Szuffix tömbök, Burrows-Wheeler transzformáció és FM indexek

Lehotay-Kéry Péter

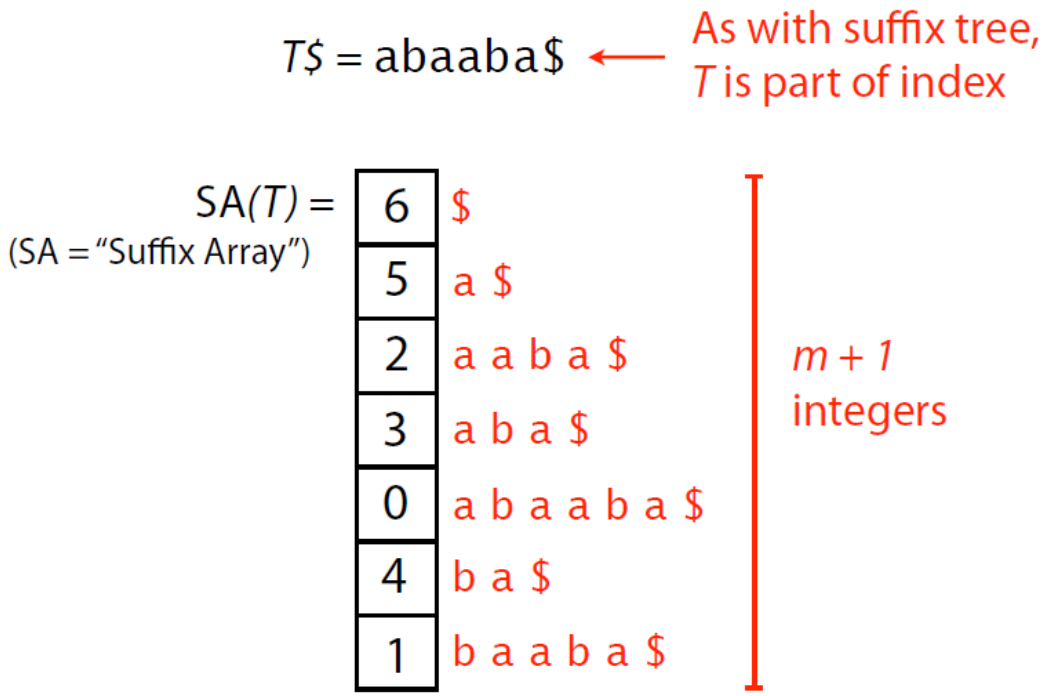
lepuaai@inf.elte.hu

Ben Langmead diasora alapján

www.langmead-lab.org/teaching-materials

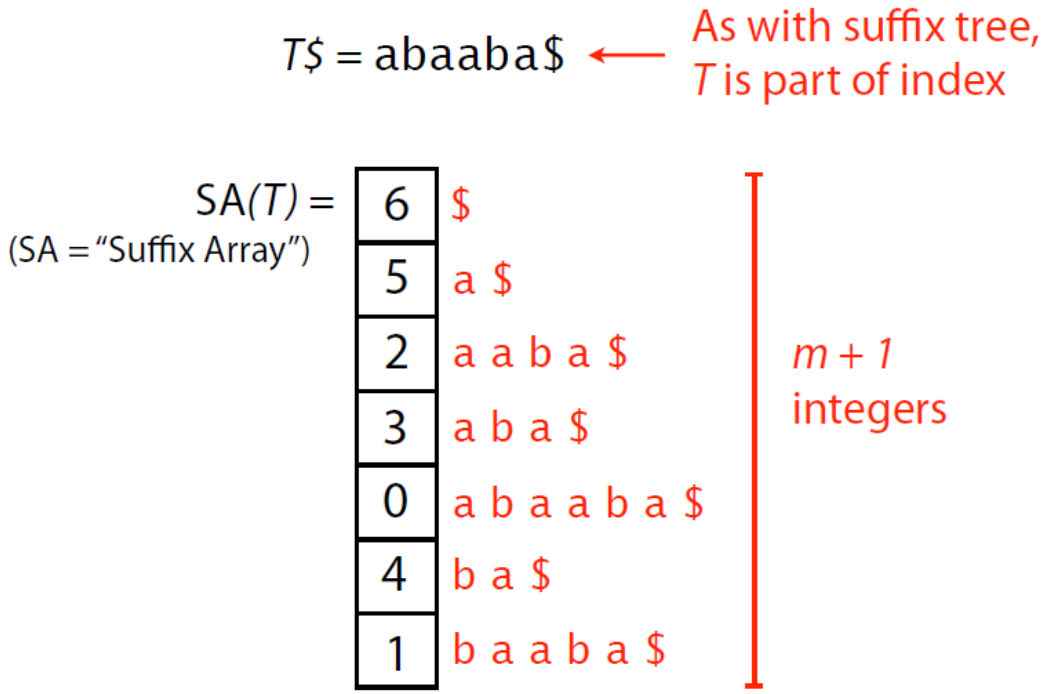
Szuffix tömb

- Szuffix tömbje T-nek egy integer tömb [0,m]-ben, mely leirja a lexikongrafikus sorrendjét T$ szuffixeinek.



Szuffix tömb: lekérdezés

- P részstringje T-nek?
  - Hogy P a részstringje legyen, prefixének kell lennie T legalább egy szuffixének.
  - Szuffixek, melyek egy prefixen osztoznak, követik egymást a szuffix tömbben.
  - Használjunk bináris keresést.



