Query Auto-Completion (QAC)

Alfan Farizki Wicaksono
Fakultas Ilmu Komputer, Universitas Indonesia

anak ui

anak ui

anak ui **tertabrak kereta**

anak ui**n**

anak ui **meninggal**

anak ui **sombong**

anak ui **pintar**

anak ui **hedon**

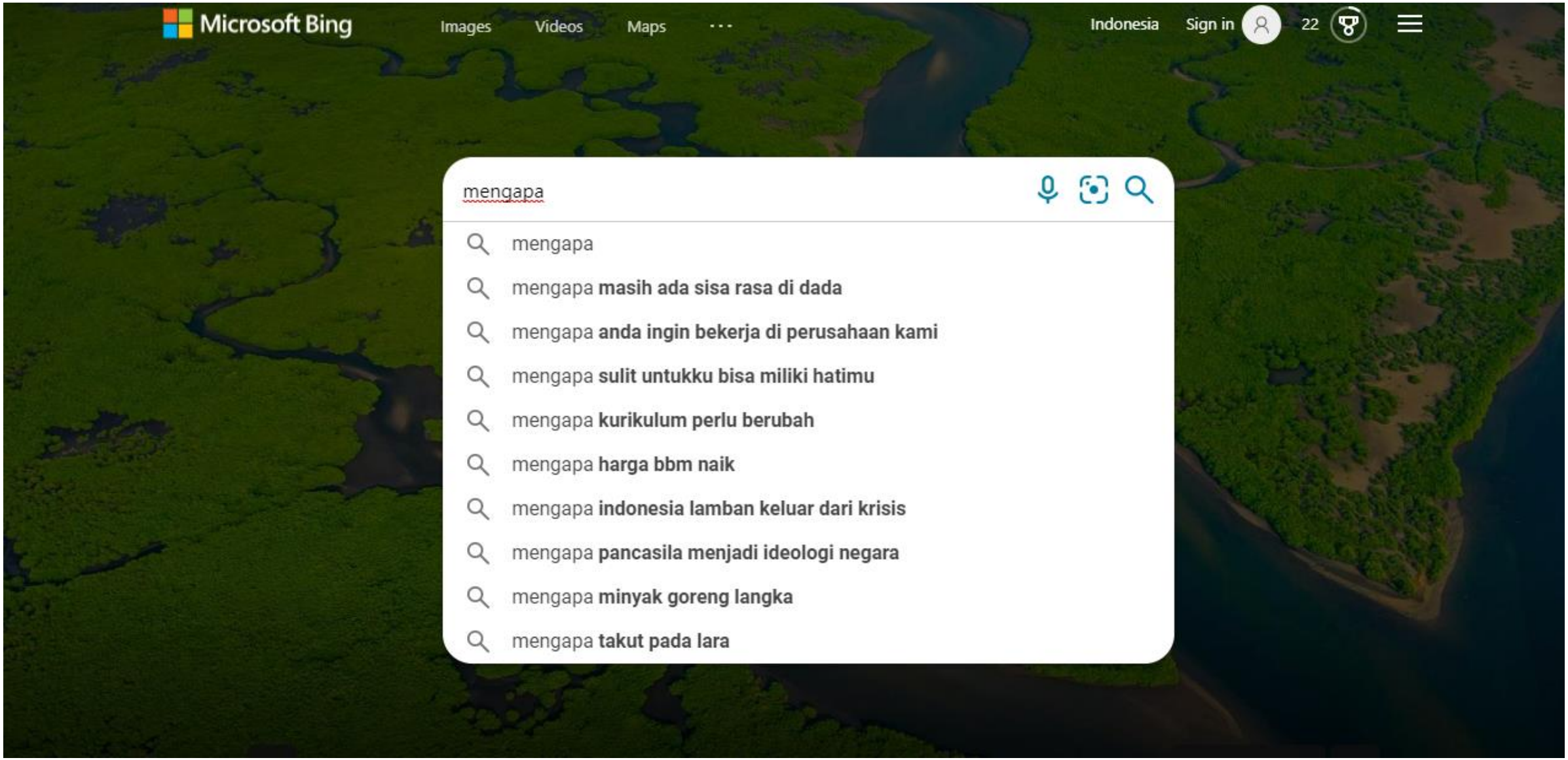
anak ui**r gantung diri**

anak ui **gaji 8 juta**

anak ui**n viral**

Penelusuran Google

Saya Lagi Beruntung


Microsoft Bing
Images
Videos
Maps
Indonesia
Sign in
22
mengapa
mengapa
mengapa masih ada sisa rasa di dada
mengapa anda ingin bekerja di perusahaan kami
mengapa sulit untukku bisa miliki hatimu
mengapa kurikulum perlu berubah
mengapa harga bbm naik
mengapa indonesia lamban keluar dari krisis
mengapa pancasila menjadi ideologi negara
mengapa minyak goreng langka
mengapa takut pada lara

Mengapa Query Auto-Completion?

