TÖL403G GREINING REIKNIRITA

16. Strengleit 2

Hjálmtýr Hafsteinsson
Vor 2022

Í þessum fyrirlestri

- Stöðuvélar (*finite state machines*)
  - Notkun í strengleit
- Knuth-Morris-Pratt reikniritið
  - Útreikningur á ***fail***-fylki
  - Bestað ***fail***-fylki

DC 7.5 − 7.7

Knuth-Morris-Pratt reikniritið

- Uppgötvað sjálfstætt á svipuðum tíma af James Morris og Donald Knuth [1970]
  - Morris var starfsmaður hjá Xerox PARC rannsóknarstofnuninni að útfæra textaritil fyrir fyrstu grafísku vinnustöðvarnar
  - Vaughan Pratt var doktorsnemi við Stanford og síðar prófessor þar

Reikniritið var ekki formlega birt í tímariti fyrr en 1977 undir nöfnum allra þriggja

- KMP-reikniritið vinnur í tveimur þrepum:
  - Búa til stöðuvél úr mynstrinu $P[1..m]$ á $O(m)$ tíma
  - Keyra textann $T[1..n]$ í gegnum stöðuvélina á $O(n)$ tíma

Heildartími: $O(n+m)$ ← Versta-tilfellis tímaflækja

Besti tími mögulegi tími (*optimal*)

Stöðuvélar

- Sáum í síðasta fyrirlestri að við þurfum ekki að bakka í textanum
  - Þurfum þá að nýta okkur það sem við höfum séð
  - og ekki gera óþarfa samanburði
- Við munum forvinna (*preprocess*) mynstrið *P* og búa til stöðuvél úr því
- Keyra svo textann *T* í gegnum stöðuvélina

Stöðuvél:

Hver staða (hnútur) er bókstafur úr mynstrinu

Höfum líka tvær aukastöður: $ sem er upphafsstaðan og ! sem er lokastaða

Hver staða hefur tvær færslur (þ.e. tvær örvar):

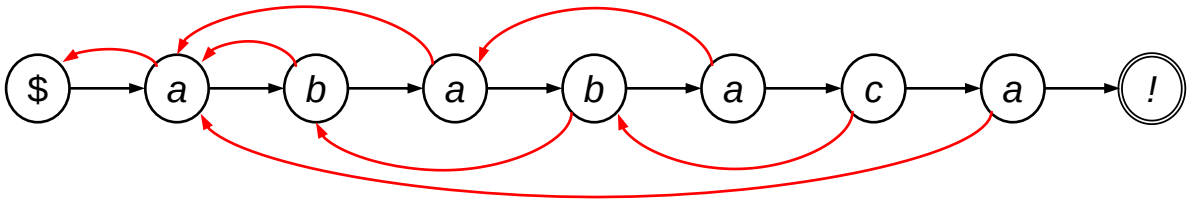
JÁ-færsla (*success*) Ef núverandi stafur í texta, *T*[*i*] er eins og stafurinn í stöðunni (*P*[*j*])

NEI-færsla (*failure*) Ef núverandi stafur í texta, *T*[*i*] er ekki eins og stafurinn í stöðunni (*P*[*j*])

Dæmi um stöðuvél

Mynstrið *P* er "ababaca"

Svartar örvar eru JÁ-færslur
Rauðar örvar eru NEI-færslur



Keyrum svo textann *T* í gegnum þessa stöðuvél

Förum áfram í *T* ef stafir passa, en stöndum í stað í *T* ef stafir passa ekki

Svört ör

Rauð ör

Athugið:

Við þurfum ekki að búa til alla stöðuvélina, heldur bara NEI-færslurnar (*fail*) fyrir hvern staf mynstursins

JÁ-færslurnar fara alltaf í næsta staf mynstursins!

