Közelítő illesztés és indexelés részstringekkel

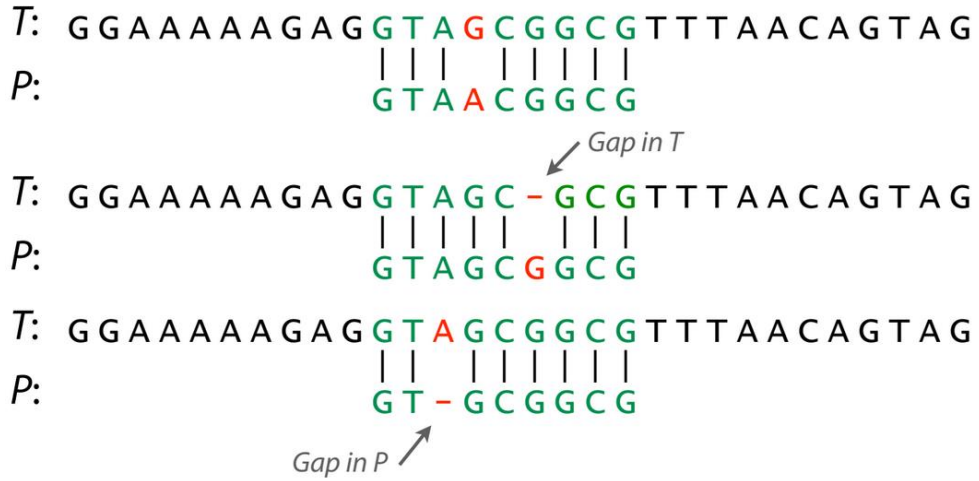
Lehotay-Kéry Péter
lepuaai@inf.elte.hu
Ben Langmead diasora alapján
(www.langmead-lab.org/teaching-materials)

Szükség van közelitő illesztésre

- Szekvencia eltérések miatt
  - Szekvenálási hibák
  - Természetes variációk

Közelitő string illesztés

- Olyan helyeket keresünk, ahol P illeszkedik T-re, megengedve egy adott számú eltérést, vagy szerkesztést.
- Egy eltérés egy karakternyi helyettesités.
- Egy szerkesztés lehet egy karakternyi helyettesités, vagy hézag.



Hamming és szerkesztés távolság

- 2 azonos hosszú X és Y stringre, a hamming távolság a szükséges minimális számú egy-karakteres behelyettesitési, ahhoz hogy egyik stringet a másikká változtassuk.

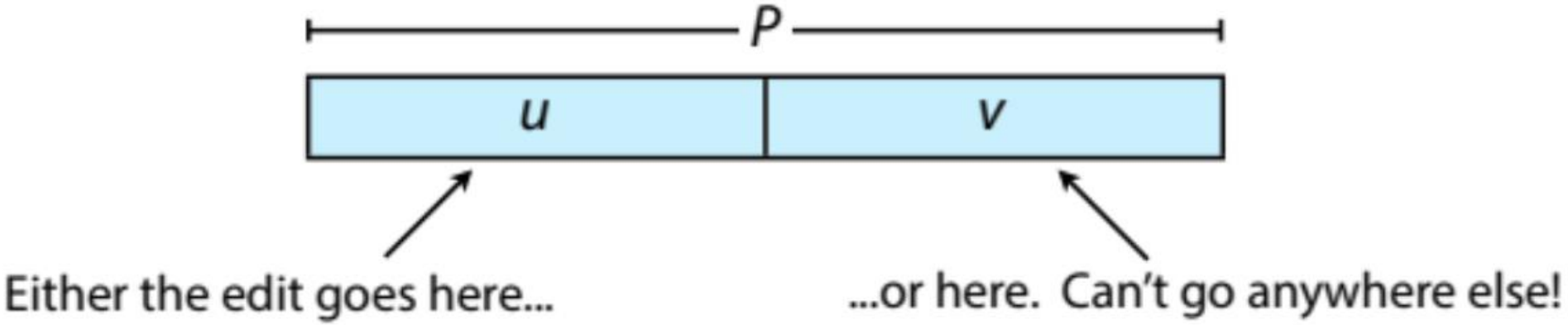
Hamming distance = 3

- Szerkesztés távolság a szükséges minimális számú szerkesztés, ahhoz, hogy egyik a másikká váljon.

