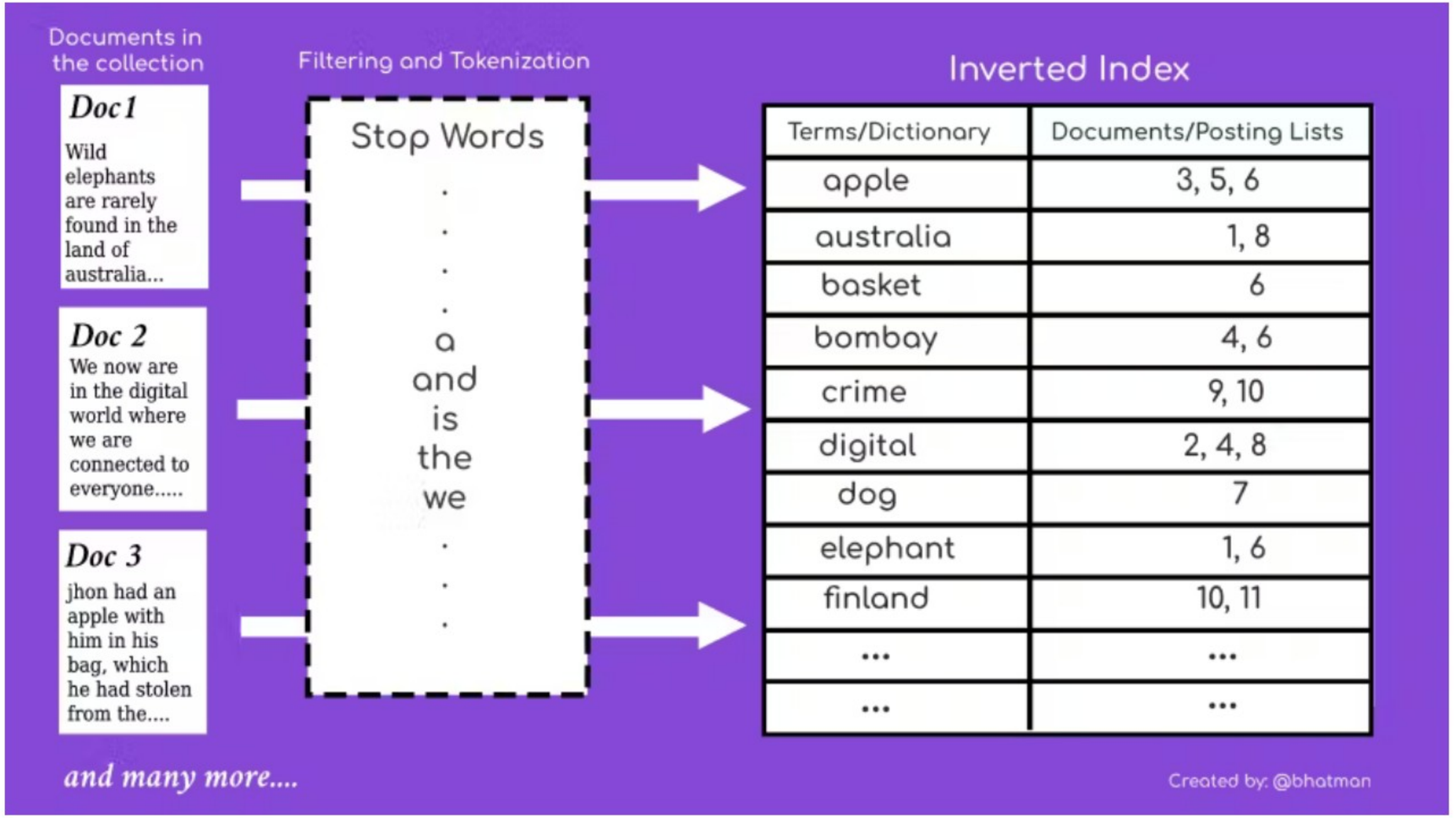
# IN[34]120 Søketeknologi

Uke 4: **FAQs & oblighjelp**

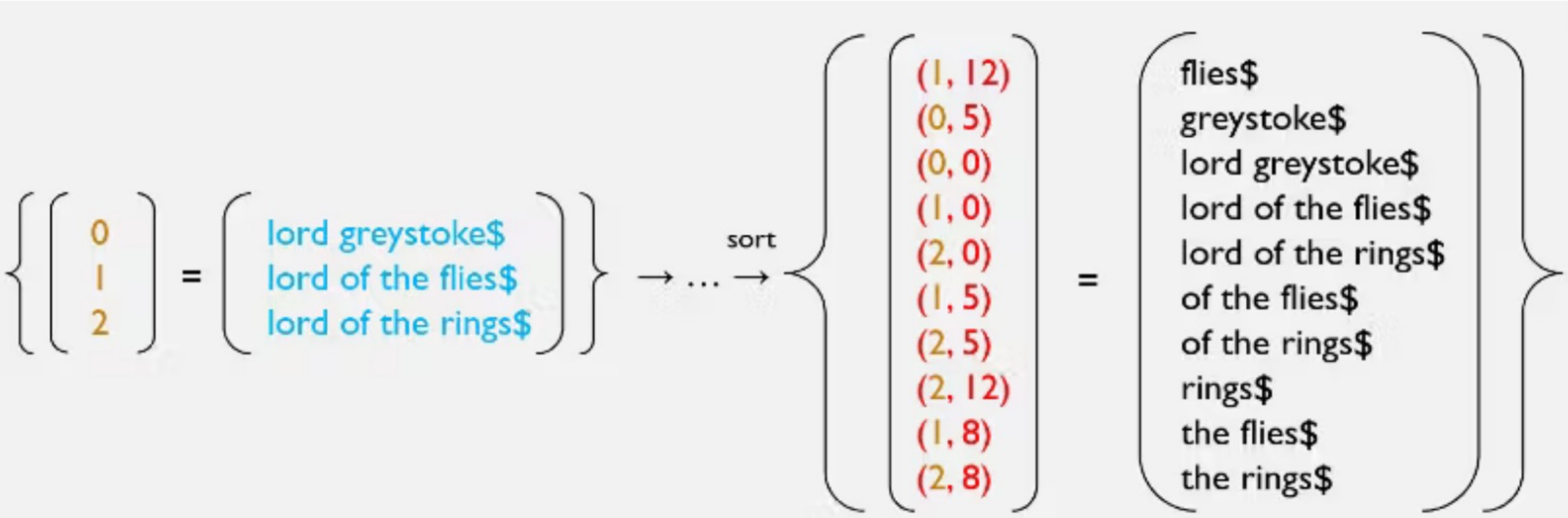
Agenda:
- Se over hvor vi har kommet
- Noen FAQs
- Oblighjelp

2024-09-17 14:15 @ Chill

Gruppelærer: Oliver, oliverrj@ifi




Rep: Inverted index m/posting lists




The application dictates what we consider to be a searchable suffix, i.e., where matches can begin and end

## Suffix Array Example

**Given String: banana**

| Suffixes | | Sorted Suffixes |
|---|---|---|
| 0 banana | | 5 a |
| 1 anana | Sort the Suffixes | 3 ana |
| 2 nana | ----------> | 1 anana |
| 3 ana | alphabetically | 0 banana |
| 4 na | | 4 na |
| 5 a | | 2 nana |