Szuffix tömb: lekérdezés

- P szuffixe T-nek?
  - Csináljunk bináris keresést, hogy P prefixe-e egy szuffixnek.
- Hányszor fordul elő P T-ben?
  - Meghatározzuk a hosszát a szuffixeknek, P prefixszel, ez megegyezik annak számával, ahányszor P előfordul T-ben.
- Legrosszabb eset?
  - O(log2m) keresés, O(n) összehasonlitás találatonként, O(n log m)

Szuffix tömb: lekérdezés

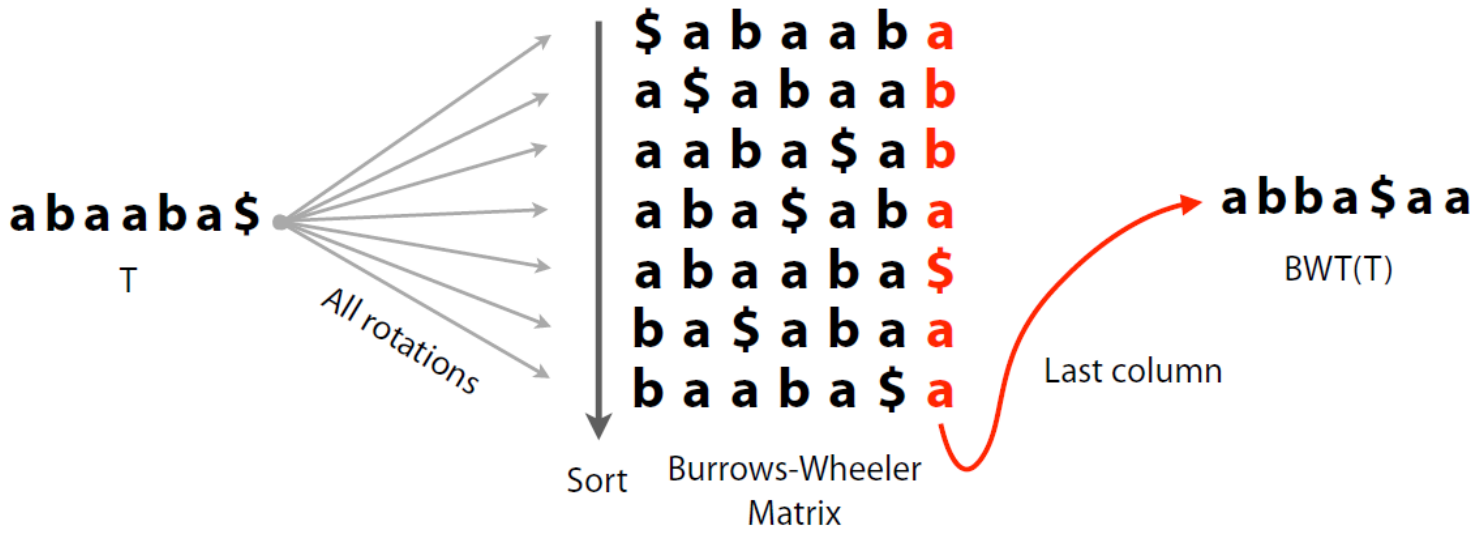
- Továbbgondolás: bináris keresés szuffixekre P-vel prefixként.
- Tegyük fel, hogy nincs $ P-ben. Igy P nem egyezhet meg egy szuffixel.
- Inicializáljuk l=0, c=floor(m/2) és r=m. (bal, középső, jobb)
- SA[l]-t fogjuk használni arra, hogy hivatkozzunk a szuffix tömb l eleméhez tartozó szuffixre T[SA[l]:] helyett.
- Keresés során fenntartjuk az invariánst: SA[l] < P < SA[r]

Szuffix tömb: lekérdezés

- Minden iterációban:
  - c=floor((r+l)/2)
  - Ha P < SA[c], megállunk, vagy legyen r=c és tovább iterálunk
  - Ha P>SA[c], megállunk, vagy legyen l=c és tovább iterálunk
- Mikor álluk meg:
  - P<SA[c] és c=l+1 -> a válasz c
  - P>SA[c] és c=r-1 -> a válasz r

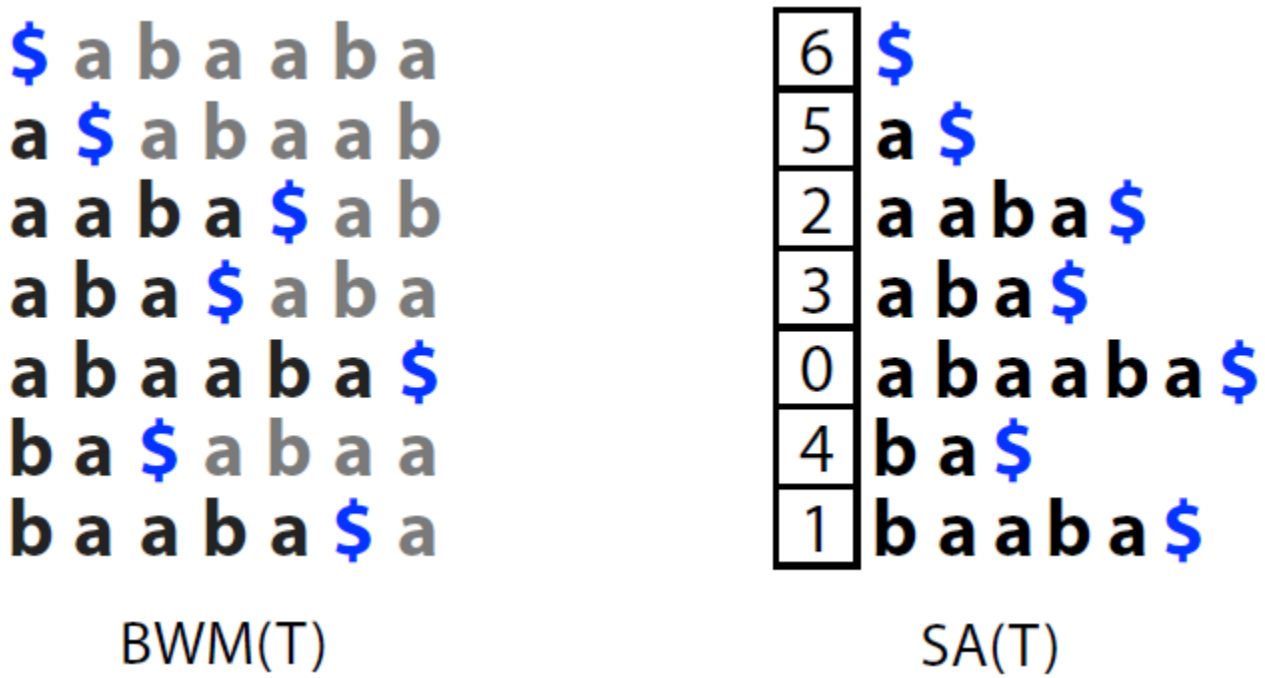
Burrows-Wheeler transzformáció

- Megfordítható permutációja egy string karaktereinek, eredetileg tömöritésre használták.