Edit distance = 2

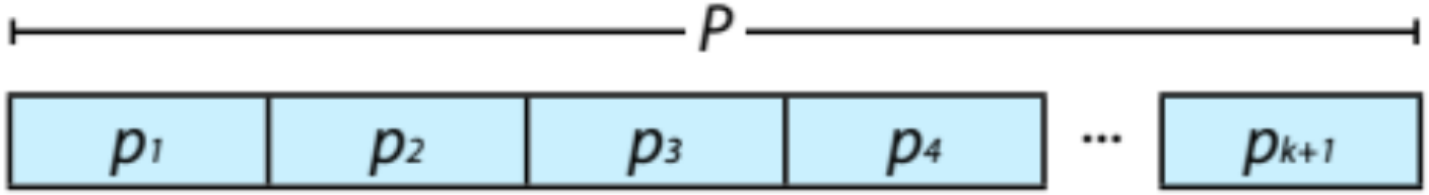
Közelitő string illesztés

- Hogyan tegyük Boyer-Moore és indexek által segitett illesztéseket közelitővé?
- Osszuk P-t nem-üres, nem-átfedő részstringekre: u és v. Ha P előfordul T-ben 1 szerkesztéssel, u-nak, vagy v-nek pontosan egyeznie kell.



Közelitő string illesztés

- Általánosabban: Legyen p1, p2, ..., pk+1 P particionálása k+1 nem-átfedő, nem-üres stringekbe. Ha P előfordul T-ben maximum k szerkesztéssel, akkor p1, p2, ..., pk+1 közül legalább egynek pontosan kell egyeznie.



- Használjuk a pontos illesztés algoritmust hogy pontos illeszkedést találjunk p1, p2, ..., pk+1-el. Keressünk hosszabb közelitő illeszkedést a pontos illeszkedés körül.

Előfeldolgozás

- Láttuk a naive algoritmust és Boyer-Moore-t. Ezek és más algoritmusok szétválaszthatók az alapján, milyen előfeldolgozást végezek.
- Naiv nem végez előfeldolgozást.
- Boyer-Moore előre feldolgozza P mintát:
  - Ha P előre adott, kikereső táblákat épithetünk a rossz karakter és jó szuffix szabályokhoz.
  - Ha T1 később jön, használjuk az előre felépitett táblákat, hogy összevessük P-t T1-el
  - T2, T3, ..-vel ugyan ez a helyzet.

Indexelés

- P-t, T-t vagy mindkettőt is feldolgozhatjuk előre. Amikor T-t dolgozzuk fel, azt mondjuk indexet készitünk, mint egy könyv indexe.
- Webes keresőmotorok is indexelést használnak:

Documents:

```
T[0] = "it is what it is"
T[1] = "what is it"
T[2] = "it is a banana"
```