**Suffix array: {5, 3, 1, 0, 4, 2}**

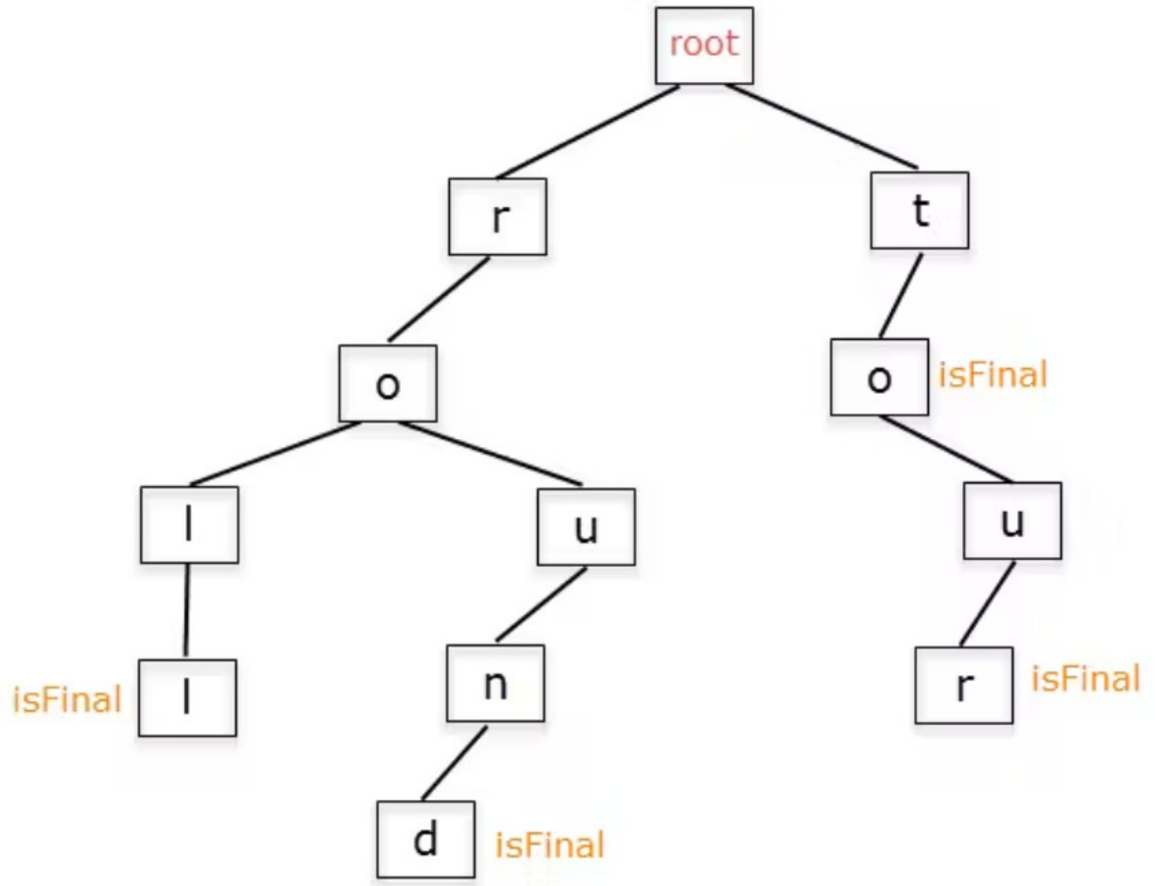
Visualisering av suffix et suffix array for en *term* (ikke det vi vil i B-1)