How is it useful for compression? How is it reversible? How is it an index?

Burrows-Wheeler Transzformáció



BWM(T) SA(T)

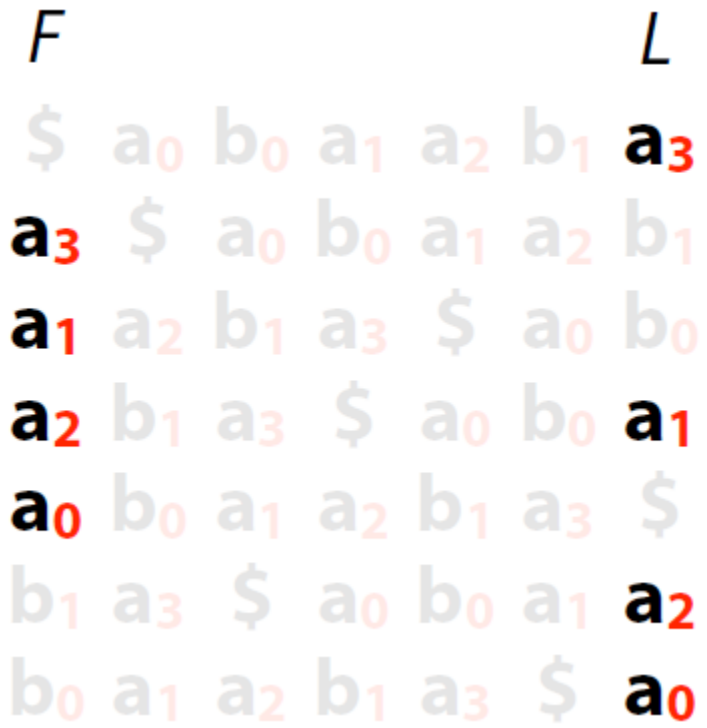
Burrows-Wheeler Transzformáció: T-rangozás

- Adjunk minden karakternek T-ben egy rangot, mely megegyezik azzal, ahányszor előfordult korábban T-ben. Legyen ennek a neve T-rangozás.
- Irjuk újra BWM-et rangokkal.

$a_0\ b_0\ a_1\ a_2\ b_1\ a_3\ \$$

Burrows-Wheeler Transzformáció

- Nézzük az első és utolsó oszlopokat: F és L.
- És nézzük csak az a-kat.
- a-k ugyan abban a sorredben fordulnak elő F-ben és L-ben. Mindkét oszlopban ezt látjuk: a3, a1, a2, a0



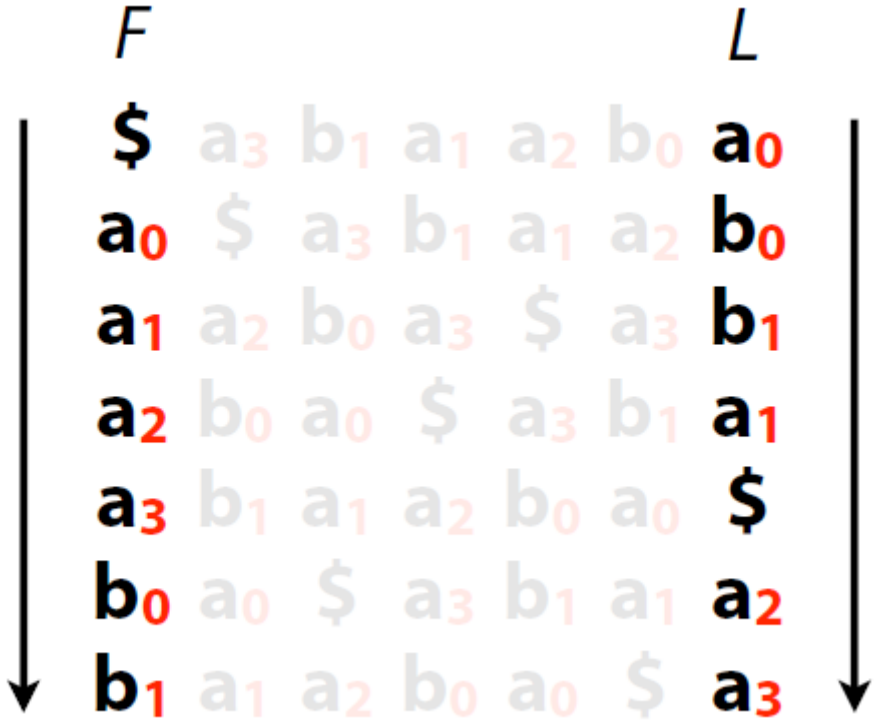
Burrows-Wheeler Transzformáció: LF térképezés

- LF térképezés: Az i. előfordulása egy c karakternek L-ben és az i. előfordulása c-nek F-ben a T-beli azonos előfordulásához tartozik.

F						*L*
$	a_0	b_0	a_1	a_2	b_1	a_3
a_3	$	a_0	b_0	a_1	a_2	b_1
a_1	a_2	b_1	a_3	$	a_0	b_0
a_2	b_1	a_3	$	a_0	b_0	a_1
a_0	b_0	a_1	a_2	b_1	a_3	$
b_1	a_3	$	a_0	b_0	a_1	a_2
b_0	a_1	a_2	b_1	a_3	$	a_0

Burrows-Wheeler Transzformáció: B-rangozás

- B-rangozás: Szeretnénk egy másik rangozást, hogy egy adott karakterre a rangok növekvő sorrendben legyenek, ahogy lenézünk F / L oszlopokra.
- F-nek igy már nagyon egyszerű a struktúrája: egy $, egy a-kból álló blokk, növekvő rangokkal, egy b-kből álló blokk növekvő rangokkal.