Geymum NEI-færslurnar í fylkinu *fail*[1..*m*]

Knuth-Morris-Pratt reikniritið

Reikniritið sem framkvæmir leitina, eftir að við höfum búið til stöðuvélina (þ.e. *fail*-fylkið):

```
KnuthMorrisPratt(T[1..n], P[1..m]):
    j ← 1
    for i ← 1 to n
        while j > 0 and T[i] ≠ P[j]
            j ← fail[j]
        if j = m        ⟨⟨Found it!⟩⟩
            return i − m + 1
        j ← j + 1
    return None
```

j er staðsetning í mynstrinu *P*
i er staðsetning í textanum *T*

Byrjum í fyrsta staf mynsturs

Bakka í mynstrinu (þ.e. fara eftir rauðum örvum fram eftir stöðuvélinni)

Mynstrið fannst í textanum, skila staðsetningu þess

Á meðan stafirnir eru eins þá færum við báða benda áfram

Tími: Virðist við fyrstu sýn vera $O(nm)$, því ytri lykkja er keyrð n sinnum og innri lykkja mest m sinnum

Heildarfjöldi hækkana á *j* er því *n*

EN...

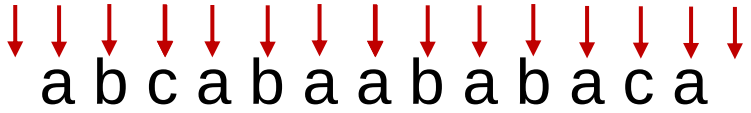
Ytri lykkjan er framkvæmd n sinnum og j er aðeins hækkað um 1 í hverri ítrun

while-lykkjan lækkar j, svo heildarfjöldi ítrana á henni (uppsafnað yfir allt reikniritið) getur mest verið n