# NB: oblig B-1: Token boundries

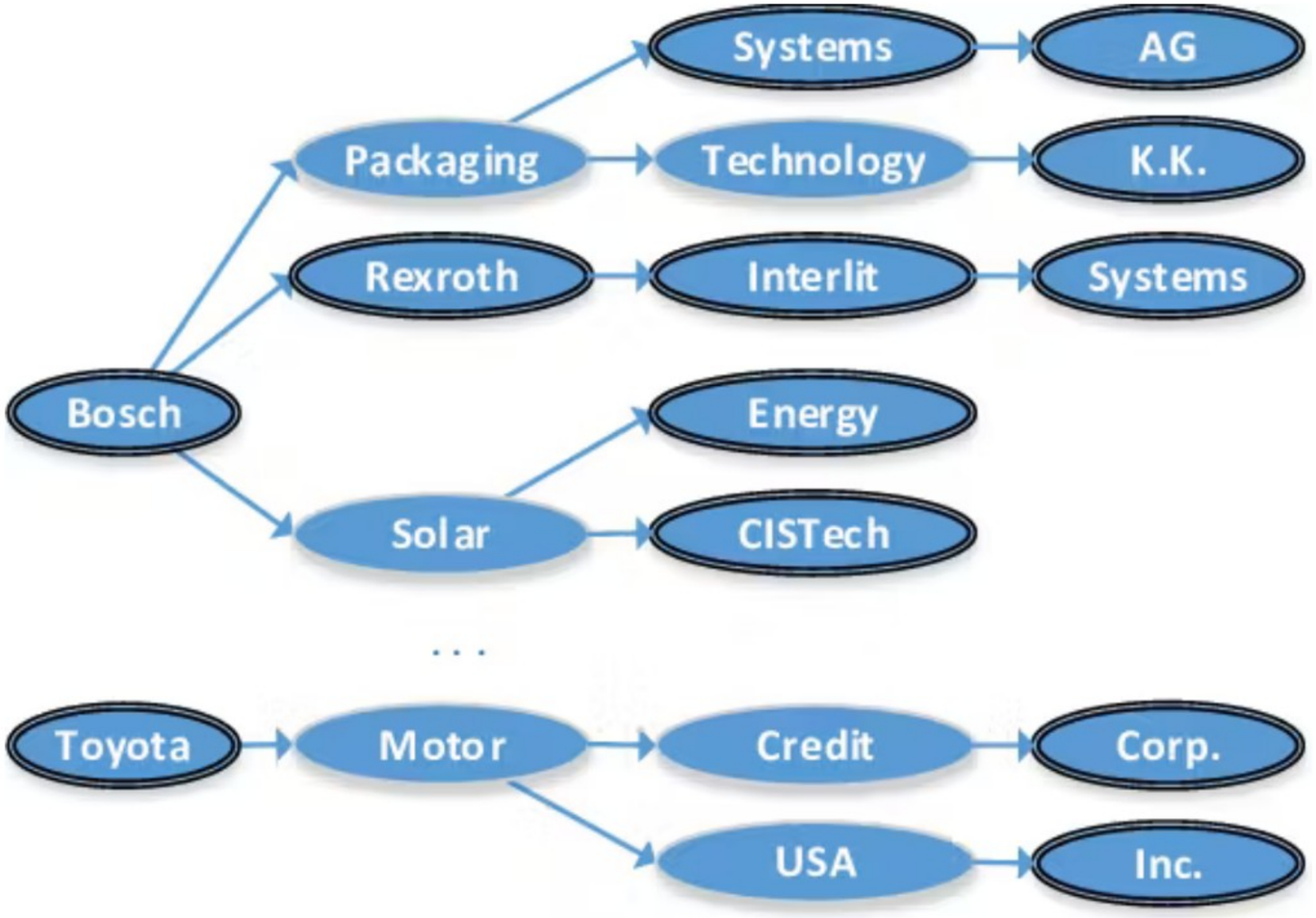
- Ett suffix for hvert token
- Ikke ett suffix for hvert tegn
- Prekoden sin tokeniser har en metode ranges()

# Tries

- Data structure - prefix tree
- Finn ut om en streng inngår i et korpus (raskt)
- Oblig B



