بنام خدا

شبیه‌سازی NFA (الگوریتم پذیرش رشته به طول k از طریق NFA با n حالت و m transition)

S_0 حالت شروع NFA

F مجموعه حالات پایانی NFA

```
S = ε-closure(S0)
c = nextchar();                 // حرف بعدی از ورودی
while (c != eof) {              // → (k) (رشته) تعداد حروف ورودی
    S = ε-closure(move(S,c));   // حداکثر کل state و trans های NFA را باید پیمایش شود
    c = nextchar()
}
if (S ∩ F != ∅) return accept   // O(m+n)
else reject;
```

پیچیدگی زمانی پذیرش NFA برای رشته به طول k، $O(k(m+n))$ است

باتوجه به الگوریتم تولید NFA از روی RE به طول r، NFA بدست آمده حداکثر 2r حالت و 4r trans دارد

$O(k(m+n))$, $n \leq 2r$, $m \leq 4r$ $\Rightarrow$ پیچیدگی زمانی $= O(k \times 6r) = O(kr)$

	initialize ساخت	پذیرش رشته به طول k	حافظه
NFA	$O(r)$	$O(kr)$	$O(r)$
DFA worst case	$O(r^2 2^r)$	$O(k)$	$O(2^r)$
DFA typical case	$O(r^3)$	$O(k)$	$O(r^2)$

الگوریتم کاهش تعداد حالات DFA

۰- اگر DFA کامل نیست با اضافه کردن state مرده آن را کامل کن

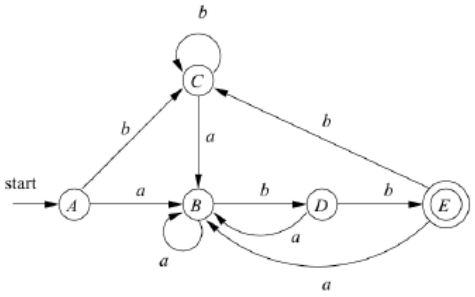
۱- state های DFA را به دو گروه ac و na ac تقسیم کن

۲- در داخل هر گروه state هایی که با ورودی یکسان به گروه‌های مختلفی منتقل می‌شوند را جدا می‌کنیم و آنها را در گروه جدیدی تشکیل دهید

۳- مرحله ۲ را آنقدر ادامه دهید تا هیچ گروه جدیدی تشکیل نشود.

۴- هر گروه نماینده یک state است. state های مرده و state های اضافی حذف می‌شوند

مثال)

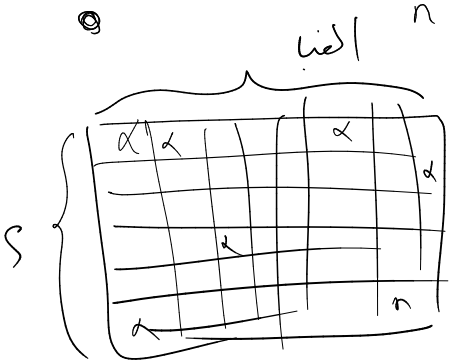


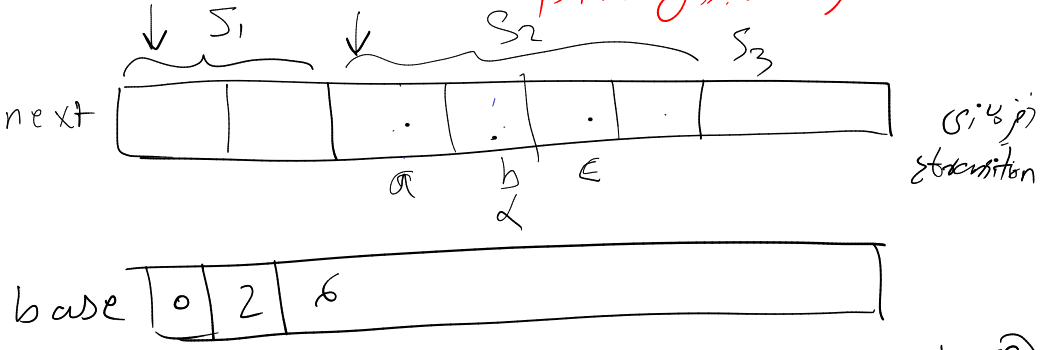
(A, B, C, D)(E)

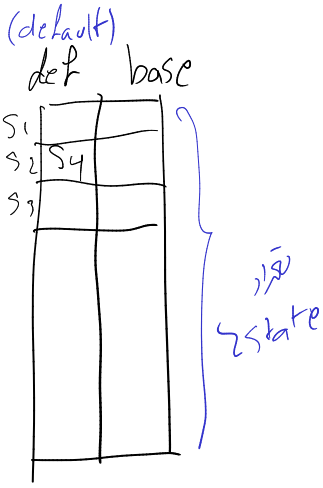
(A, B, C)(D)(E)