- Membantu user dalam formulasi query
- Mengurangi keystrokes yang dibutuhkan untuk input query
- Membantu koreksi spelling error
- Cache results! Reduce server load

High Level Algorithm

Given a query pattern **P**

- Retrieve set of candidates "matching" **P** from set **S** of possible target queries.
- Rank candidates by frequency.
- Possibly re-rank highest ranked candidates with more complex ranking
- Return the top-K highest ranking candidates as suggestions

QAC Modes

1. Prefix match
2. Substring match
3. Multi-term prefix match
4. Relaxed match

Contoh: Target -> **"FIFA world cup 2022"**

	Mode 1	Mode 2	Mode 3	Mode 4
FIFA wo	x	x	x	x
orl		x		
FI wor			x	x
FIFO walrd cu				x

Unni Krishnan, Alistair Moffat, Justin Zobel: A Taxonomy of Query Auto Completion Modes. ADCS 2017: 6:1-6:8

Prefix Completion

Prefix Completion

- Task: given a query prefix P, retrieve the top-K most frequent completions.
- Data: **search query logs** yang sudah dikumpulkan lama
- Requirements:
  - Efficient retrieval time required
  - Space efficient index

Matthias Petri, U. of Melbourne, Slide Kuliah Web Search & Text Analysis

Tahap 1

Preprocess data by sorting query log in lexicographical order and based on frequency of unique queries.

bunnings
bachelor in paradise
bunnings
bbc news
bunnings
bbc news
bbc news
big w
big w
bunnings

<bunnings, 4>
<big w, 2>
<bbc news, 3>
<bachelor in paradise, 1>

Tahap 2 – Trie + Frequency Table

Insert all unique queries and their frequencies into a trie data structure

Trie Data Structure:

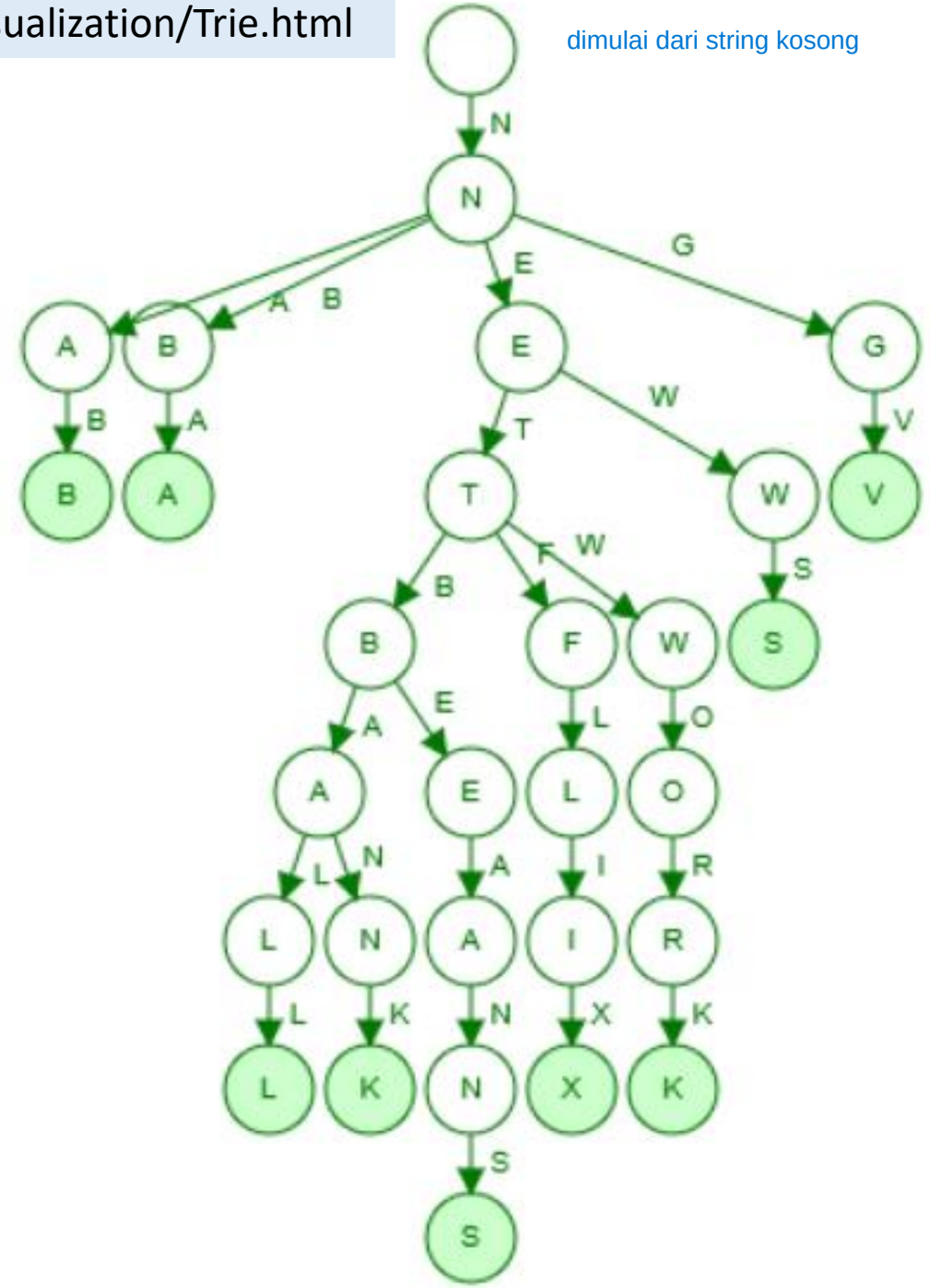
- A tree data structure that contains a set of strings
- Edges of the tree are labeled
- **Children** of nodes are **ordered**
- A **path from root to a node** represents a **prefix** of all strings in the subtree starting at that node