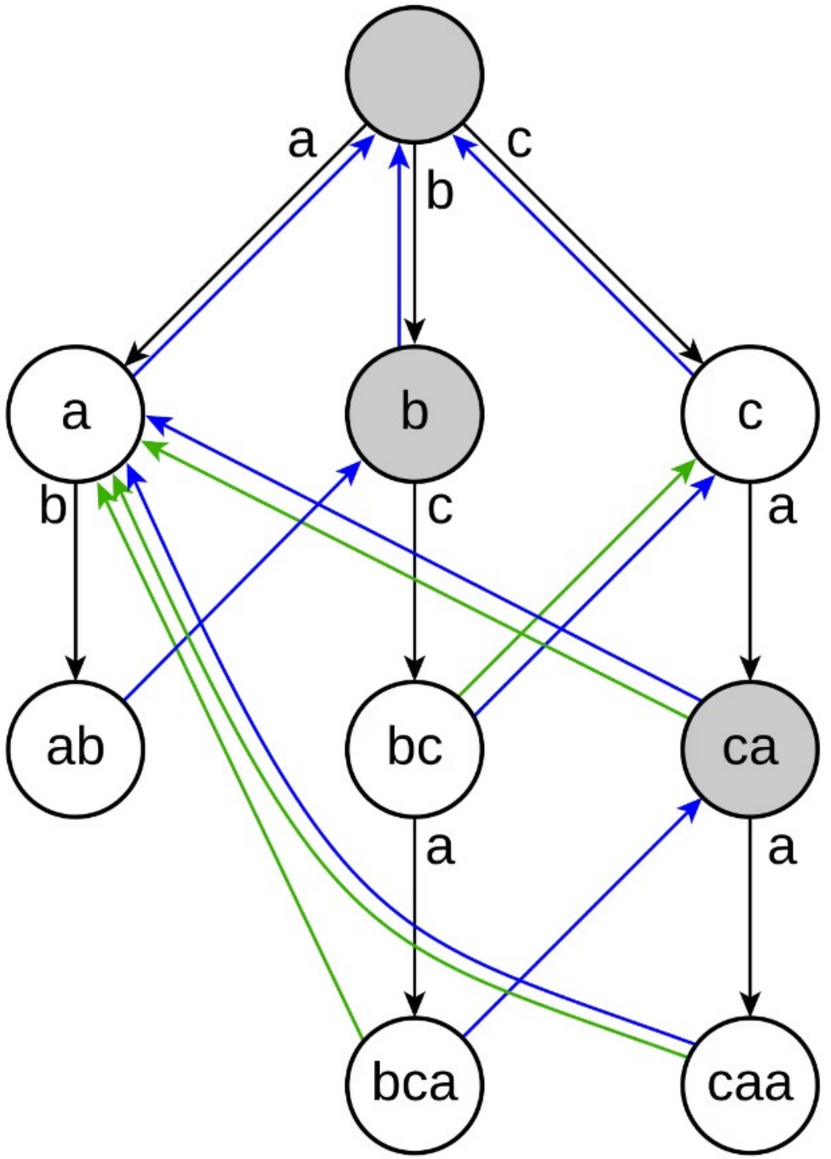

Visualisering av trie




Sauce:
https://www.researchgate.net/publication/318394164_Improving_Company_Recognition_from_Unstructured_Text_by_using_Dictionaries

# Aho-Corasick-algoritmen (*trie search*)

- Ha en liste med states
- Beveg deg ned i trien
- Se om du kommer frem til en final node
- NB: Forenklet versjon



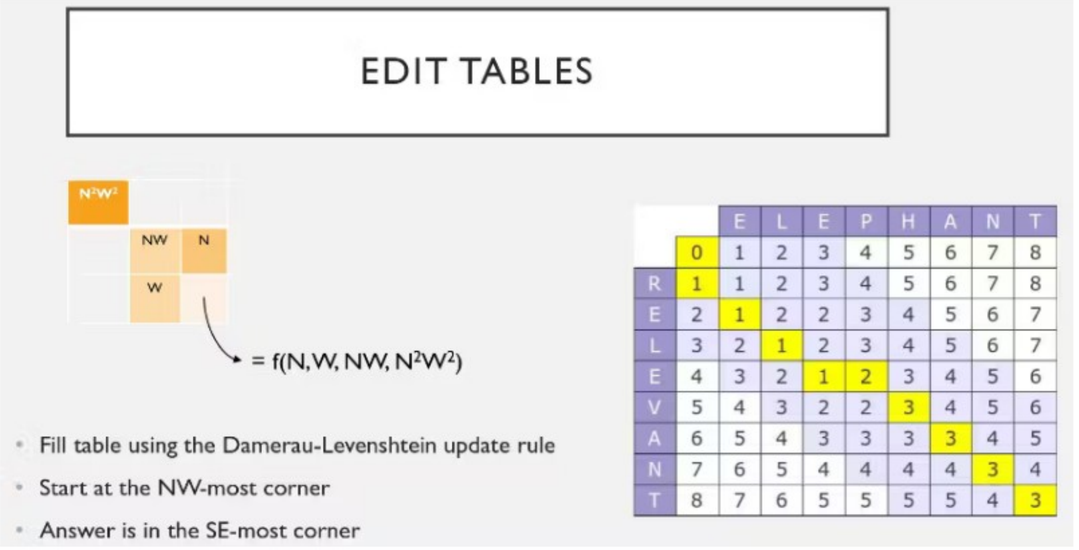
# Edit distance

- Metrikk: Hvor like er 2 strenger

| | | E | L | E | P | H | A | N | T |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| R | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| E | 2 | 1 | 2 | 2 | 3 | 4 | 5 | 6 | 7 |
| L | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| E | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| V | 5 | 4 | 3 | 2 | 2 | 3 | 4 | 5 | 6 |
| A | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 4 | 5 |
| N | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 3 | 4 |
| T | 8 | 7 | 6 | 5 | 5 | 5 | 5 | 4 | 3 |