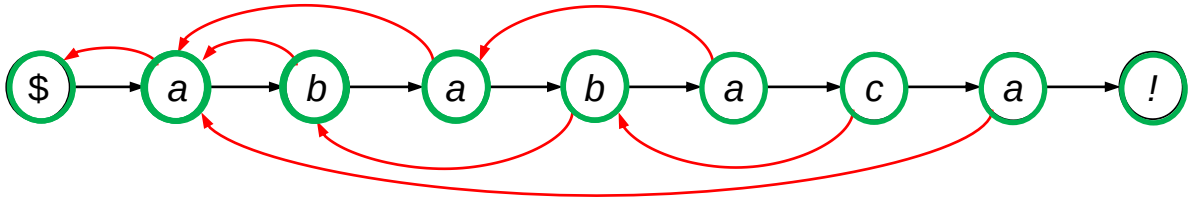
Heildartími: $O(n)$

Sýnidæmi

Textinn *T*: abcabaababaca

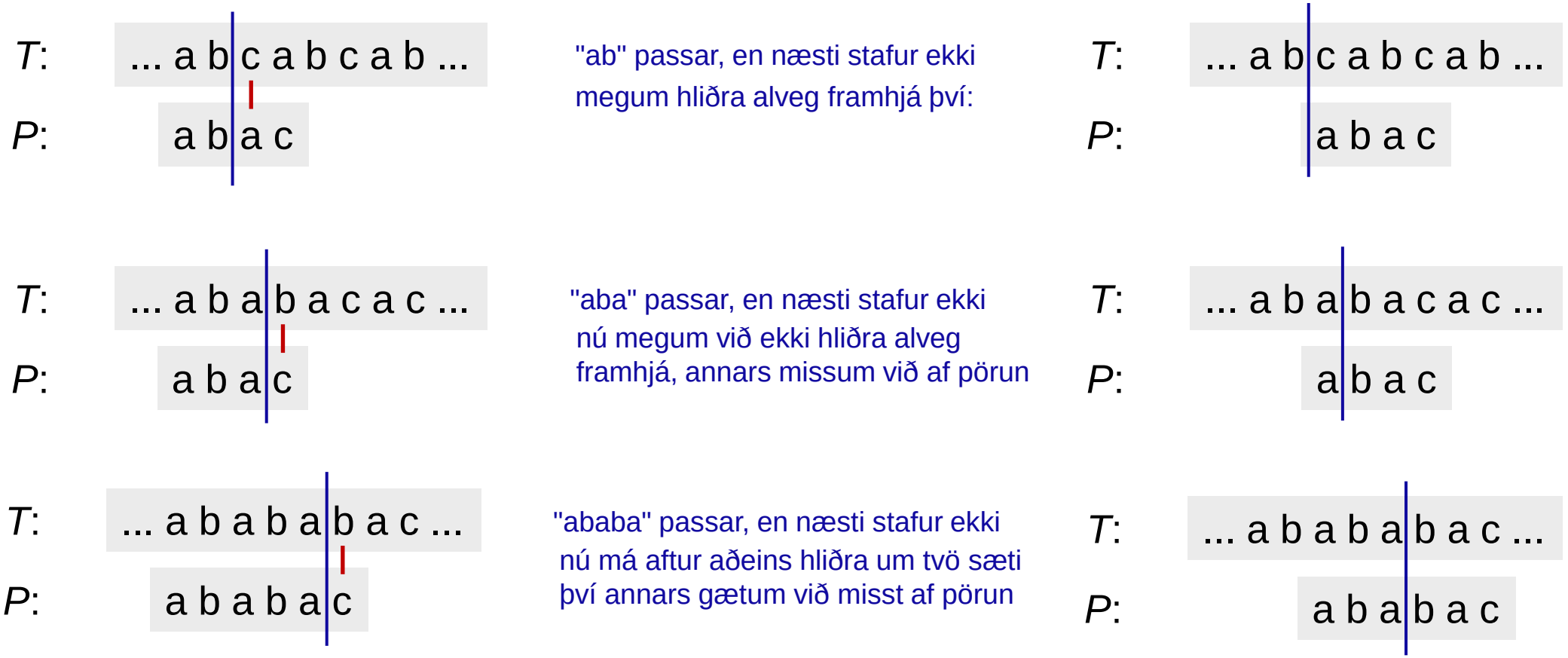


Stöðuvélin:



Hversu mikið megum við hliðra?

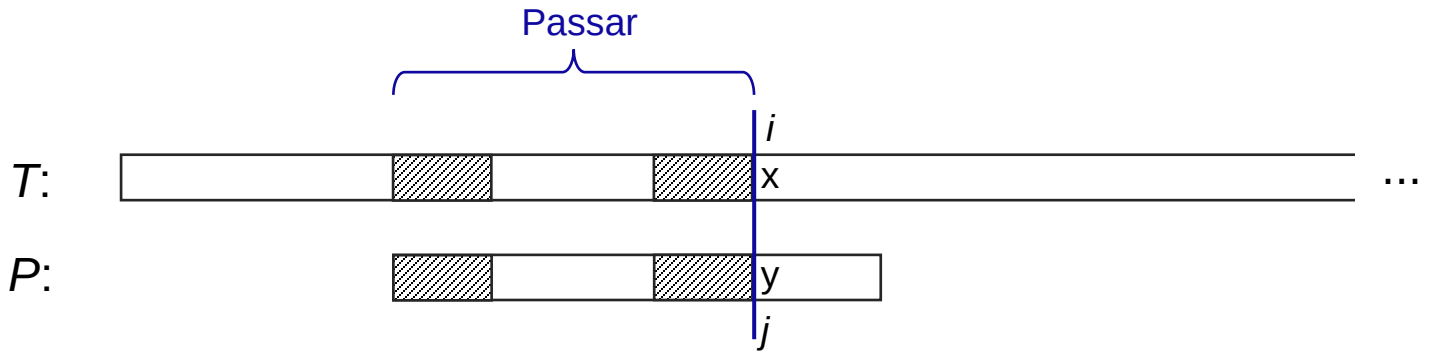
- Skoðum nokkur dæmi til að fá tilfinningu fyrir því hversu mikið við megum hliðra



fail-fylkið

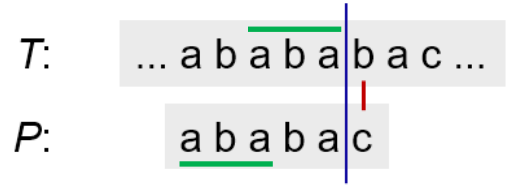
- *fail*-fylkið segir okkur hversu langt má hliðra