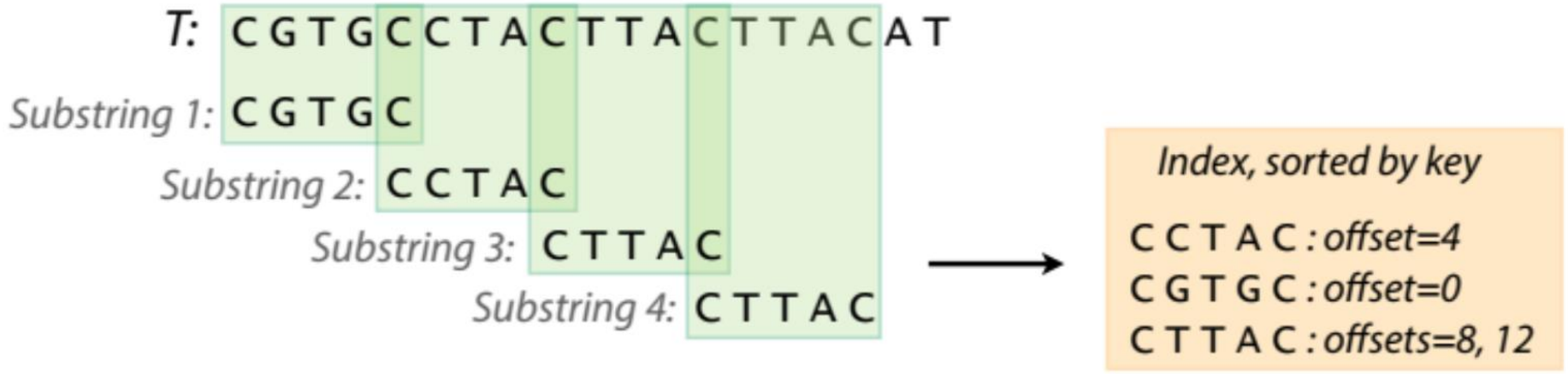
http://en.wikipedia.org/wiki/Inverted_index

Index:

```
"a":      [2]
"banana": [2]
"is":     [0, 1, 2]
"it":     [0, 1, 2]
"what":   [0, 1]
```

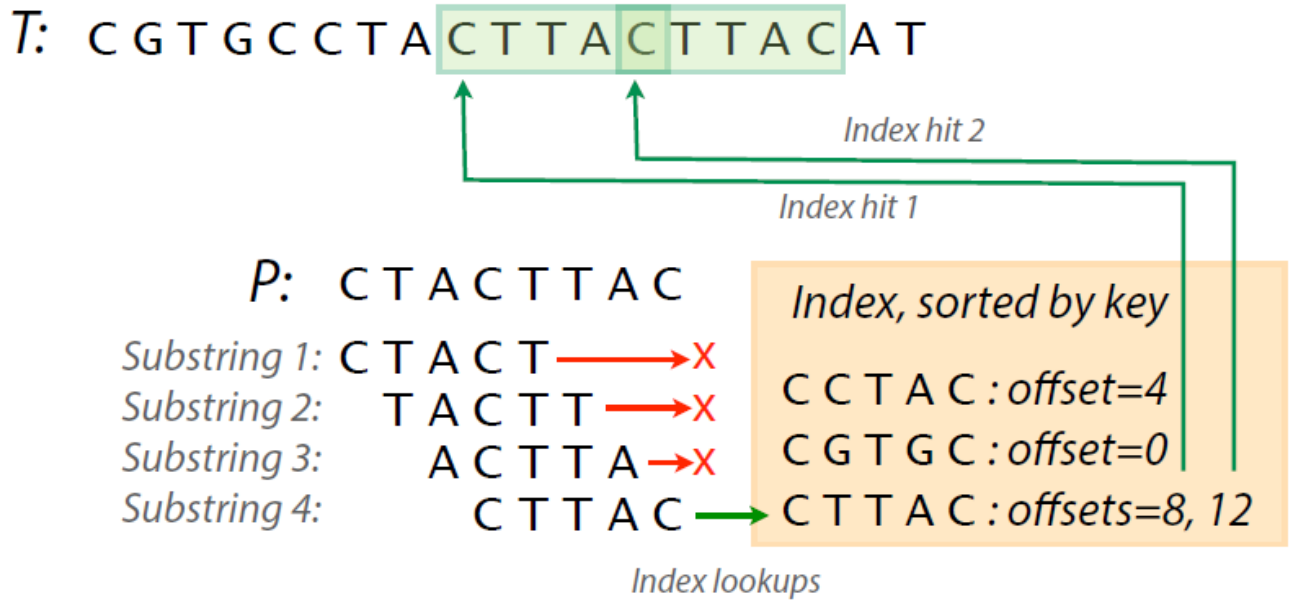
Részstring index

- T indexelése:
  - Kinyerjük a szekvenciákat T-ből (általában részstringeket), mutatókkal, hogy hol fordultak elő.
  - Rendezzük a darabokat és mutatókat egy adatszerkezetbe. Különböző szerkezetek lehetnek, mindegyik magába foglal valamilyen csoportositást, rendezést.