# Edit tables

- Regne ut edit distance mellom 2 strenger
- Hopper alltid 1 hakk (hvorfor?)
- Kommer typisk på eksamen, f.x. "*hva er feil i denne tabellen?*"




| | | E | L | E | P | H | A | N | T |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| R | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| E | 2 | 1 | 2 | 2 | 3 | 4 | 5 | 6 | 7 |
| L | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| E | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| V | 5 | 4 | 3 | 2 | 2 | 3 | 4 | 5 | 6 |
| A | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 4 | 5 |
| N | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 3 | 4 |
| T | 8 | 7 | 6 | 5 | 5 | 5 | 5 | 4 | 3 |

# Oblig B

Frist: 2024-09-27

B-1 eller B-2.
Man velger hvem av dem man gjør.

Kan gjøre begge for gøy...

# Oblig B-1

- → Suffix array
- → Trie search
- → Får ferdig trie fra prekoden

## Assignment B-1

**Deadline:** 2024-09-27

The purpose of this assignment is twofold:

- Build a simple suffix array and implement "phrase prefix searches" using this. For example, for a supplied query phrase like *to the be*, we'd return documents that contain phrases like *to the bearnaise*, *to the best*, *to the behemoth*, and so on. I.e., we require that the query phrase starts on a token boundary in the document, but it doesn't necessarily have to end on one.
- Given a trie ("dictionary") with a potentially very large number of entries (words and phrases of arbitrary length), implement a simple version of the Aho-Corasick algorithm that efficiently detects the subset of dictionary entries that also occur within a given text buffer. For example, if the dictionary contains *harry potter* and *wizard*, we'd efficiently detect the presence of these strings within the buffer *a wizard named harry potter*. Detected entries should start and end on token boundaries. If entries overlap in the document then all matches should be reported.

Your solution should only contain edits to the files `suffixarray.py` and `stringfinder.py`. Changes to other files will be ignored.

# Oblig B-2

- → Approximate string matching
- → = Trie search m/edit distance
- → Basically Shang+Merretts paper

## Assignment B-2

**Deadline:** 2024-09-27

The purpose of this assignment is to implement the core idea outlined in the paper by Shang and Merrett, that shows how to reasonably efficiently compute the bounded edit distance between a query string and a large collection of candidate strings. The collection of candidate strings are represented in a trie. For example, for the query *banana* I should get back the *n* strings in my string collection that have the smallest edit distance to *banana*. We can bound the edit distance, so that we only report matches that are less than or equal to *k* edits away from the query string. Imposing such an upper bound allows us to prune down the search space and evaluate far fewer candidates than otherwise needed.

To achieve this your task is twofold:

- Implement the required edit table logic required to compute Damerau-Levenshtein distance, and equip the edit table with an interface suitable to be used together with a trie search as described in the paper by Shang and Merrett. I.e., your implemenation should be able to correctly compute both the edit distance between two arbitrary, known strings of reasonable length, as well as enable a column-by-column computation driven by a search procedure external to the table.
- Implement an efficient search procedure over the trie, that uses your edit table implementation. I.e., branches in your trie that correspond to an edit distance larger than *k* from the query should not be visited and evaluated.

Your solution should only contain edits to the files `edittable.py` and `editsearchengine.py`. Changes to other files will be ignored.

# FAQs

Et utvalg.

# FAQ 1 - administrativt

- → Kan man gjøre obligene i gruppe?
- → Når kommer feedback for oblig A?
- → Når åpner oblig B på devilry?
- → Hvilke filer skal jeg levere?

# FAQ 2 - assignment A

- Hva er greia med compressed?
- Hva skal finalize_index()gjøre?
- Hvorfor er obligen rettet med flere tester?
-

# FAQ 3 - assignment B-1

- Kan jeg stjele get_terms fra A?
- Hvorfor bruker løsningen min for mye minne?
- Kan man bruke bisect_left?
- Kan man importere andre ting, f.x. Sieve?