Burrows-Wheeler Transzformáció

- Melyik BWM sor kezdődik b1-el?
  - Hagyjuk ki a $-vel kezdődő sort (1 sor)
  - Hagyjuk ki az a-val kezdődő sorokat (4 sor)
  - Hagyjuk ki a b0-val kezdődő sort (1 sor)
- Válasz: 6. sor

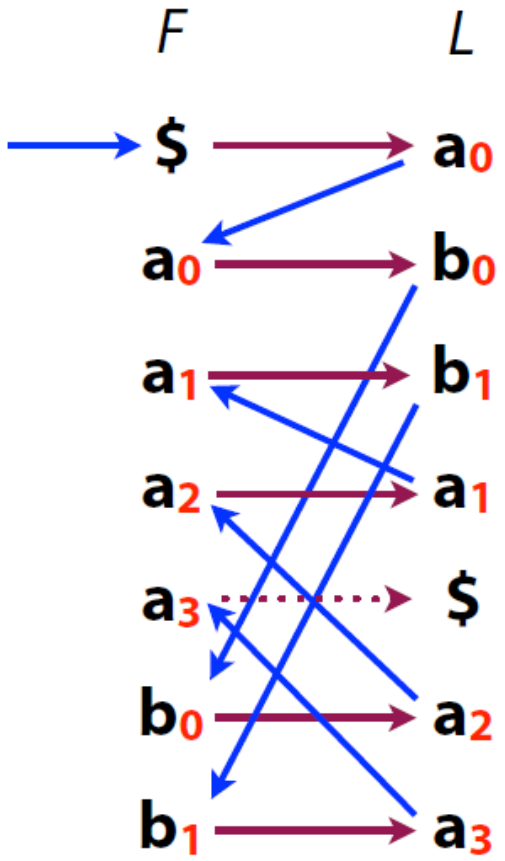
F	*L*
$	a_0
a_0	b_0
a_1	b_1
a_2	a_1
a_3	$
b_0	a_2
b_1	a_3

Burrows-Wheeler Transzformáció

- Tegyük fel, hogy T-nek 300 A-ja van, 400 C-je, 250 G-je, 700 T-je és $<A<C<G<T
- Melyik BWM sor (0-alapú) kezdődik G100-al? (B-rangokkal)

Burrows-Wheeler Transzformáció: Megfordítás

- Kezdjük az első sorban. F-ben biztos van $. L-ben a $-val szembeni karakter: a0
- a0: LF térképezés azt mondja, ez ugyan az az előfordulása a-nak mint F-ben az első a-nak. Ugorjunk az a0-val kezdődő sorba. L-ben az a0-val szembeni karakter: b0.
  - Ismételjük b0-ra, a2-t kapunk
  - …
  - Ismételjük a3-ra, $-t kapunk, kész vagyunk
- A meglátogatott karakterek megforditva: a3b1a1a2b0a0$ = T

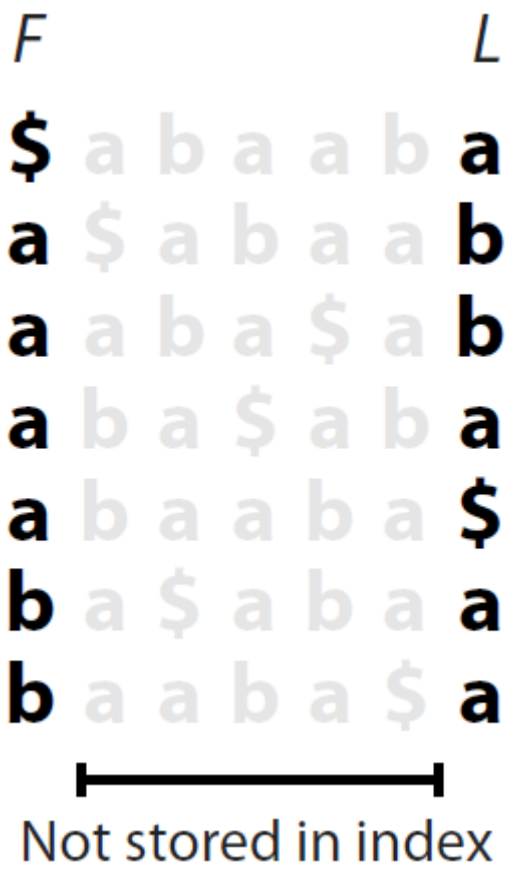


Burrows-Wheeler Transzformáció

- Láttuk, BWT hogyan használható tömöritésre:
  - Rendezzük a karaktereket, egy tömörithetőbb stringbe.
- És hogyan fordítható meg:
  - Ismételt alkalmazása az LF térképezésnek, újragyártása T-nek jobbról balra.
- Hogyan használható indexelésre?