Höfum stöðuna:

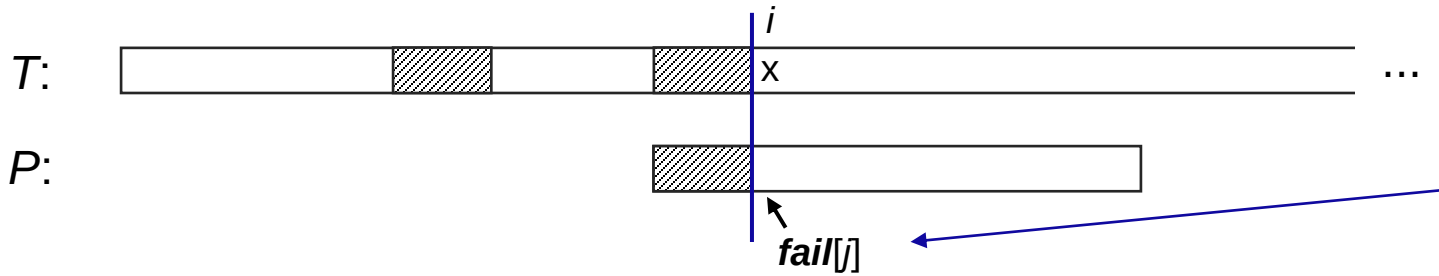


Finnum lengsta eiginlegan forstreng (*proper prefix*) á *P* sem passar við eftirstreng (*suffix*) á *T*[1..*i*−1]

t.d.



Hliðrum svo að þetta passi:



fail[*j*] segir okkur hversu langt eigi að hliðra ef pörun klikkar í staf *j*

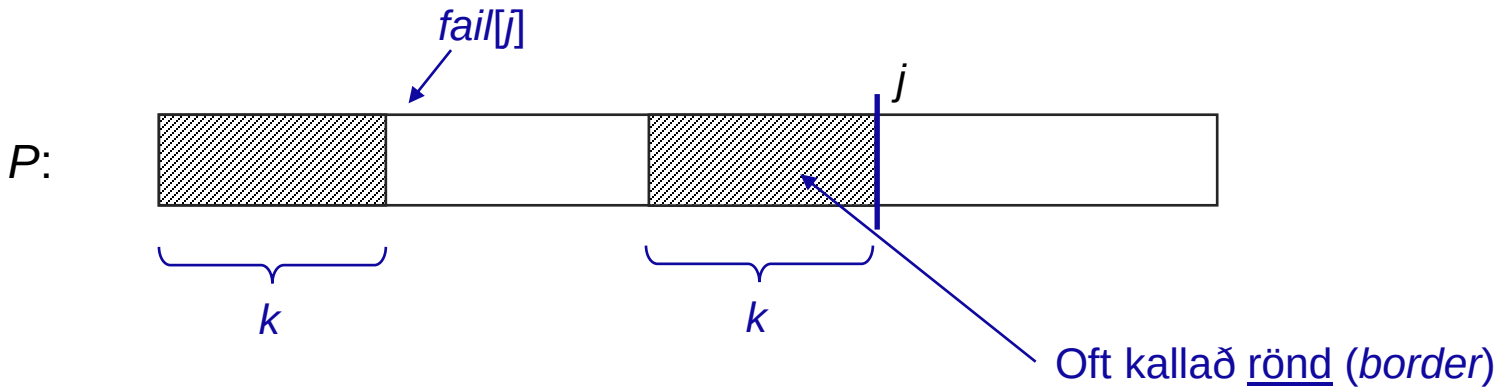
Útreikningur á *fail*

- Við vitum að $P[1..j-1]$ er eftirstrengur á $T[1..i-1]$, svo að það er nóg að skoða P

Skilgreining á *fail*:

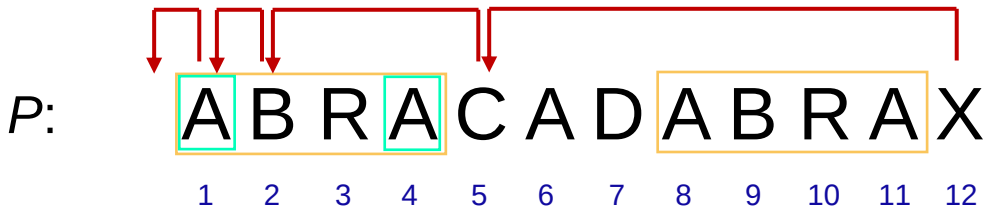
> $P[1..fail[j]-1]$ er lengsti eiginlegi forstrengur á $P[1..j-1]$, sem er líka eftirstrengur á $P[1..j-1]$

Myndrænt:



Sýnidæmi um *fail*

Dæmi:



Innri lykkja í KMP reikniriti:

while $j > 0$ and $T[i] \neq P[j]$
$j \leftarrow fail[j]$

Við erum að bera *T*[*i*] saman við *P*[12] og þeir passar ekki

Í reikniritinu er þá *j* sett sem *fail*[12]

Skoðum *fail*[12]: Röndin er "ABRA" svo *fail*[12] = 5

Berum þá *T*[*i*] saman við *P*[5] og ef þeir passar ekki

Þá *j* sett sem *fail*[5]

Skoðum *fail*[5]: Röndin er "A" svo *fail*[5] = 2

Ef það passar ekki þá er *j* sett sem *fail*[2]

Skoðum *fail*[2]: Röndin er tóm svo *fail*[2] = 1

Þarf að vera eiginlegur hlutstrengur

fail[1] er upphafsstillt sem 0

Þá dettum við út úr *while*-lykkju og við förum í næsta textastaf

Æfing

- Búið til *fail*-fylkið fyrir mynstrið "ABCAB"

P:

A	B	C	A	B
1	2	3	4	5

fail:

0				
1	2	3	4	5

- Teiknið stöðuvélina fyrir mynstrið

Útreikningur á *fail*-fylki

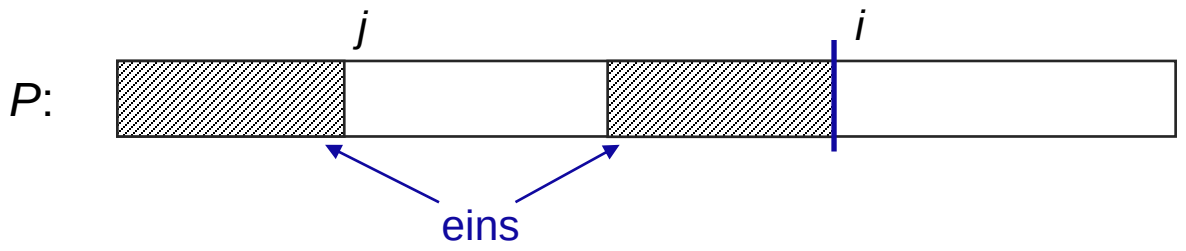
- Gætum reiknað hvert stak fyrir sig sjálfstætt
  - Það myndi taka $O(m^3)$ tíma
- Til betra reiknirit sem notar eins konar kvika bestun
  - Nýtir sér áður-útreiknuð gildi

```
ComputeFailure(P[1..m]):
    j ← 0
    for i ← 1 to m
        fail[i] ← j          (∗)
        while j > 0 and P[i] ≠ P[j]
            j ← fail[j]
        j ← j + 1
```