# Pro tip for oblig B

- I morgen: gruppe 1. Truls.
- Gjennomgang med *mange + store* hint
- (Spesielt for B-1)
- 10:15 @ Fortress

# Spørsmål? 

# NB: Forelesning på fredag !!

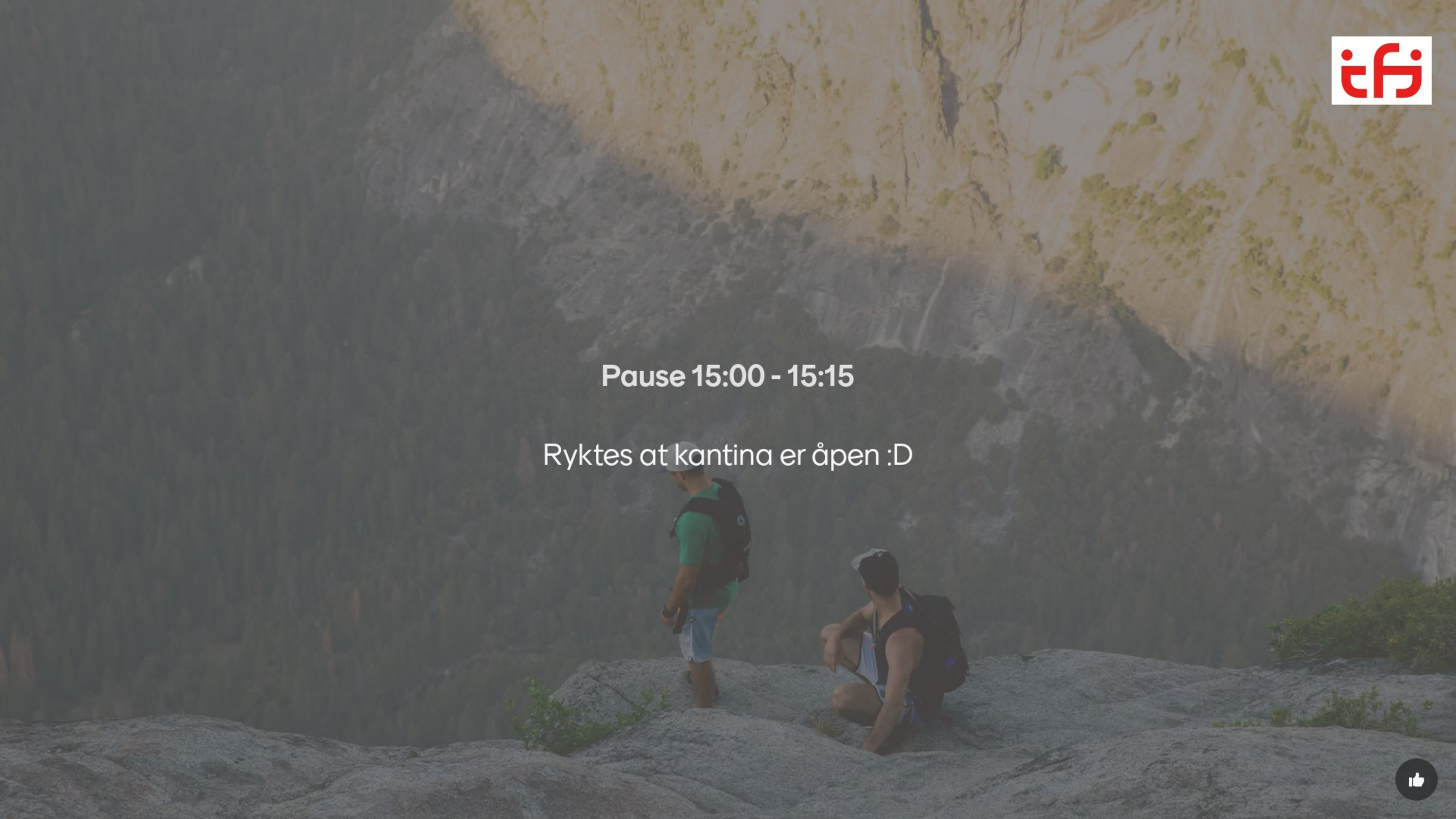
- Ble flyttet fra i går
- Annen tid og annet sted:
- 2024-09-20 kl 10:15 @ Prolog
- (Ingen forelesning på mandag 23.09 heller)

# Resten av tiden (til 16): Oblighjelp

Neste gang:

Live-progge oblig A? Må se an rettingen.

Spørsmål? Mattermost, mail, brevdue: oliverrj@ifi.uio.no


Pause 15:00 - 15:15
Ryktes at kantina er åpen :D