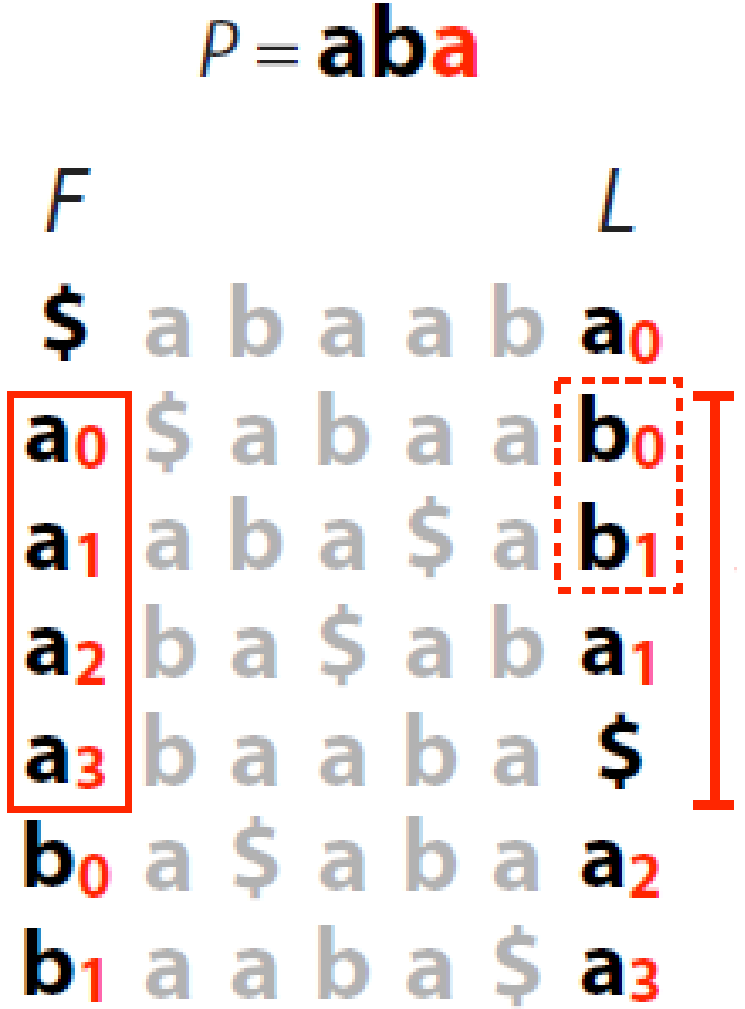
FM index

- Az indexek magja BWM F és L oszlopaiból áll:
- F nagyon egyszerűen reprezentálható:
  - 1 integer/ábécé karakter
- És L tömörithető



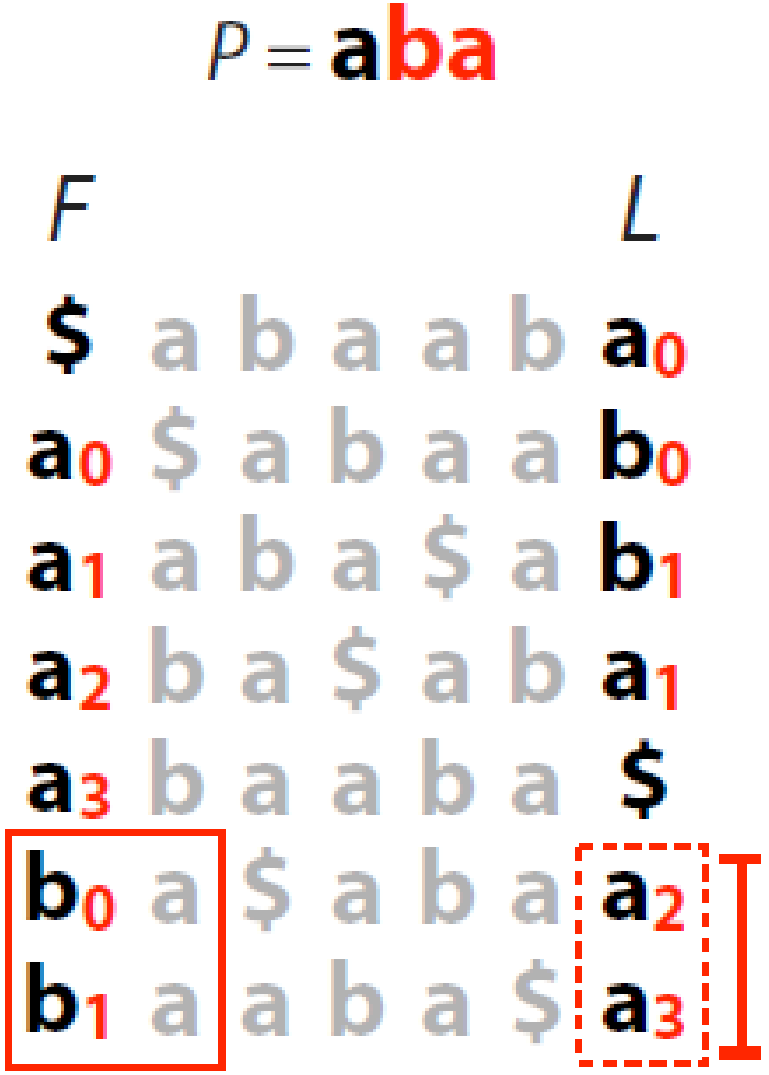
FM Index: lekérdezés

- Tekintsük BWM(T) sorait P-vel prefixként.
- Tegyünk igy P legrövidebb szuffixére, ezután lépjünk hosszabb szuffixek felé, amig nem marad sorunk, vagy elfogy P.
- Megvannak az a-val kezdődő sorok.
- Nézzük meg ezeket L-ben: b0, b1 b előfordulásai következnek P-ben



