Penerapan Pohon N-ary Dalam Fitur *Auto-complete* Pada Search Engine

Henry Anand Septian Radityo - 13521004
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521004@std.stei.itb.ac.id

***Abstract*—Fitur Auto-complete merupakan fitur yang biasa ada di dalam web browser ketika pengguna sedang mencari sesuatu pada kolom pencari. Fitur ini mendeteksi apa saja kemungkinan kata ketika seorang pengguna memasukkan sedang memasukkan sesuatu dan menampilkannya dalam bahasa tertentu sesuai dengan masukan pengguna. Fitur ini sangat berguna untuk mempercepat pengguna dalam mencari informasi di internet. Fitur Auto-complete bekerja dengan menggunakan konsep dari n-ary tree untuk dapat menampilkan kemungkinan – kemungkinan masukan dari kamus yang tersedia dalan bentuk pohon n-ary.**

***Keywords*—Auto-complete, N-ary Tree, web browser, internet**

I. PENDAHULUAN

Dewasa ini, pengguna internet di Indonesia telah mencapai 200 juta orang. Tren peningkatan jumlah pengguna internet terdeteksi mengalami peningkatan pesat selama lima tahun terakhir. Salah satu faktor yang mendasarinya adalah pandemi Covid-19 yang mengharuskan masyarakat untuk bekerja dan belajar dari rumah. Aturan *work from home* menjadikan membuat masyarakat harus mengenal dan menggunakan internet untuk melakukan pekerjaannya.

Figur 1 Pertumbuhan Jumlah Pengguna Internet Indonesia Tahun 2018 – 2022
(*sumber : https://databoks.katadata.co.id/datapublish/2022/03/23/ada-2047-juta-pengguna-internet-di-indonesia-awal-2022*)

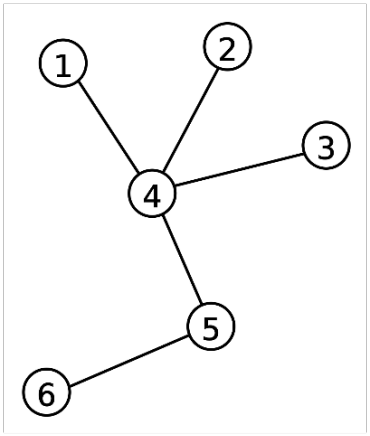
Peningkatan pengguna internet yang pesat juga menyebabkan peningkatan informasi yang tersebar di internet. Banyaknya informasi yang tersebar di internet membuat pengguna yang ingin mencari informasi di internet harus menggunakan mesin pencari tertentu.

Pada saat menggunakan mesin pencari, seringkali ketika sedang memasukkan sesuatu pada kolom pencarian, muncul beberapa kata yang menjadi kemungkinan dari masukan pengguna. Fitur tersebut dapat bekerja dengan didasari oleh struktur data yang memanfaatkan pohon n-ary untuk menampung daftar kata pada suatu bahasa lalu melakukan pencarian kata yang telah dimasukkan pengguna.

II LANDASAN TEORI

A. Pohon

Pohon adalah bagian dari graf, namun pohon tidak memiliki arah dan juga sirkuit. Pohon terdiri dari simpul dan juga sisi yang menghubungkan simpul – simpulnya. Sebuah pohon dapat didefinisikan sebagai T = (V,E) dengan V merupakan himpunan simpul dan E merupakan himpunan sisi yang tiap sisinya terdiri dari dua buah simpul.



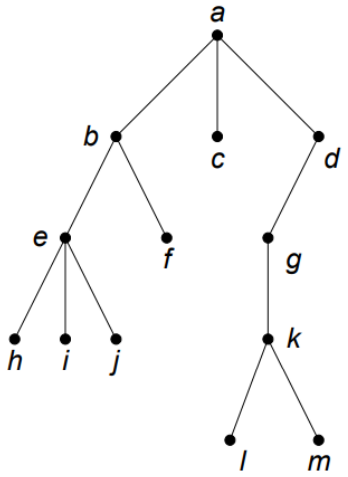
Figur 2 Ilustrasi Pohon
(*sumber : https://upload.wikimedia.org/wikipedia/commons/2/24/Tree_graph.svg*)