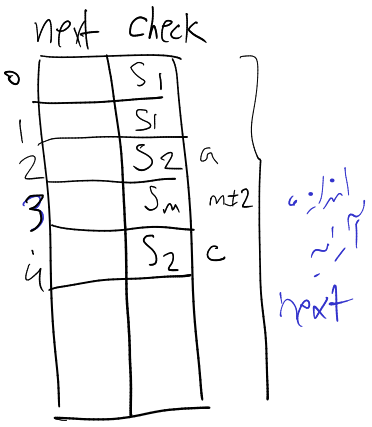
(A, C)(B)(D)(E)

روش فشرده سازی جدول DFA

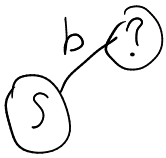


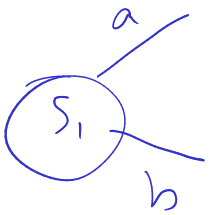


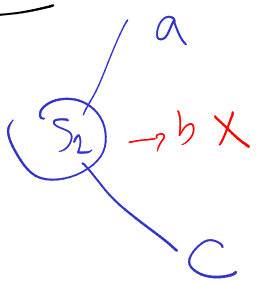




next [base [s] + b]







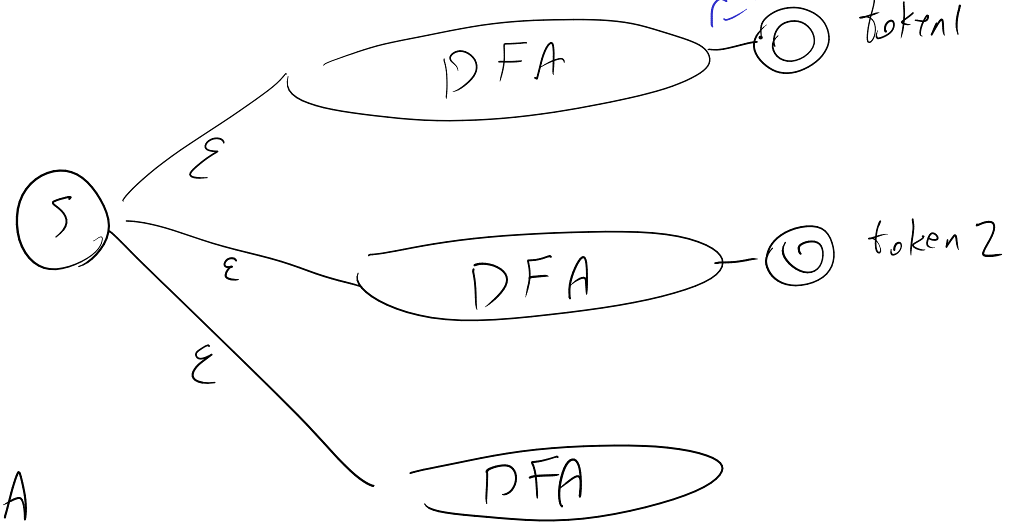
```
nextstate(s, a){
    if (check[base[s]+a]==s)
        return next[base[s]+a]
    else
        return next[def[s], a]
```

next: آرایه‌ای حاوی transitionهای stateهای مختلف

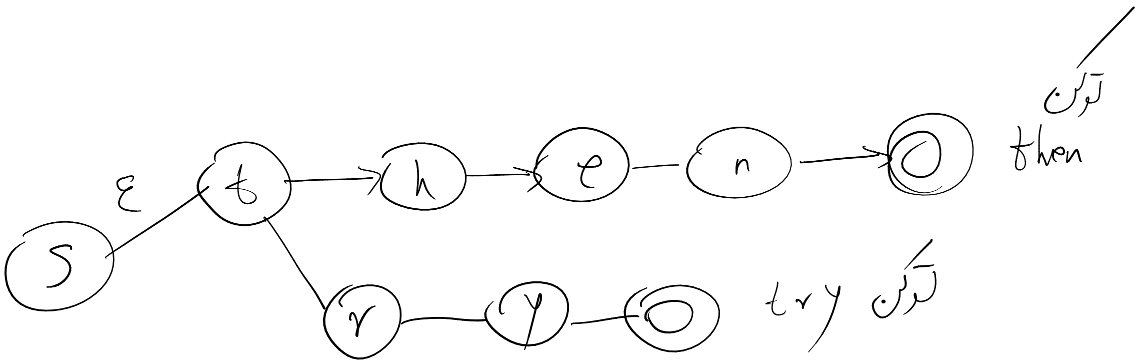
base: ایندکس ابتدای شروع transitionهای هر state را مشخص می‌کند زیرا در next دقیقاً مشخص نیست اطلاعات هر state از کجا شروع می‌شود

check: مشخص می‌کند اطلاعات next در هر خانه مربوط به کدام state است.