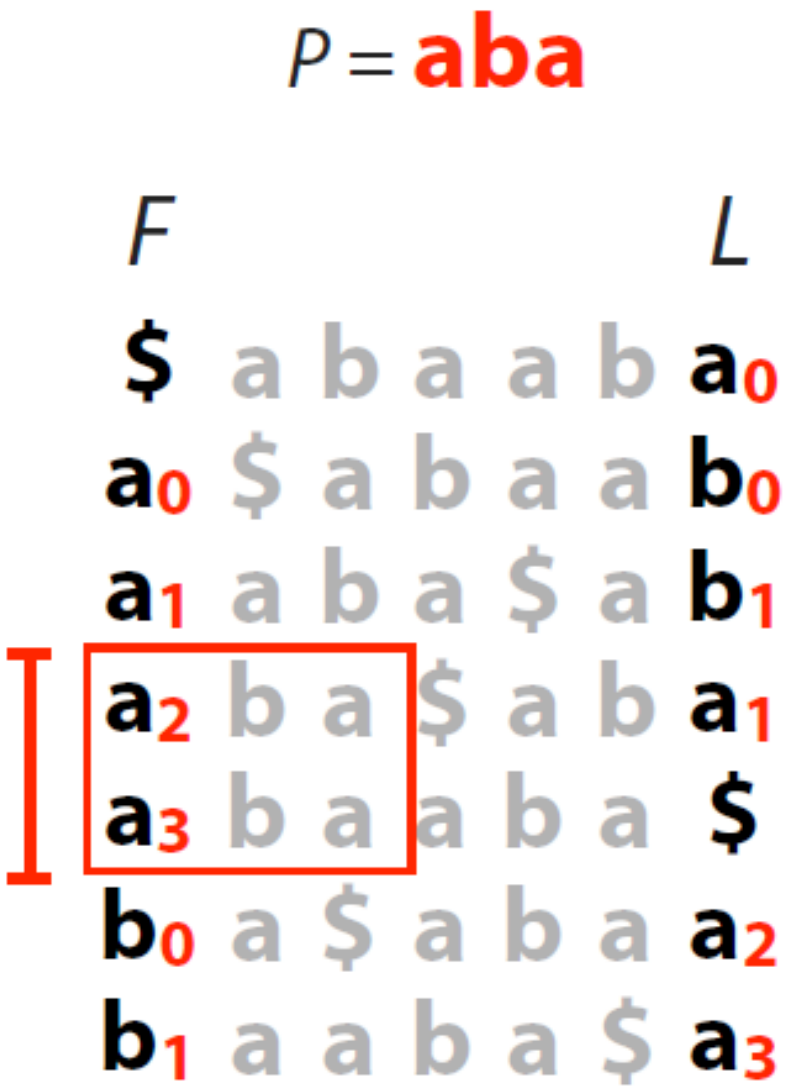
FM Index: lekérdezés

- Keressünk ba-val kezdődő sorokat.
- LF térképezéssel tekintsük előbbi b-kkel kezdődő sorokat.
- Megvannak a ba-val kezdődő sorok.
- a2, a3 következik P-ben.



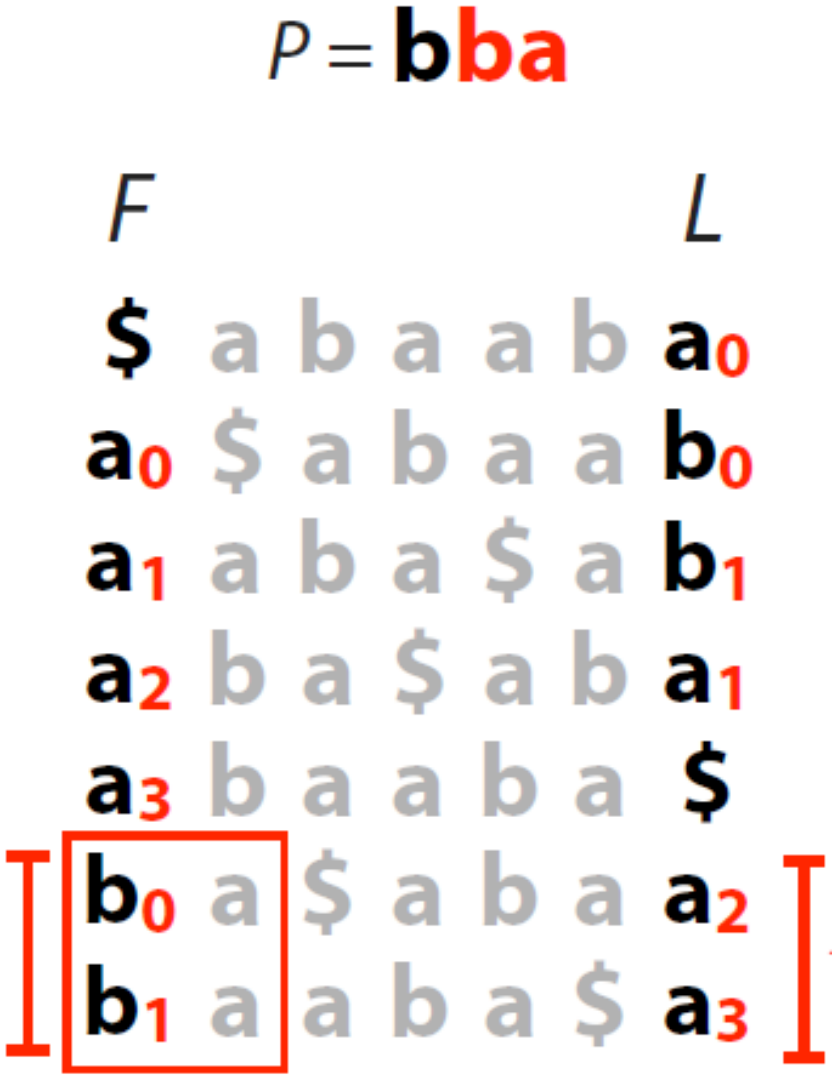
FM Index: lekérdezés

- Keressük aba-val kezdődő sorokat.
- Megvannak a sorok aba prefix-szel.



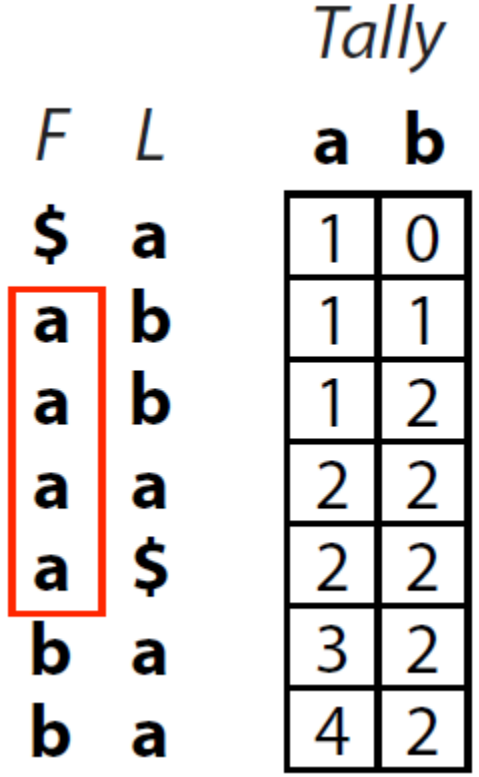
FM Index: lekérdezés

- Ha P nem fordul elő T-ben, nem sikerül megtalálnunk a következő karaktert L-ben:



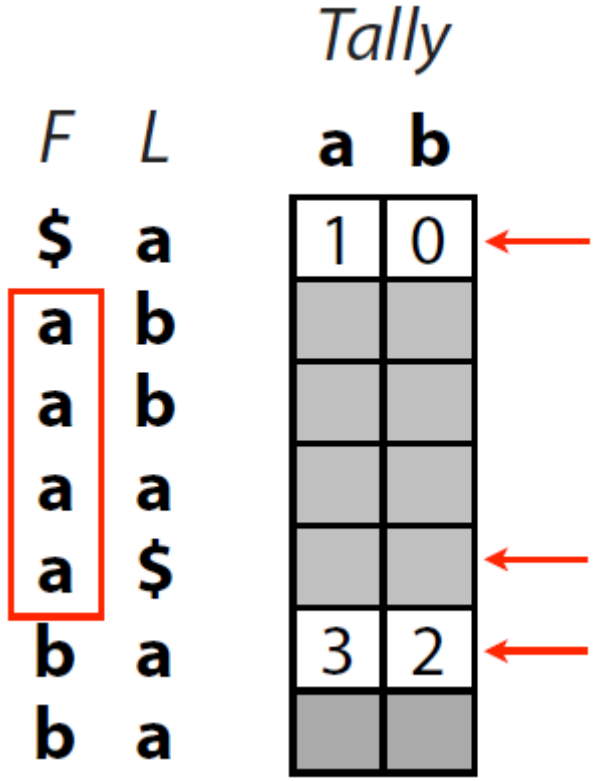
FM Index: gyorsabb rang számitás

- Megelőző karakterek keresése lassú (O(m))
- Létezik O(1) módja annak, hogy meghatározzuk, melyik b-k előzik meg az a-kat?
- Számoljuk ki előre minden sorra L-ben, hány a és b van az adott sorig:
- O(1) idő, de $m \times |\Sigma|$ integer



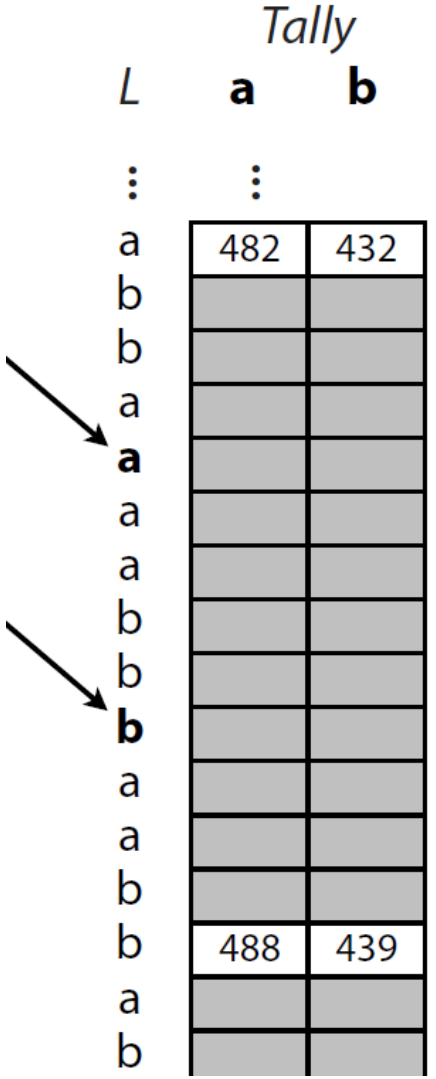
FM Index: gyorsabb rang számitás

- Másik ötlet: számitsuk ki hány a és b van L-ben néhány sorig, pl minden 5. sorig.
- Ezek lesznek az ellenőrzési pontok.
- Hogy feloldjuk egy c karakter kikeresését egy nem ellenőrzési pont sorban:
  - Tekintsük át L-t a legközelebbi ellenőrzési pontig.
  - Használjuk az ellenőrzési pontot, majd hangoljuk aszerint, hány c karaktert láttunk az ide vezető úton.



