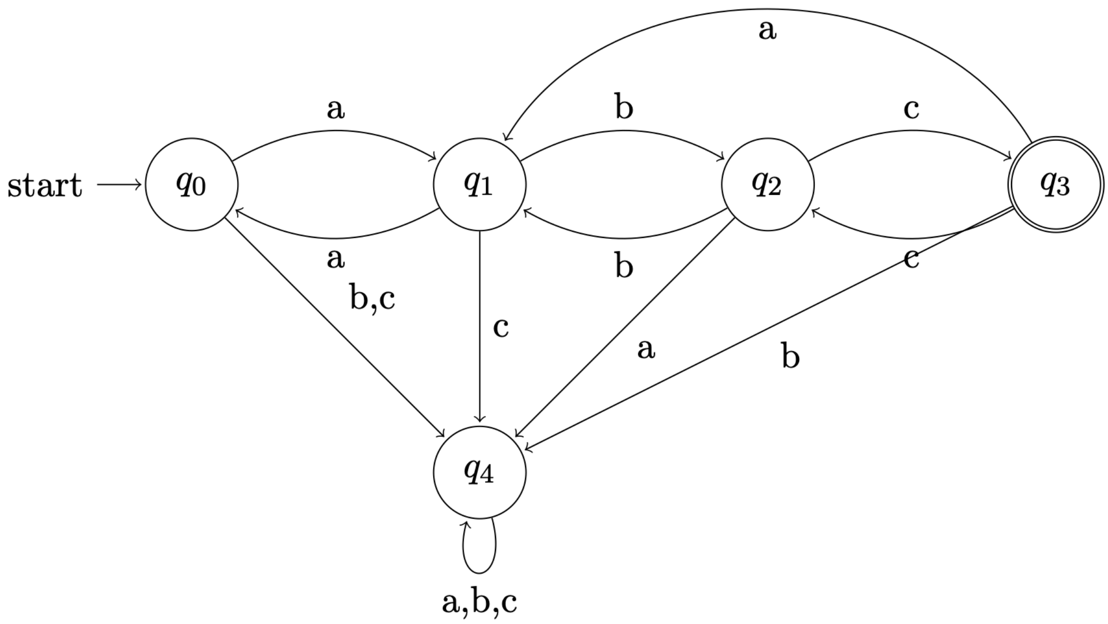
1.




Corespondences (DFA to NFA):

- $q_0$ to $\{q_0\}$
- $q_1$ to $\{q_0, q_1\}$
- $q_2$ to $\{q_2\}$
- $q_3$ to $\{\}$

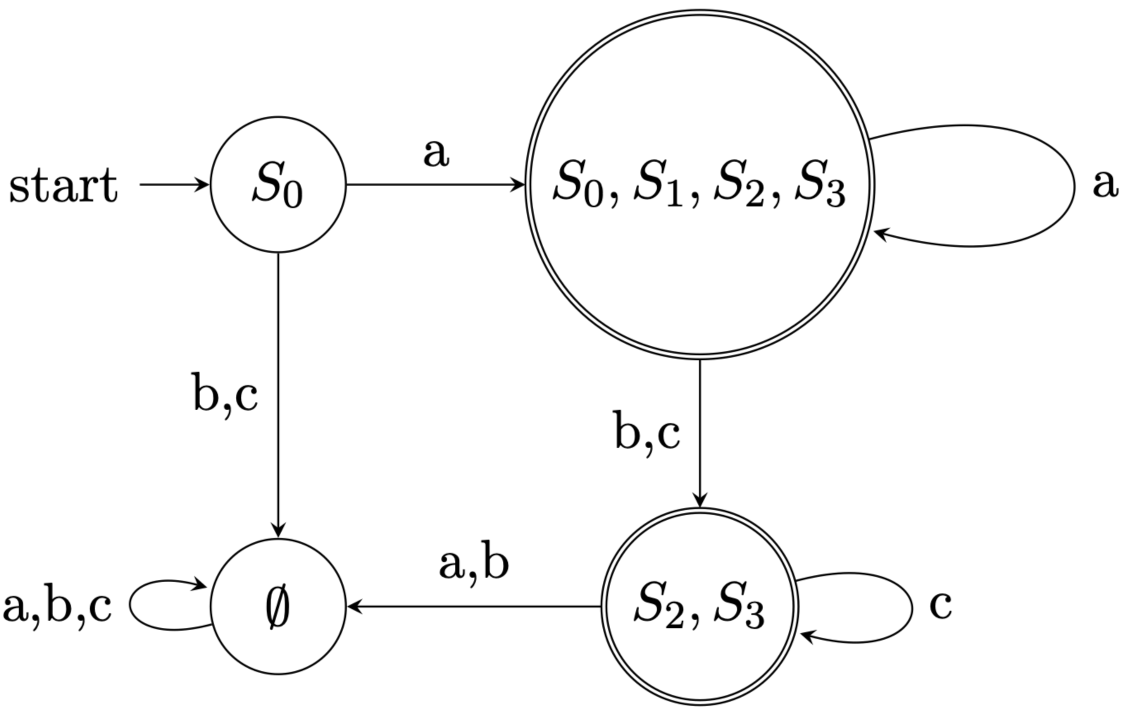
2.