https://www.cs.usfca.edu/~galles/visualization/Trie.html

Trie Visualization

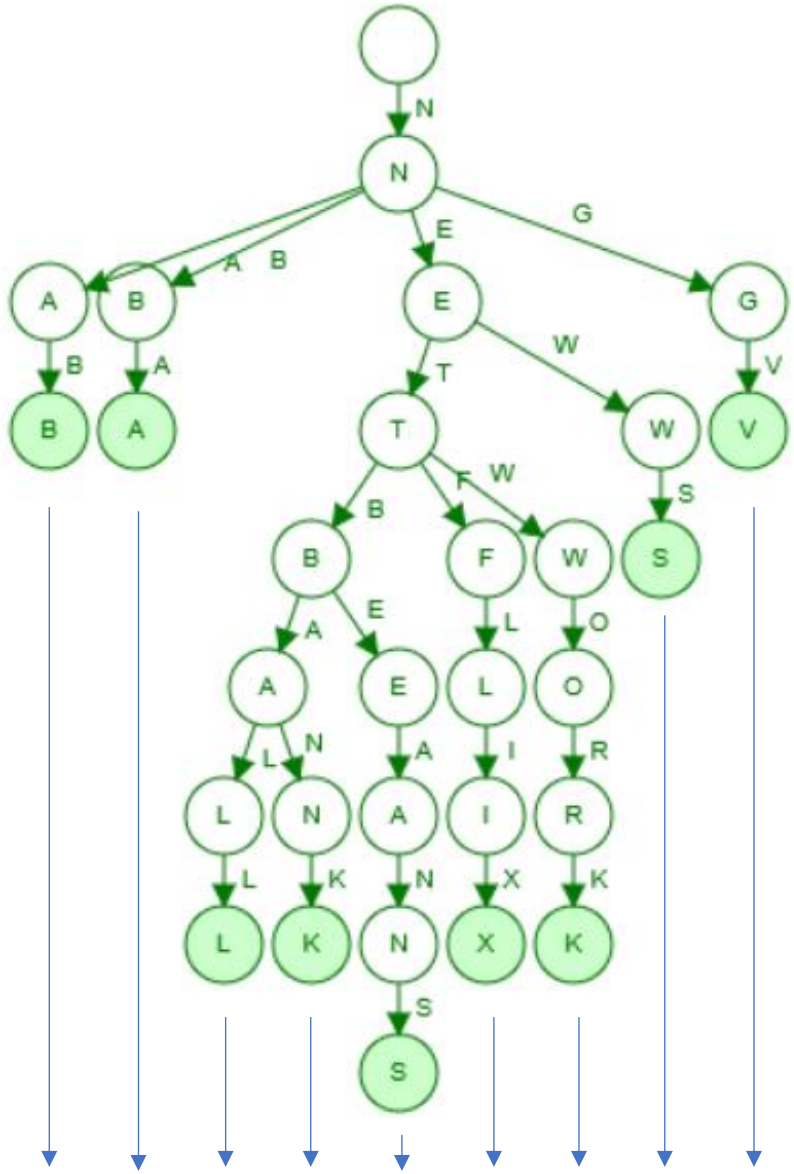
Set of strings

<nba, 6>
<news, 6>
<nab, 8>
<ngv, 9>
<netflix, 7>
<netbank, 8>
<network, 1>
<netball, 3>
<netbeans, 4>