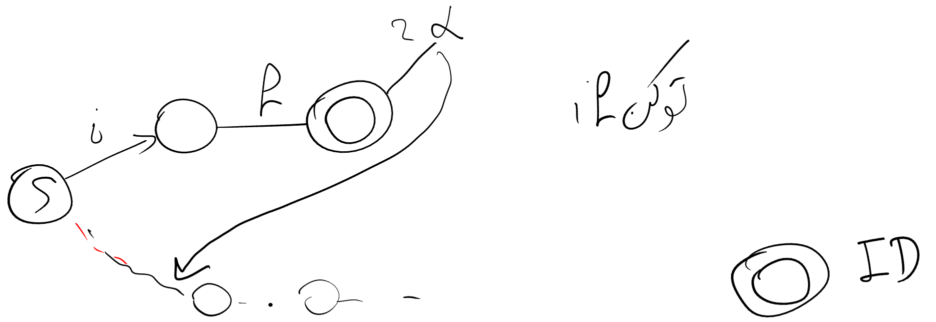
def: اگر در دسترسی مستقیم به next با بررسی check متوجه شدیم با حرفی که دیده‌ایم از state مورد نظر tran نداریم می‌توانیم به state مشخص شده در def مراجعه کنیم



NFA → DFA

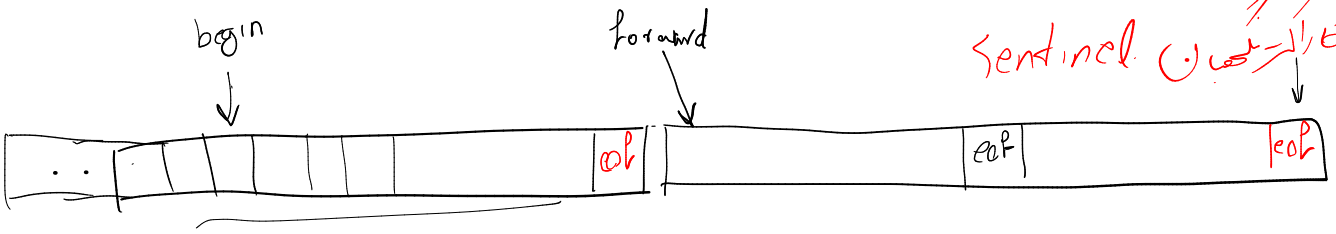


if2



اینجا اگر در state پایانی if با فاصله یک رقم یا حرف (a..z AZ) دیدیم می‌توانیم def را state شروع دنبال کردن توکن ID بذاریم.

بافر - خطا - lex (جلسه بعد)



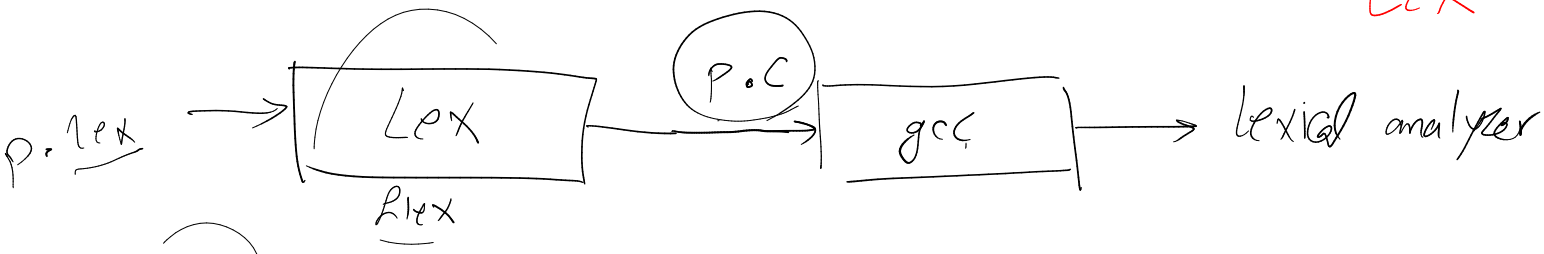
بافر و کاراکترهای Sentinel

```
switch ( *forward++ ) {
        case eof:
                if (forward is at end of first buffer ) {
                        reload second buffer;
                        forward = beginning of second buffer;
                }
                else if (forward is at end of second buffer ) {
                        reload first buffer;
                        forward = beginning of first buffer;
                }
                else /* eof within a buffer marks the end of input */
                        terminate lexical analysis;
                break;
        Cases for the other characters
}
```

خطا (panic mode)

ignore کردن یا تغییر دادن یک یا تعداد بیشتر کاراکتر ورودی (با مرتب جابجا کردن کاراکترها) با هدف گذر از خطا و امکان تشخیص توکن‌های بعدی

Lex



R1
R2
{
}
R3