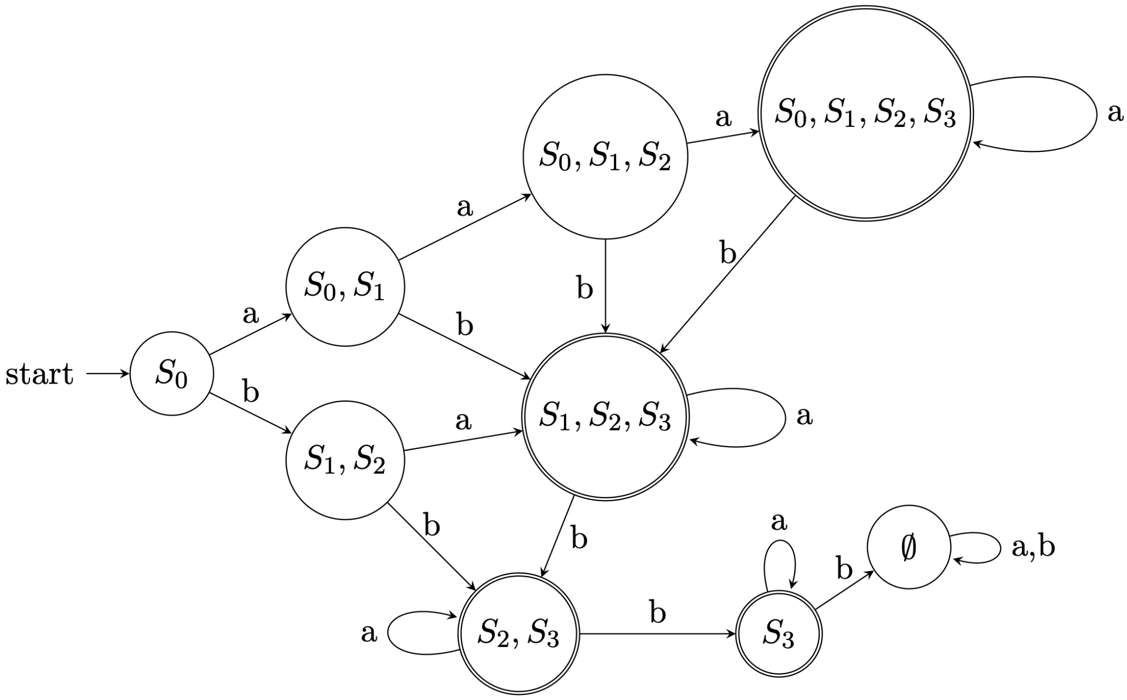

Corespondences (DFA to NFA):

- $q_0$ to $\{q_0\}$
- $q_1$ to $\{q_1\}$
- $q_2$ to $\{q_2\}$
- $q_3$ to $\{q_0, q_3\}$
- $q_4$ to $\{\}$

3.




4.