Matthias Petri, U. of Melbourne, Slide Kuliah Web

Trie

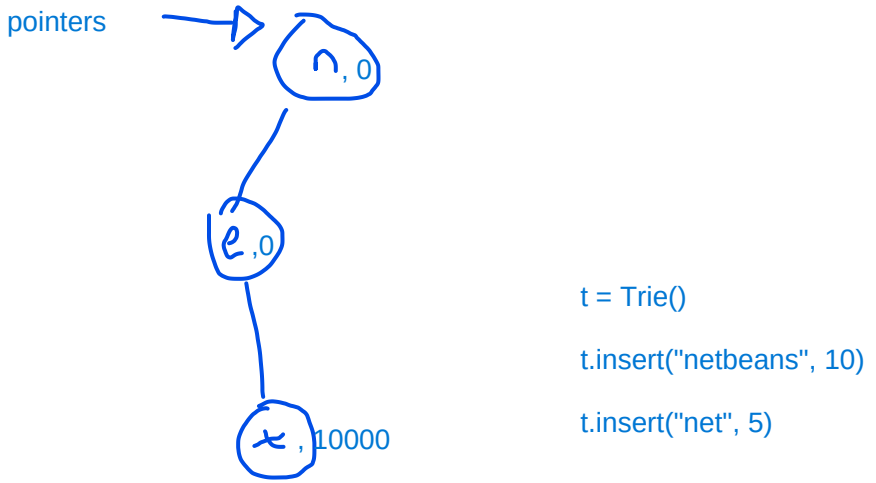


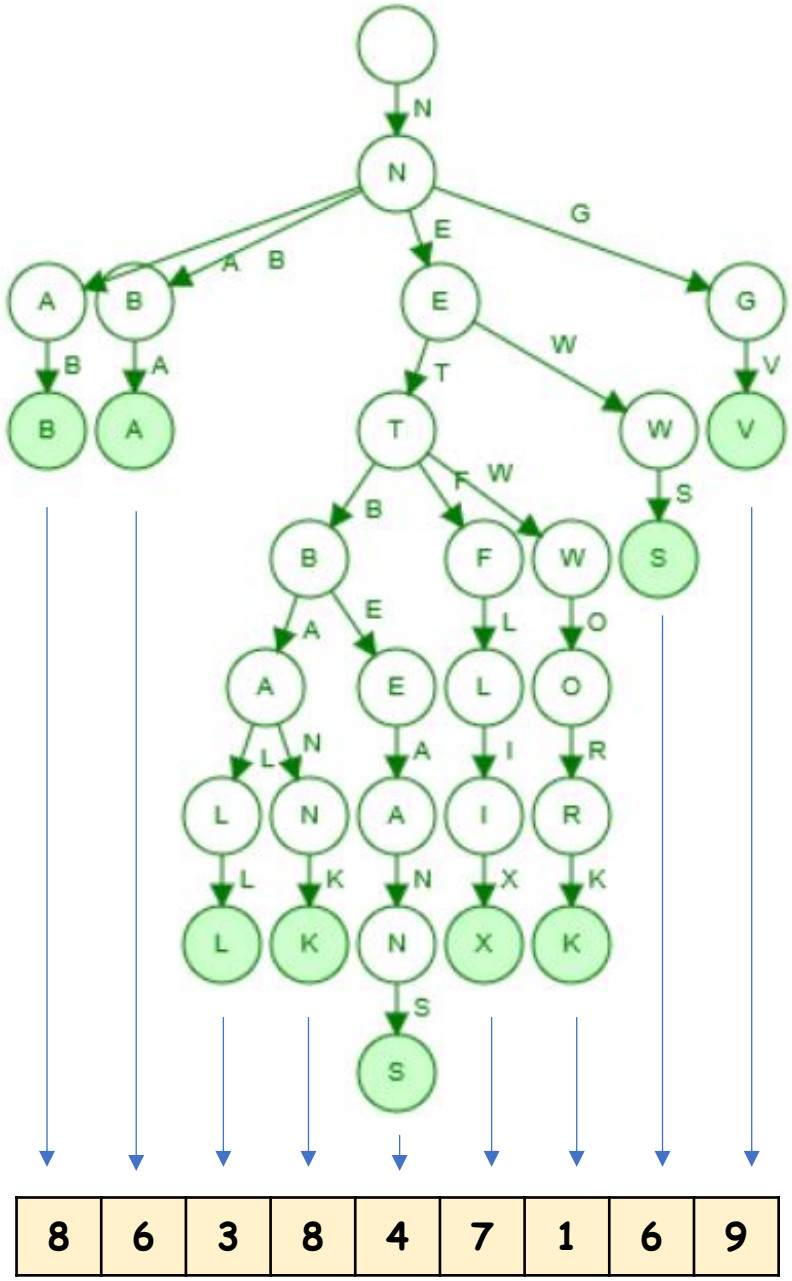
Frequency Table

Implementasi Trie dengan Python

```
class TrieNode:
    """Sebuah Node di struktur data Trie"""

    def __init__(self, char):
        self.char = char # karakter yang disimpan di node
        self.freq = 0    # frekuensi prefix dari
                         # root hingga karakter/node ini

        # keys = characters, values = nodes (anak-anak)
        self.children = {}
```

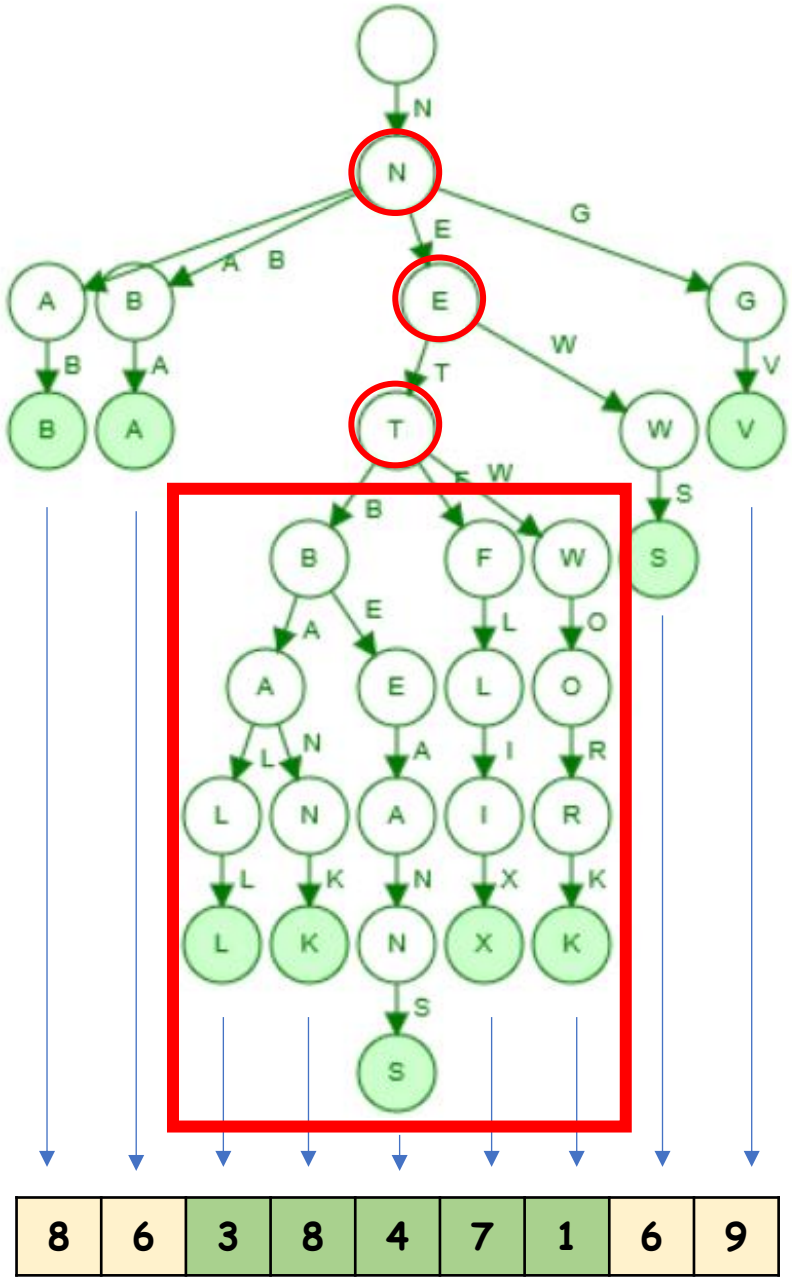
```
class Trie(object):
    """Abstraksi struktur data Trie"""
    def __init__(self):
        self.root = TrieNode("")

    def insert(self, word, freq):
        """tambahkan sebuah term pada Trie"""
        node = self.root
        # loop ke setiap karakter di word
        for char in word:
            # jika ada di salah satu anak
            # langsung node pindah ke anak tersebut
            if char in node.children:
                node = node.children[char]
            else:
                # jika tidak ditemukan, buat node baru
                new_node = TrieNode(char)
                node.children[char] = new_node
                node = new_node
        node.freq += freq
```