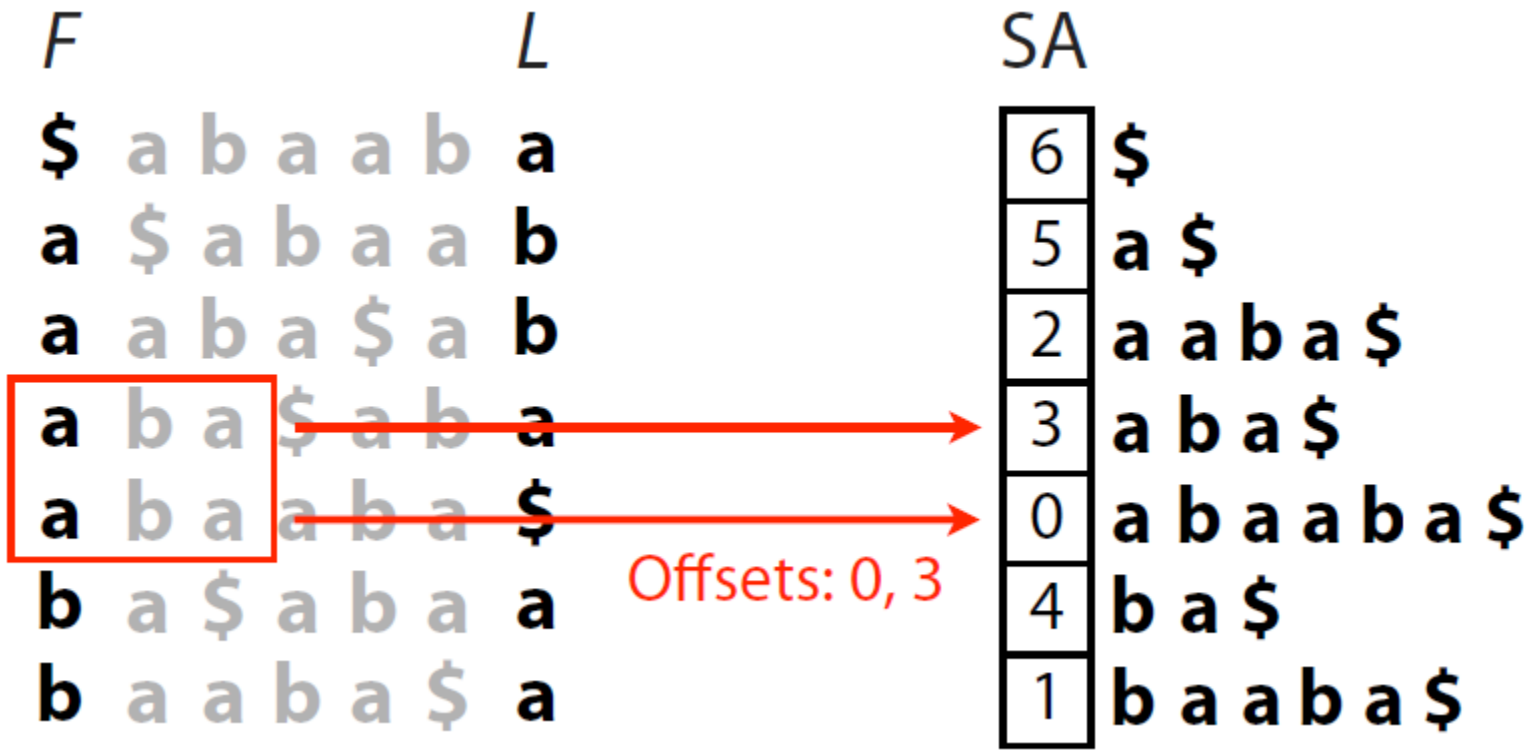
FM Index: gyorsabb rang számitás

- Mennyi lesz a kiemelt a rangja?
- Mennyi lesz a kiemelt b rangja?



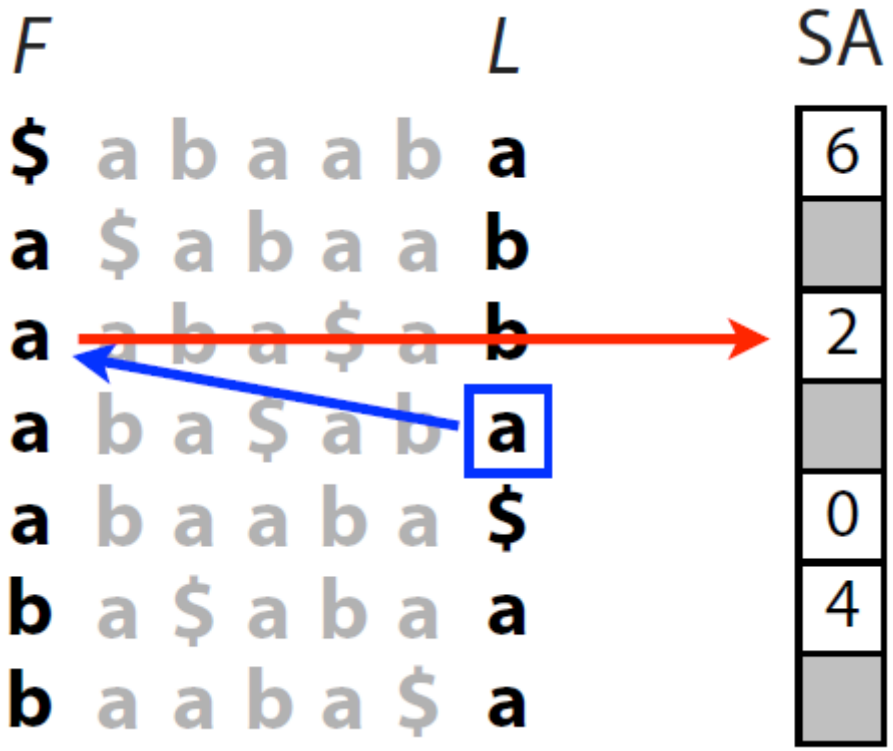
FM Index: Találati poziciók feloldása

- Ha a szuffix tömbök részei lennének az indexnek, egyszerűen kikereshetnénk a poziciókat. De SA-nak m integer kell.



FM Index: Találati poziciók feloldása

- Ötlet: tároljunk néhány, de ne az összes bejegyzést a szuffix tömbből.
- LF térképezés megmondja, hogy az ‚a' a 3. sor végén az a-hoz tartozik a 2. sor elején.
- És a 2. sornak a szuffix tömb értéke 2.
- Szóval a 3. sor szuffix tömb értéke 2+1 (hány lépést tettünk a 2. sorig)



FM Index: kis memória lenyomat

- FM Index komponensei:
  - Első oszlop (F): ~ | Σ | integer
  - Utolsó oszlop (L): m karakter
  - SA minta: $m \cdot a$ integer, ahol a egy tört, mely azt jelöli, mennyi sort tartottunk meg.
  - Ellenőrző pontok: $m \times |\Sigma| \cdot b$ integer, ahol b egy tört, mely azt jelöli, mennyi sort tartottunk meg.
- Példa: DNS ábécé (2 bit nucleotidonként), T=emberi genome, a=1/32, b=1/128
  - Első oszlop (F): 16 byte
  - Utolsó oszlop (L): 2 bit * 3 milliárd karakter = 750 MB
  - SA minta: 3 milliárd karakter * 4 byte/karakter / 32 = ~ 400MB
  - Ellenőrző pontok 3 milliárd * 4 byte/karakter / 128 = ~ 100 MB
  - Összesen < 1.5 GB

Feladat

- Készitsetek méréseket, hogy az IndexSorted és FM indexek mennyi idő alatt nyerik ki a lambda virus genomjából (query), hogy a P minta hol található meg benne, valamint nézzétek meg azt is, mennyi memóriát foglalnak ezek az indexek ekkor!
- Lambda virus elérhetősége:
  - http://d28rh4a8wq0iu5.cloudfront.net/ads1/data/lambda_virus.fa
- P minta:
  - ACTAAGT