## مسئله‌ی 6. طراحی واژه‌یاب

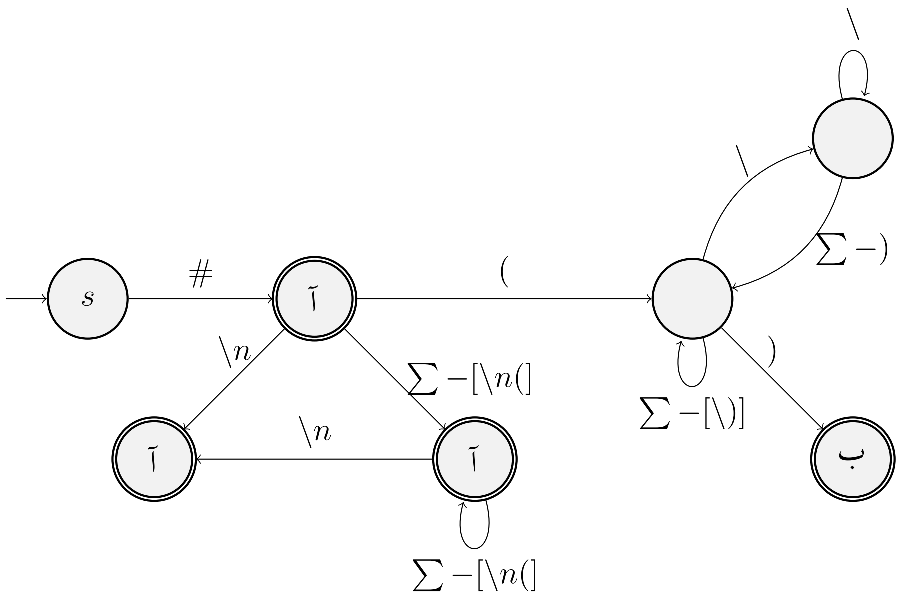
پاسخ.

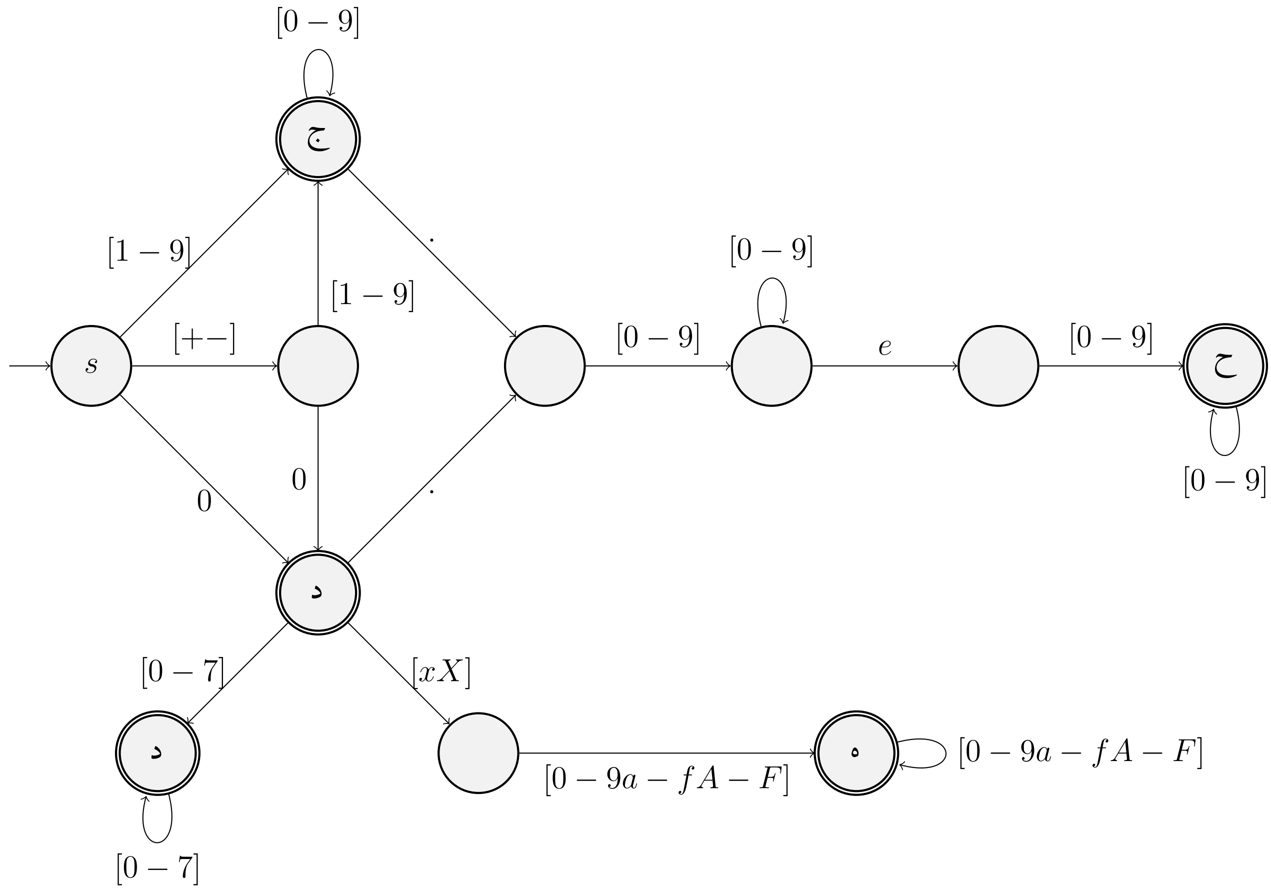
۱ـ اگر در یک حالت بودیم و کاراکتری دریافت کردیم که یال گذار مربوط به آن رسم نشده بود، توکن وارد شده قابل تشخیص نیست(یعنی جزو هیچ کدام از ۸ دسته ی گفته شده نیست) و عملا باید به یک حالت error برویم(یا اینکه به حالت start برویم تا ادامه کاراکتر ها را اسکن کنیم) اما این یال ها نیز برای سادگی شکل رسم نشده اند. DFA به سه بخش تقسیم شده است. اگر سه راس start هر سه بخش را ادغام کنیم، DFA نهایی به دست می آید.