N
N
A
B
E
G
A
B
E
G
B
A
T
W
V
B
A
T
W
V
F
W
S
B
F
W
S
E
A
L
O
A
E
L
O
L
N
A
I
R
L
N
A
I
R
L
K
N
X
K
L
K
N
X
K
S
S
8
6
3
8
4
7
1
6
9

net

search



Rekomendasikan Top-K queries, K = 4

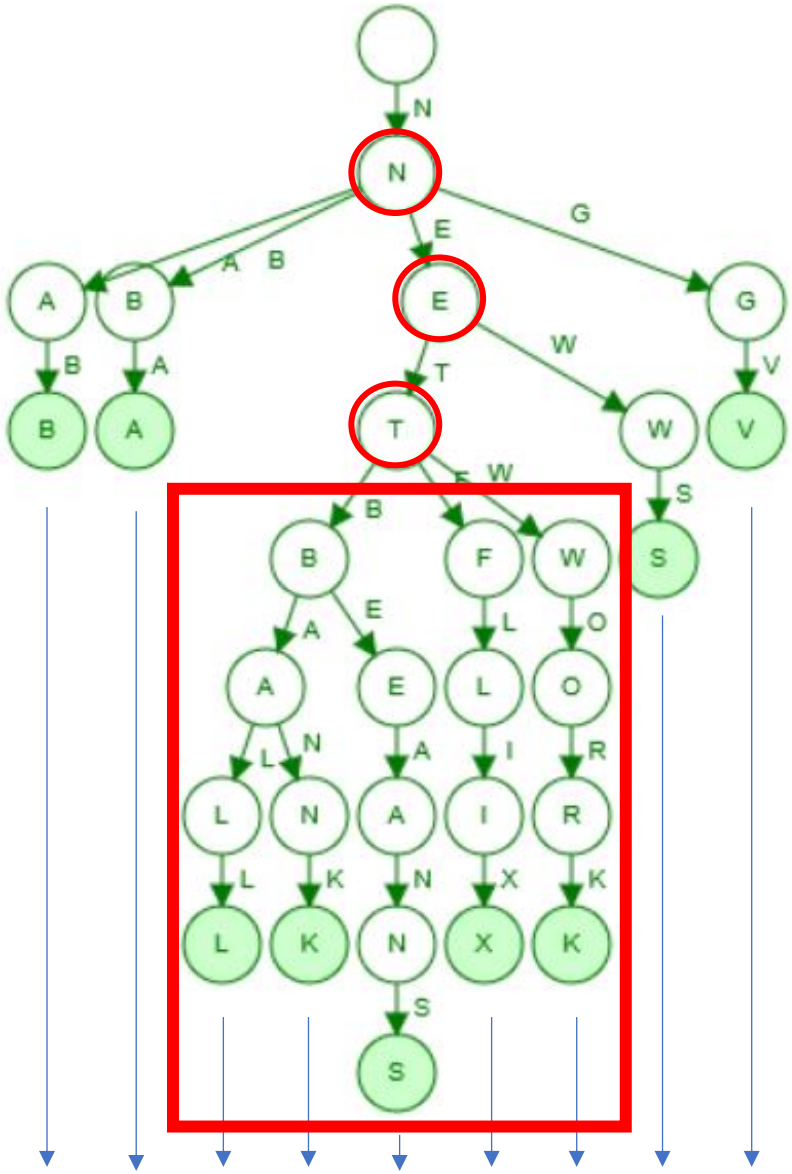
net

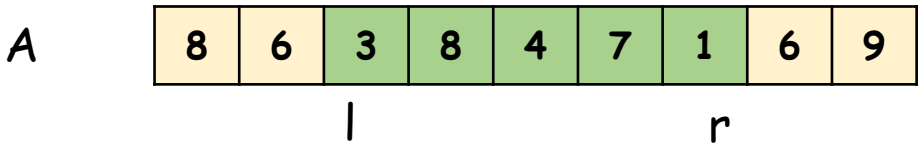
search

netbank
netflix
netbeans
netball

Untuk sebuah pattern **P**, pencarian node pada **Trie** yang merepresentasikan subtree dengan prefix **P** dapat dilakukan dengan kompleksitas **O(|P|)**

Range Maximum Queries (RMQ)