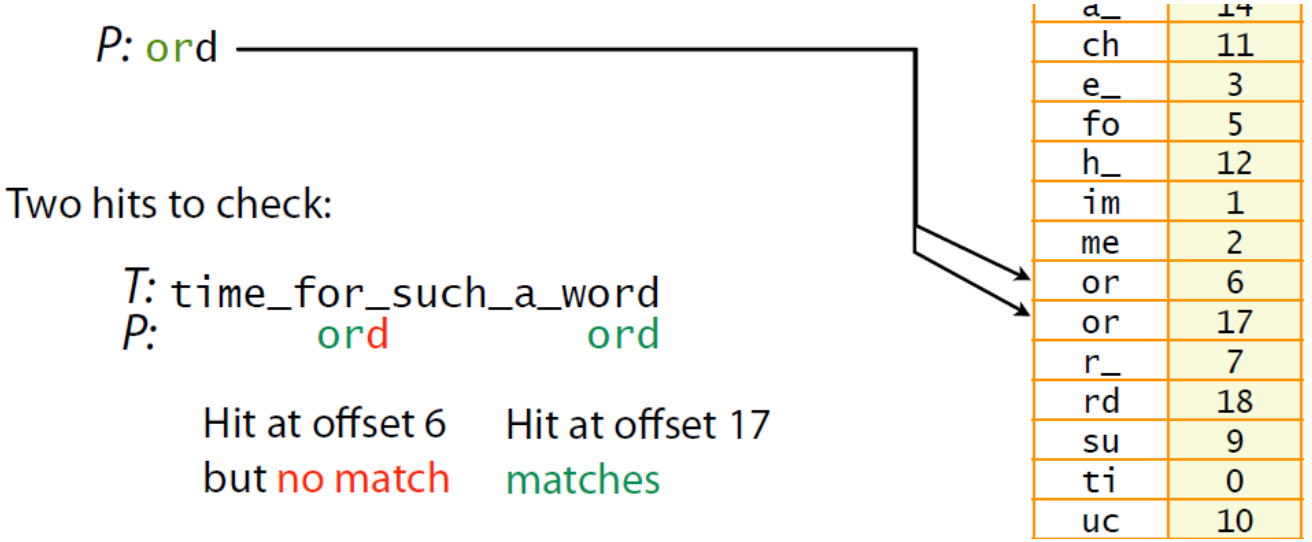
Részstring index

- Index lekérdezése P-vel:
  - Kinyerjük a részstringeket P-ből. Ezeket használjuk, hogy lekérdezzük az indexet. Index megmondja nekünk, hol fordulhatnak elő T-ben.
  - Minden előfordulásra, megnézzük, van-e illeszkedés ezen a környéken