برای نشان دادن الفبای حاصل از اجتماع چند کاراکتر، آن چند کاراکتر را درون [] گذاشته ایم.

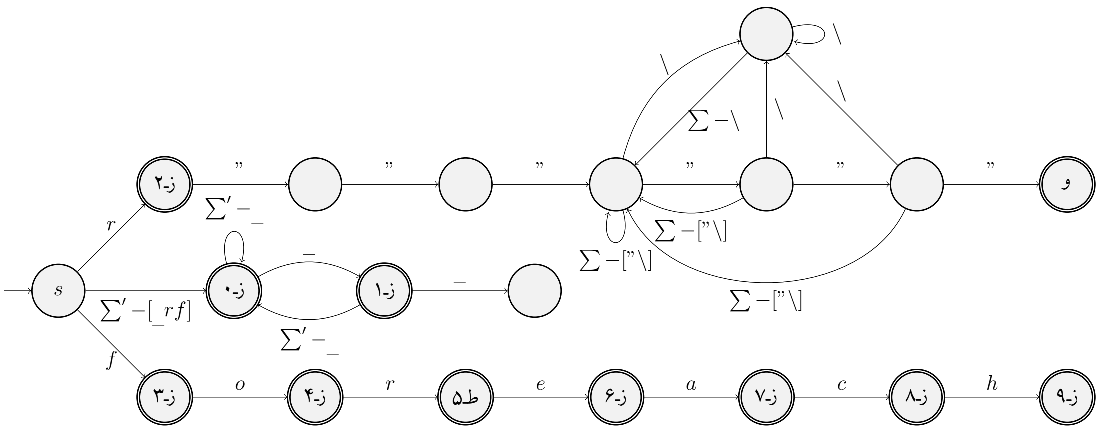
دسته‌ي توکن شناسایی شده درون حالت های قبول نوشته شده است.

$$\sum' = [a - zA - Z0 - 9\_]$$




در این قسمت از DFA برای ساده سازی شکل تعدادی از یال ها که توصیفشان در ادامه می آید رسم نشده اند. بعضی از حالات در اینجا دارای یک عدد نیز هستند که با _ از نام دسته‌ی توکن شناسایی شده جدا شده اند. تمام توکن های ز_۲ تا ز_۹ یک یال با ورودی _ به ز_۱ دارند. تمام توکن های ز_۲ تا ز_۹ یک یال با ورودی $\sum -x'$ به ز_۰ دارند که x برابر با ورودی های یال های دیگری است که از این راس ها خارج می شوند.



**۲ـ**

```
%option noyywrap

%{
enum {T_DECIMAL=1, T_OCTAL, T_HEX};
long yylval;
%}

WHITESPACE      [[:space:]]+
DECIMAL         ([+-]?[1-9][0-9]*)
OCTAL           ([+-]?0[0-7]*)
HEX             ([+-]?0[Xx][0-9A-Fa-f]+)

%%

{WHITESPACE}    {;}

{DECIMAL}       {yylval = abs(strtol(yytext, NULL,10));
                printf(%d "T_DECIMAL\n", yylval);
                return T_DECIMAL;}

{OCTAL}         {yylval = abs(strtol(yytext, NULL,8));
                printf(%d "T_OCTAL\n", yylval);
                return T_DECIMAL;}

{HEX}           {yylval = abs(strtol(yytext, NULL,16));
                printf(%d "T_HEX\n", yylval);
                return T_DECIMAL;}

.               {printf("error:%s", yytext);
                exit(0);}


%%
```

```c
int main(){
    int token;
    while(token = yylex());
}
```

مسئله‌ی 7.  **NFA به DFA**

پاسخ.