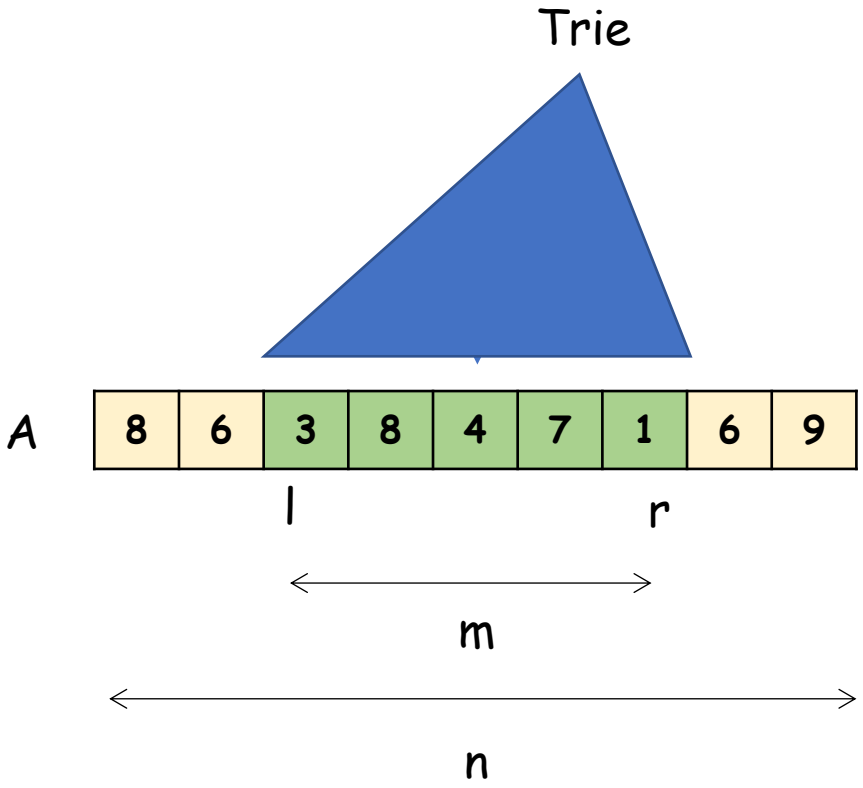


net

search

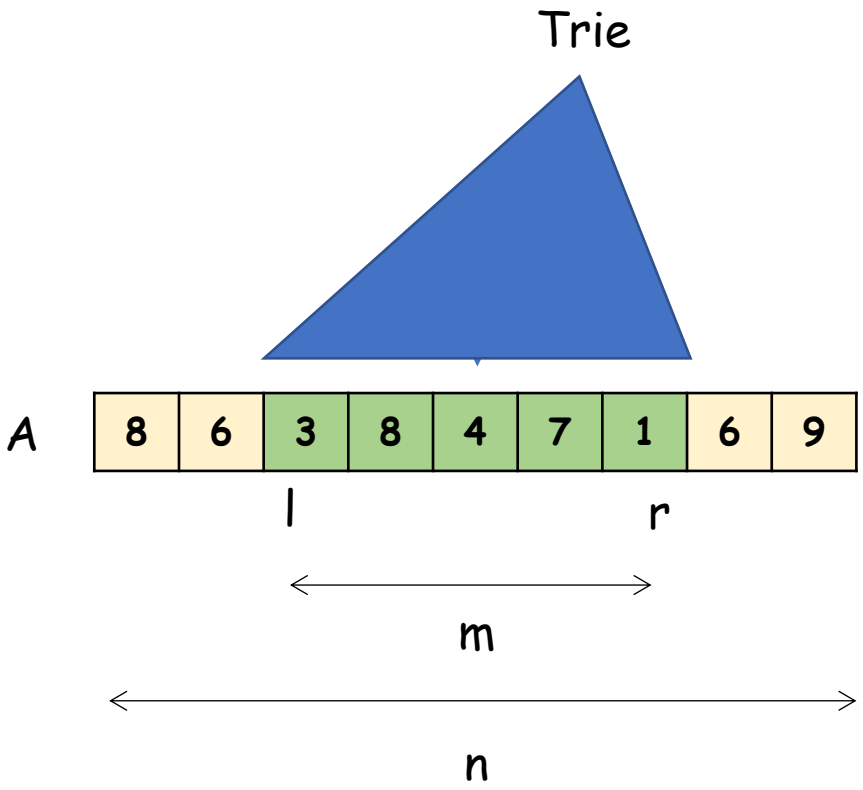
netbank
netflix
netbeans
netball

Task: Given an array **A** of **n** numbers, and a range **[l, r]** of size **m**, find the positions of the **K** largest numbers in **A[l, r]**



Task: Given an array **A** of **n** numbers, and a range **[l, r]** of size **m**, find the positions of the **K** largest numbers in **A[l, r]**

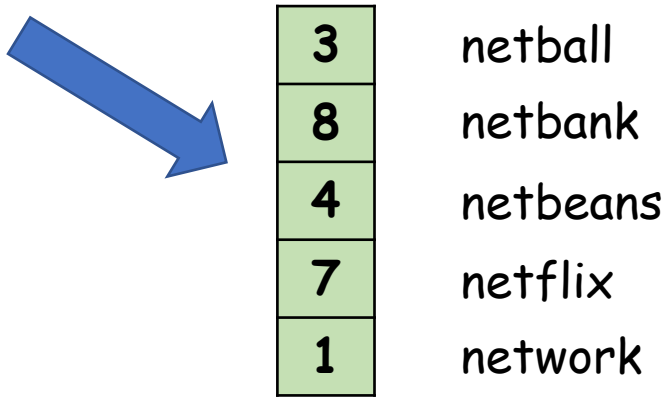
Algoritma Sederhana:

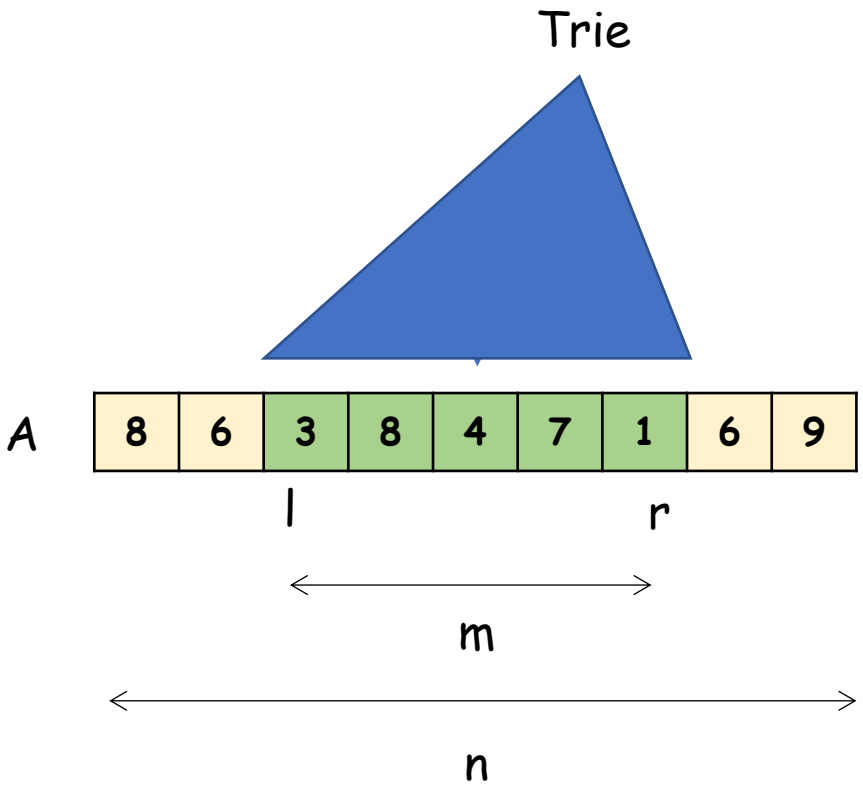


Task: Given an array **A** of **n** numbers, and a range **[l, r]** of size **m**, find the positions of the **K** largest numbers in **A[l, r]**

Algoritma Sederhana:

1. Copy **A[l, r]** ke array baru **B** dalam waktu **O(m)**

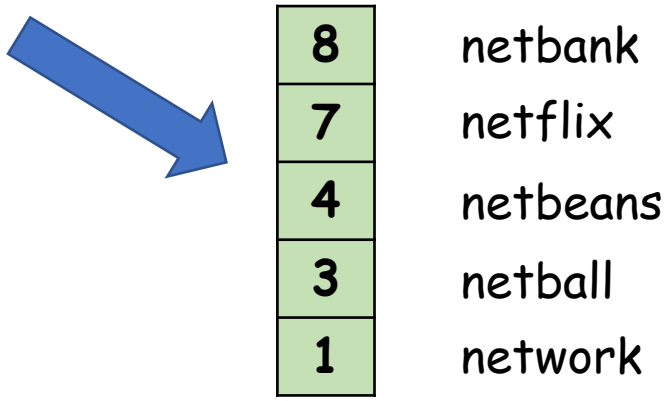


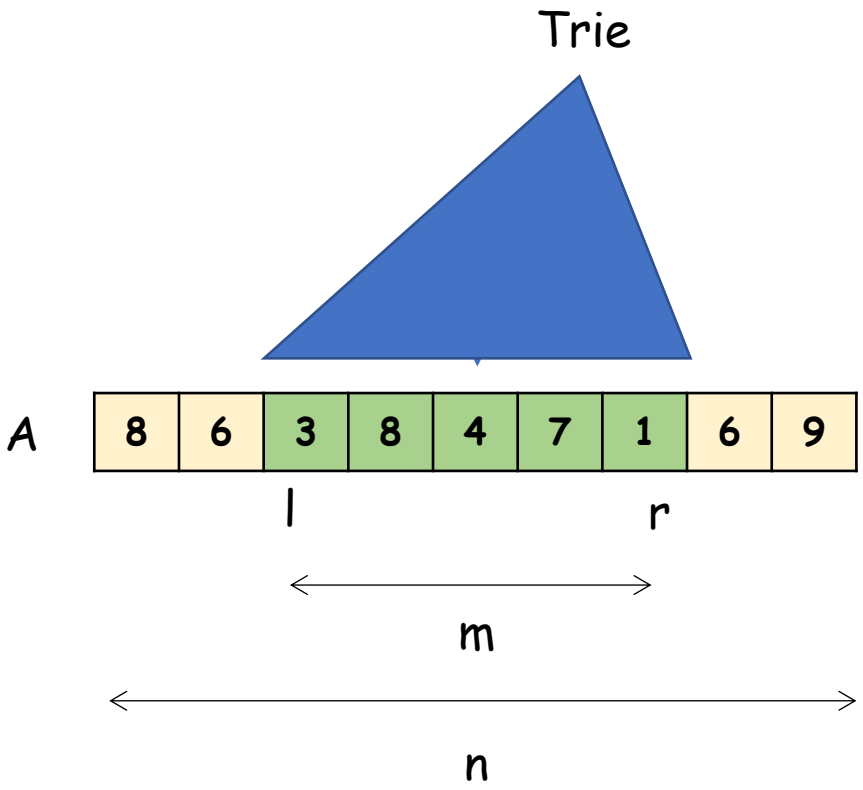


Task: Given an array **A** of **n** numbers, and a range **[l, r]** of size **m**, find the positions of the **K** largest numbers in **A[l, r]**

Algoritma Sederhana:

1. Copy **A[l, r]** ke array baru **B** dalam waktu **O(m)**
2. Sort **B** dalam waktu **O(m log m)**

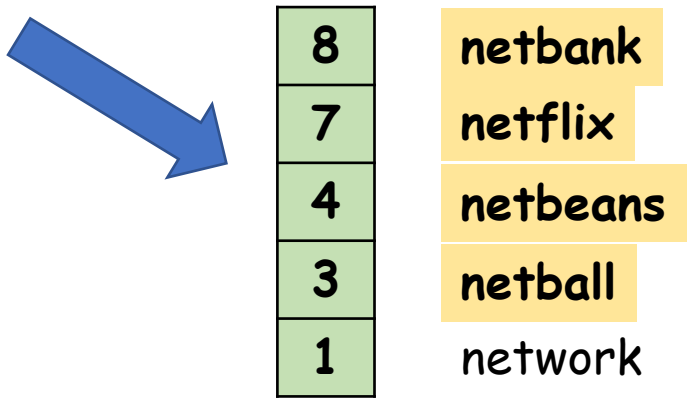




Task: Given an array **A** of **n** numbers, and a range **[l, r]** of size **m**, find the positions of the **K** largest numbers in **A[l, r]**

Algoritma Sederhana:

1. Copy **A[l, r]** ke array baru **B** dalam waktu **O(m)**
2. Sort **B** dalam waktu **O(m log m)**
3. Kembalikan posisi **K** angka terbesar di **A[l, r]**



Masalah

- m bisa besar!
- Perlu O(m) space

Range Maximum Queries - Index

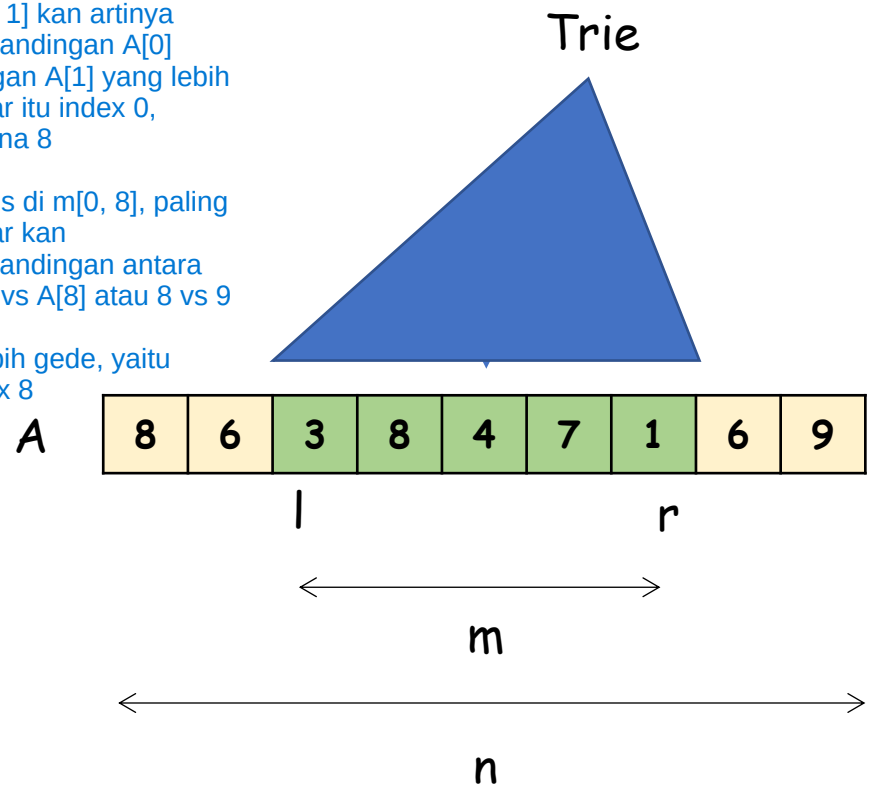
Precompute matrix berukuran **n x n**, dimana **M[i,j]** berisi posisi dimana elemen terbesar pada lokasi array **A[i,j]** berada.

m[0, 1] artinya lokasi pada array A yang paling maksimal adalah 0

karena A[0] itu 8, terus m[0, 1] kan artinya perbandingan A[0] dengan A[1] yang lebih besar itu index 0, karena 8

Terus di m[0, 8], paling besar kan perbandingan antara A[0] vs A[8] atau 8 vs 9

9 lebih gede, yaitu index 8



	0	1	2	3	4	5	6	7	8
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**8**
1		**1**	**1**	**3**	**3**	**3**	**3**	**3**	**8**
2			**2**	**3**	**3**	**3**	**3**	**3**	**8**
3				**3**	**3**	**3**	**3**	**3**	**8**
4					**4**	**5**	**5**	**5**	**8**
5						**5**	**5**	**5**	**8**
6							**6**	**7**	**8**
7								**7**	**8**
8									**8**

Range Maximum Queries - Index

Dengan index, pencarian nilai maksimum pada range **A[i,j]** menjadi **O(1)**.

Pertama ambil A[2, 7], yang baling gede 8, diman ada di index 3 (0 based index)

2.
- A[2, 2]
- A[4, 7]

di cari maksimum nya masing-masing.

Jadi, sekarang bagaimana mencari **K** nilai terbesar?

- Cari posisi nilai terbesar **p** pada **A[i,j]**
- Proses ke **A[i,p-1]** dan **A[p+1,j]**
- Lakukan terus hingga mendapatkan **K** elemen terbesar

O(K log m)

Ada yang bisa buat pseudo-codenya?
*** hint: pakai struktur data heap**