Pada pohon setiap pasang simpul dalam pohon terhubung dalam lintasan tunggal. Jumlah dari sisi atau *edge* dapat dihitung dari jumlah simpul sebuah pohon atau dapat didefinisikan sebagai berikut.

$$e = n - 1$$

B. Pohon Berakar (*rooted tree*)

Pohon berakar adalah salah satu bagian dari pohon yang memiliki beberapa simpul yang berperan sebagai akar. Pohon berakar juga memiliki arah pada setiap sisinya yang menghubungkan tiap simpul dengan simpul lain. Pohon berakar juga memiliki syarat tidak memiliki sirkuit dan perhitungan jumlah sisinya dapat dihitung dengan cara yang sama dengan pohon biasa.



Figur 3 Ilustrasi Pohon Berakar
(sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf)

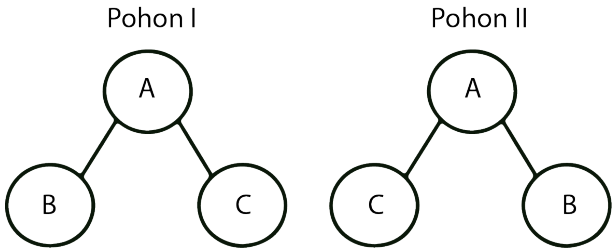
Terdapat beberapa terminologi pada pohon berakar yaitu:

- Anak (*child*)
  Anak merupakan suatu *node* yang ditunjuk oleh *node* lain. Apabila suatu *node* A ditunjuk oleh node B maka A merupakan anak dari B.
- Orang tua (*parent*)
  Simpul A dikatakan *parent* dari simpul B apabila B merupakan anak dari A. Selain itu, dapat dikatakan pula A merupakan simpul yang terhubung langsung dengan B dan memiliki tingkat yang lebih tinggi dari B.
- Lintasan (*path*)
  Lintasan atau *path* adalah sebuah jalur yang dapat dilewati dari satu simpul menuju simpul lain.
- Saudara kandung (*sibling*)
  Dua simpul dikatakan saudara kandung atau *sibling* apabila simpul tersebut memiliki orang tua yang sama.
- Upapohon (*subtree*)
  Upapohon atau sub pohon merupakan bagian dari suatu pohon.
- Derajat (*degree*)
  Derajat adalah jumlah simpul anak atau jumlah upapohon yang ada pada suatu simpul.
- Daun (*leaf*)
  Daun adalah suatu simpul yang tidak memiliki anak atau child.
- Simpul Dalam (*internal nodes*)
  Simpul dalam merupakan kebalikan dari daun atau simpul yang memiliki anak.
- Tingkat (*level*)
  Tingkat atau *level* adalah jumlah sisi untuk menghubungkan akar ke suatu simpul. Tingkatan dari suatu pohon biasanya dimulai dari 0 yang terdapat pada akar pohon.
- Kedalaman (*depth*)
  Kedalaman atau *depth* adalah jumlah tingkat terbanyak yang ada di suatu pohon. Terkadang kedalaman juga disebut dengan tinggi atau *height*.

Pohon berakar juga dibedakan menjadi beberapa jenis yaitu:

1. Pohon Terurut

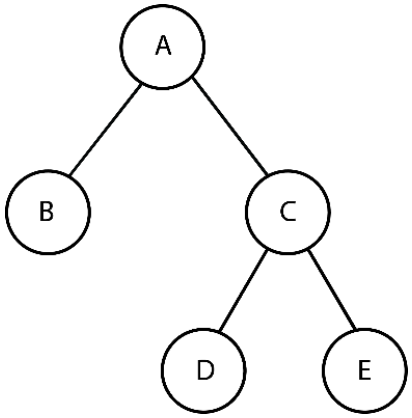
   Pohon terurut atau ordered tree merupakan pohon yang memperhatikan urutan antar anak. Pada figur 4 pohon I dan pohon II dikatakan berbeda dikarenakan urutan dari kedua anak di pohon tersebut berbeda. Pohon terurut dapat digunakan untuk merepresentasikan silsilah keluarga dimana urutan dari tiap anak diperhatikan.