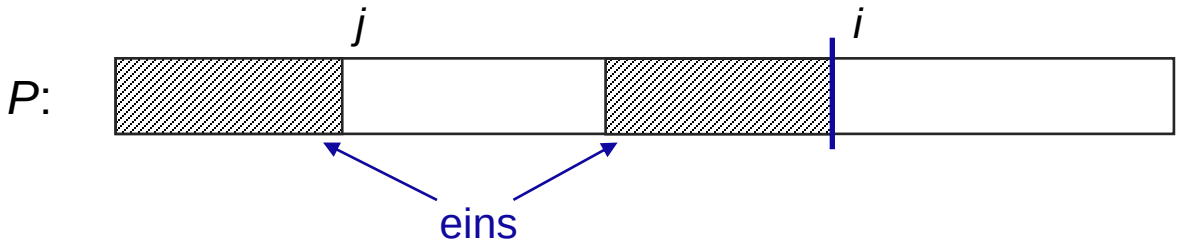
Vísirinn *i* er númerið á *fail*-hólfinu sem verið er að finna

Vísirinn *j* er innihaldið í núverandi *fail*-hólfi

Dæmigerð staða í reikniritinu:



Hegðun reikniritsins *ComputerFailure*



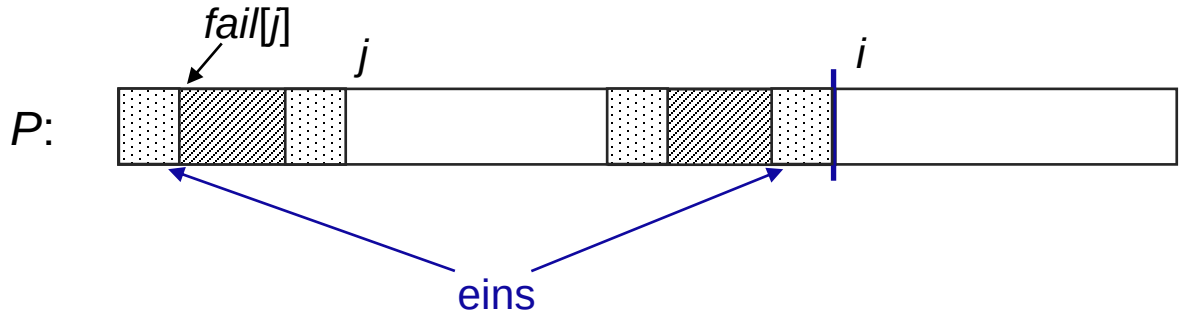

```
ComputeFailure(P[1..m]):
  j ← 0
  for i ← 1 to m
      fail[i] ← j        (∗)
      while j > 0 and P[i] ≠ P[j]
          j ← fail[j]
      j ← j + 1
```

Á meðan *P*[*i*] = *P*[*j*] þá hækka báðir vísarnir

Við erum að byggja upp forstreng sem er eins og eftirstrengur

En um leið og stafirnir passa ekki þá þurfum við að rekja okkur niður eftir *fail*-bendum

Því þeir gefa okkur lengsta forstreng sem passar við eftirstreng á þeim hluta sem við erum búin að para



Sýnidæmi fyrir *ComputeFailure*

Hluti af sýnidæmi úr kennslubókinni:

$j \leftarrow 0, i \leftarrow 1$	$\$^j$	$\mathbf{A}^i$	B	R	A	C	A	D	A	B	R	X	...
$fail[i] \leftarrow j$		**0**											...

```
COMPUTEFAILURE(P[1..m]):
    j ← 0
    for i ← 1 to m
        fail[i] ← j        (∗)
        while j > 0 and P[i] ≠ P[j]
            j ← fail[j]
        j ← j + 1
```

Tímaflækja og rökstuðningur

- Sjáum að *j* hækkar aðeins um 1 í hverri ítrun
  - en getur svo lækkað í innri lykkjunni
- Heildarfjöldi ítrana á innri lykkju getur því mest verið *m*

```
ComputeFailure(P[1..m]):
  j ← 0
  for i ← 1 to m
      fail[i] ← j        (∗)
      while j > 0 and P[i] ≠ P[j]
          j ← fail[j]
      j ← j + 1
```

Heildartími reikniritsins er því $O(m)$

Hægt að sýna að *ComputerFailure* er kvik bestunar útfærsla á endurkvæmu formúlunni:

$$fail[i] = \begin{cases} 0 & \text{if } i = 0, \\ \max\limits_{c \geq 1} \left\{ fail^c[i-1] + 1 \;\middle|\; P[i-1] = P[fail^c[i-1]] \right\} & \text{otherwise.} \end{cases}$$