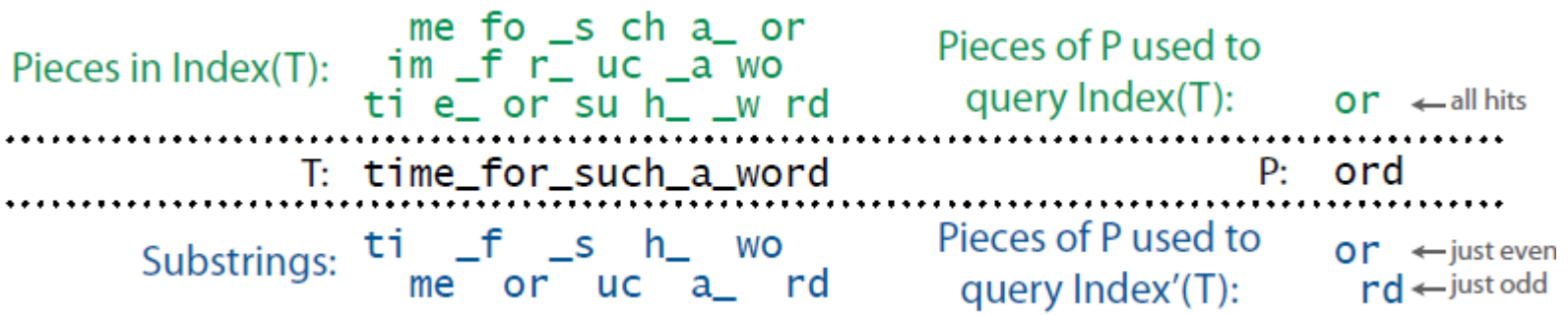
Részstring index

- Index T-hez: rendezett tábla, minden 2-hosszú részstringhez és előfordulásukhoz.
- Lekérdezéskor: kinyerjük a 2-hosszú prefixét P-nek, kikeressük őket az indexből, megvizsgáljuk a találatokat.



Részstring index: minden második részstring

- Ahelyett, hogy minden 2-hosszú részstringet kinyerjük, kihagyunk párat, mondjuk minden másodikat.
- T lekérdezésekor:
  - Kinyerjük 2 legszélső 2-hosszú részstringjét P-nek, megkeressük az indexben, kipróbálunk minden nevezőt.
  - Első kikeresés megtalál majd minden illeszkedést a páros előfordulásokra, második a páratlan előfordulásokra.



T: `time_for_such_a_word`

Index'(T):

Substring	Offset
_f	4
_s	8
a_	14
h_	12
me	2
or	6
rd	18
ti	0
uc	10
wo	16

Részstring index: minden N-edik részstring

- Ahogy kivettük minden 2. részstringet, úgy kivehetjük minden 3., 4., stb
- Ha kivesszük minden N. részstringet, használnunk kell P első N részstringjét a lekérdezéshez.

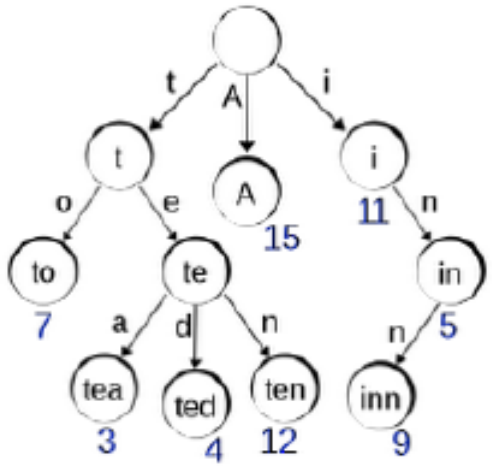
Indexelés: adatszerkezetek

- Indexek párositják a részstringeket, vagy részszekvenciákat a találatokhoz.
- Több alkalmas adatszerkezet is létezik:

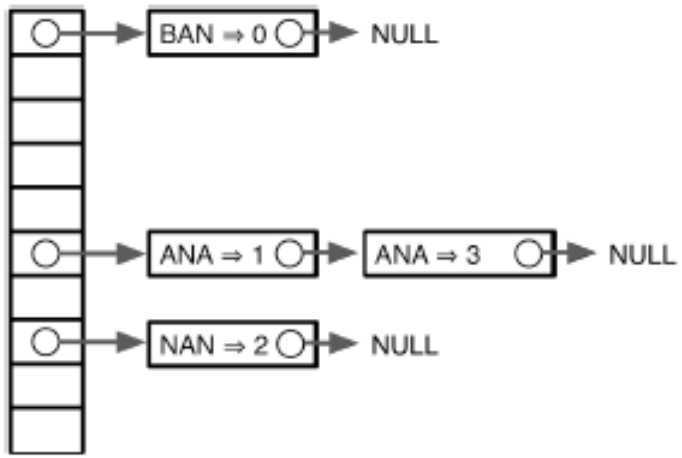
Sorted list:

_f	4
_s	8
a_	14
h_	12
me	2
or	6
rd	18
ti	0
uc	10
wo	16

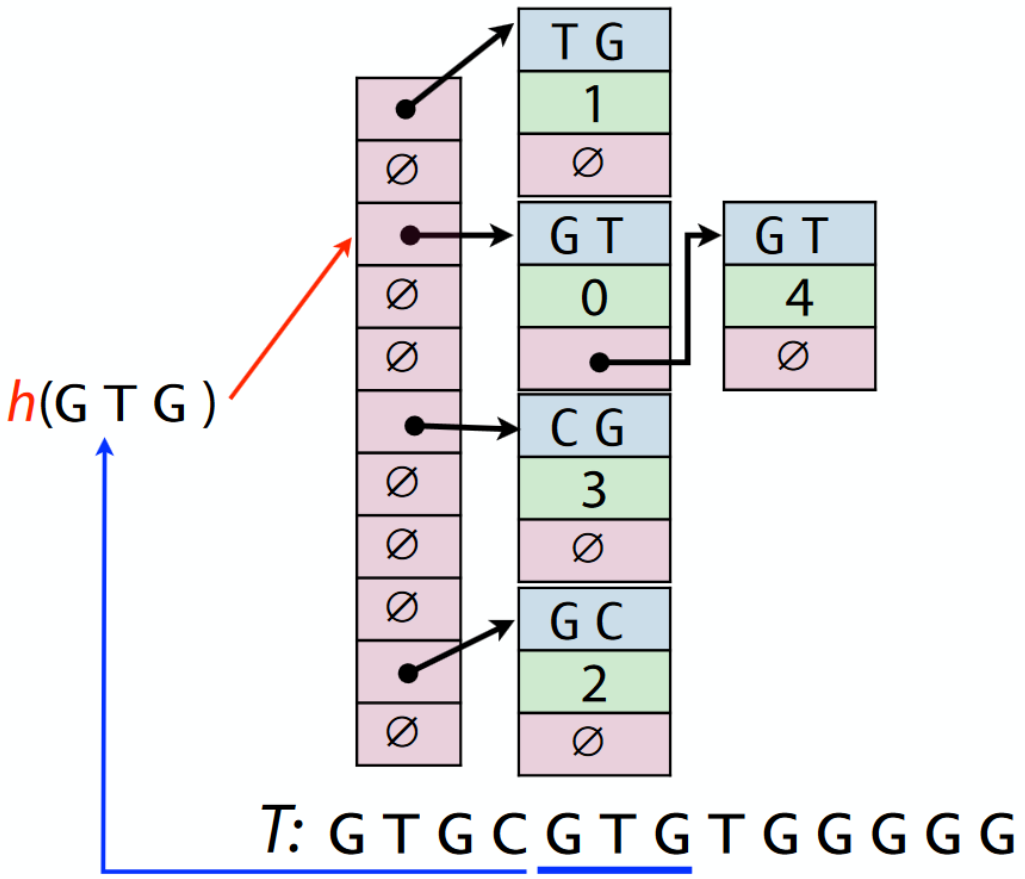
Trie:



Hash table:



Indexelés: Hash tábla áttekintés



Feladat

- Irjunk közelitő illesztést úgy, hogy az használjon indexeket! Adjuk vissza a nem egyező karakterek számát is!
- A házi feladat notebookjának linkjét küldjétek el e-mailben:
  - lkp@caesar.elte.hu
- A hiba, amibe órán ütköztünk a BoyerMoore approximáció során az volt, hogy a BoyerMoore előfeldolgozás nem működött 1 karakterre, ezt lekezeltem, itt a javitott notebook:
  - https://notebooks.azure.com/lkpeter/libraries/bioinfcourse/html/bm_appr.ipynb