Figur 4 Ilustrasi pohon terurut
(sumber : Dokumen Penulis)

2. Pohon Biner (*binary*)

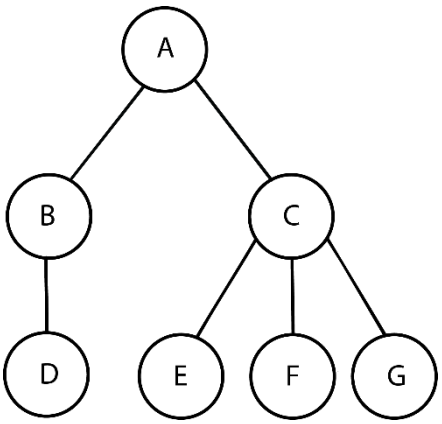
   Pohon biner merupakan pohon yang jumlah derajat maksimum dari tiap simpulnya dua. Pada pohon biner biasanya anak dari simpul disebut sebagai *left* (kiri) dan *right* (kanan).



Figur 5 Ilustrasi pohon biner
(sumber : Dokumen Penulis)

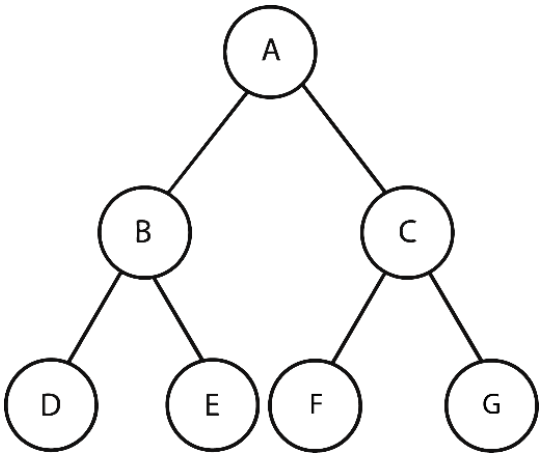
3. Pohon N-ary

   Pohon n-ary merupakan pohon yang memiliki derajat maksimum sebanyak n dengan n merupakan bilangan bulat lebih dari nol. Pohon n-ary dapat dikatakan teratur apabila tiap simpulnya memiliki tepat n buah anak.



Figur 6 Ilustrasi pohon n-ary
(sumber : Dokumen Penulis)

4. Pohon Biner Seimbang
   Pohon seimbang adalah pohon yang tinggi dari upapohon kiri dan juga kanan memiliki selisih maksimal satu tingkat.



Figur 7 Ilustrasi Pohon Biner Seimbang
(sumber : Dokumen Penulis)

C. Algoritma pencari dalam pohon

Pada pohon dengan derajat maksimum berapapun, pencarian dari sebuh pohon selalu dimulai dari simpul akar, kemudian pencarian akan dilanjutkan pada level dibawahnya atau anak dari simpul akar. Algoritma ini akan terus berlanjut hingga sampai di simpul daun atau ketika pencarian tidak ditemukan.

D. Internet

Internet adalah jaringan besar yang terhubung dan dapat saling berkomunikasi antara satu dengan lainnya. Internet dapat menghubungkan tiap perangkat yang dibantu oleh satelit dan juga alat telekomunikasi lain. Selain itu, internet juga dibentuk oleh jutaan orang di seluruh dunia dan menjadi sarana untuk bertukar informasi melalui berbagai media seperti teks, gambar, video, audio, dan lainnya.