þar sem $fail^c$ er ítraða *fail*-fallið:

$$fail^c[j] = fail[fail^{c-1}[j]] = \overbrace{fail[fail[\cdots[fail}^{c}[j]]\cdots]]$$

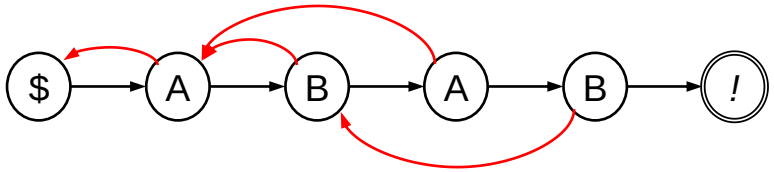
Bestun á *fail*-færslum

- Ef $T[i] \neq P[j]$ þá berum við $T[i]$ saman við $P[fail[j]]$

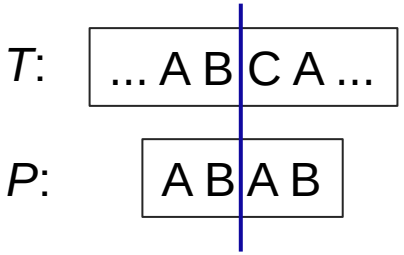
  En $P[j]$ og $P[fail[j]]$ gætu verið sami stafurinn!

Dæmi:

Mynstrið "ABAB" hefur KMP stöðuvélina

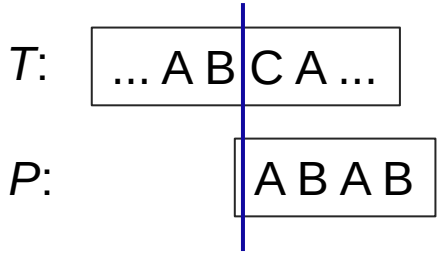


Í keyrslu:



C í texta passar ekki við A í mynstri

Hliðrum *P*, þ.a. nú borið saman við *P*[1]



En $P[1] = P[3] = A$

Ef $P[3]$ passaði ekki þá getur $P[1]$ ekki heldur passað!

Regla:

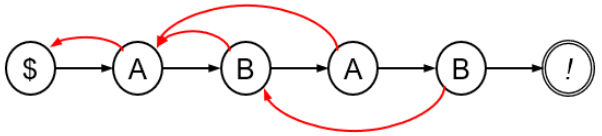
Stöðuvélin ætti aldrei að hafa *fail*-ör yfir í sama staf

Framkvæmd á bestun *fail*-færsla

- Gætum lagað þetta með því að renna í gegnum *fail*-fylkið eftir *ComputeFailure*

```
OptimizeFailure(P[1..m], fail[1..m]):
    for i ← 2 to m
        if P[i] = P[fail[i]]
            fail[i] ← fail[fail[i]]
```

Kallað á þetta fall þegar *ComputeFailure* hefur lokið keyrslu



Dæmi:

Mynstrið "ABAB"

Upphaflegt *fail*:

0	1	1	2
1	2	3	4

P[2] ≠ *P*[*fail*[2]] (B ≠ A) svo *fail*[2] óbreytt

P[3] = *P*[*fail*[3]] (A = A) svo *fail*[3] = *fail*[*fail*[3]]

P[4] = *P*[*fail*[4]] (B = B) svo *fail*[4] = *fail*[*fail*[4]]

Bestað *fail*:

0	1	0	1
1	2	3	4

Fyrirlestraæfingar

1. Teiknið upp stöðuvélina fyrir mynstrið "banana".

2. Hver er lengsti eiginlegi forstrengur (*proper prefix*) "abababababa", sem er líka eftirstrengur (*suffix*) hans?

3. Hvert er hæsta gildið sem getur verið í fylkinu ***fail***[1..*m*]? Sýnið dæmi um mynstur sem gefur það gildi.