Figur 8 Ilustrasi Internet
(sumber : https://www.jojonomic.com/blog/pengertian-internet/)

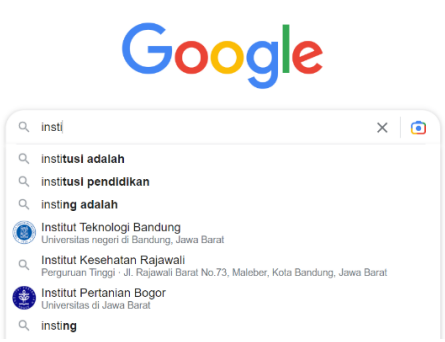
E. Mesin Pencari (*Search Engine*)

Mesin pencari atau search engine adalah suatu perangkat lunak yang dapat digunakan untuk mencari berkas atau layanan yang tersedia di internet. Hasil dari pencarian umumnya ditampilkan dalam bentuk daftar dan diurutkan menurut kerelevanannya terhadap kata kunci. Informasi yang diberikan oleh mesin pencari dapat berupa teks, gambar, audio, atau video. Terdapat mesin pencari yang populer seperti safari, google, yahoo, dan bing.

Figur 9 Mesin Pencari Populer
(sumber : https://www.webhopers.com/what-is-search-engine-its-types)

F. *Auto Complete*

Auto-complete merupakan fitur yang biasa tersedia di dalam search engine yang digunakan untuk memudahkan pengguna dalam memasukkan kata kunci pada kolom pencari. Fitur ini mendeteksi kemungkinan kata menurut bahasa yang dimasukkan oleh pengguna dan mencari daftar kata yang sesuai dengan masukan pengguna. Database atau kumpulan data dari kata – kata di suatu bahasa dapat dibentuk dalam struktur data *n-ary tree* yang nantinya dapat dicari dengan menggunakan algoritma pencari.



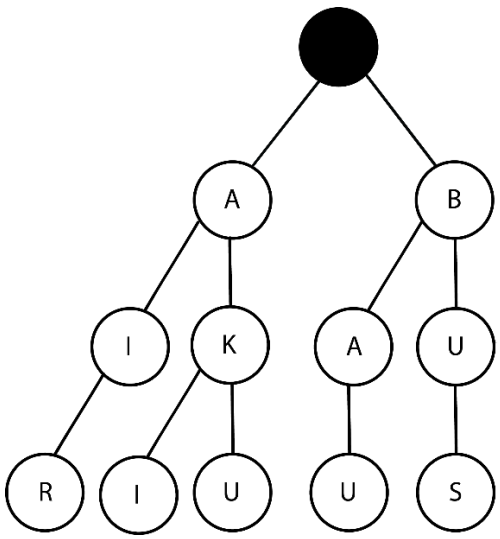
Figur 10 Auto-complete dalam search engine google
(sumber : Dokumen Penulis)

III. Analisis Dan Pembahasan

Pada fitur *auto-complete* membutuhkan suatu kamus yang sudah berbentuk pohon n-ary sehingga mudah untuk dilakukan pencarian dengan menggunakan algoritma pencari, terdapat beberapa tahapan yang harus dibuat dalam penerapannya. Pertama, pembuatan pohon n-ary dari kumpulan kata yang ada. Kemudian dilakukan pencarian menggunakan algoritma pencari dari pohon, sampai ditemukan simpul yang sesuai. Apabila simpul yang sesuai sudah ditemukan maka simpul – simpul di bawahnya akan dimunculkan sebagai rekomendasi dari kata – kata yang mungkin.

A. Pembentukan Pohon n-ary dari kamus

Dalam membentuk stuktur data pohon n-ary yang dapat mewakili suatu kumpulan kata, maka setiap simpul dari pohon dapat digunakan untuk menyimpan sebuah karakter. Sebuah kata akan dibagi menjadi beberapa simpul sesuai dengan panjang kata tersebut, lalu karakter yang muncul pertama akan menjadi orang tua dari karakter kedua, kemudian karakter kedua akan menjadi orang tua dari karakter ketiga, pembentukan pohon ini terus berlanjut hingga terbentuk suatu pohon yang merepresentasikan suatu himpunan kata.



Figur 11 Ilustrasi pohon n-ary untuk menyimpan kata
(sumber : Dokumen Penulis)

Pohon n-ary pada figur 9 merupakan suatu contoh pohon yang digunakan untuk merepresentasikan himpunan kata {“air”,”aki”,”aku”,”bau”,”bus”}. Kata – kata yang memiliki awalan sama akan diwakili oleh satu simpul, seperti “aku” dan “aki” yang memiliki awalan “ak” akan dibentuk dua simpul pertama dan hanya memiliki perbedaan di simpul daun saja.

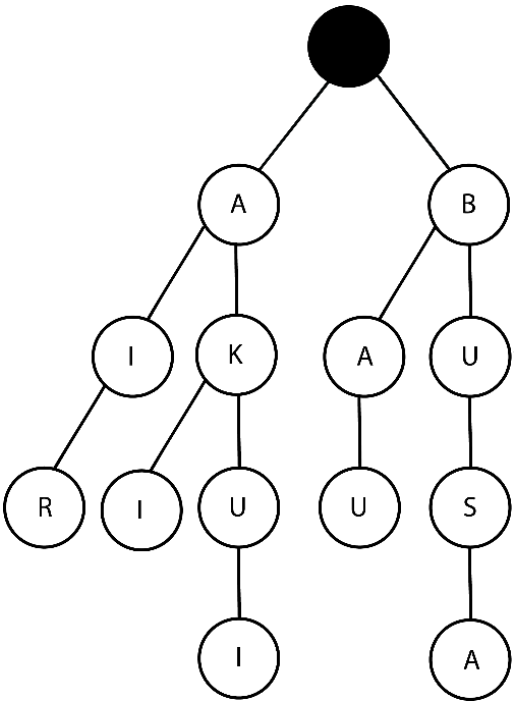
Dalam pembentukan pohon n-ary untuk merepresentasikan himpunan kata, dibutuhkan suatu simpul yang digunakan sebagai akar. Simpul ini nantinya akan berperan dalam memulai pencarian sebagai simpul awal saat dilakukan algoritma pencarian kata.

B. Mitigasi kata rancu

Pohon n-ary yang terbentuk di figur 9 masih memiliki beberapa kekurangan, yaitu terdapat kemungkinan untuk menemukan sesuatu yang bukan kata. Sebagai ilustrasi, pada figur 10 terdapat kata yang berbeda namun memiliki awalan yang sama dan hal ini membuat sebuah kata menjadi rancu apabila tidak memiliki penanda yang menjadi tanda sebuah kata telah berakhir.

Pada kata “aku” dan “akui” juga “bus” dan “busa” program akan kesulitan untuk menampilkan rekomendasi kata “aku” dan “bus” karena kedua kata tersebut tidak berakhir di simpul daun atau simpul terakhir.

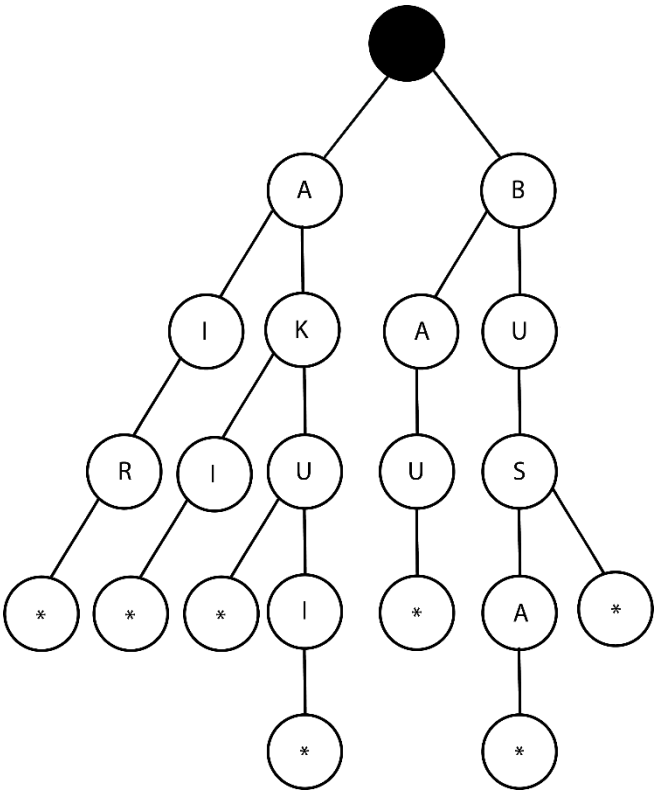
Pohon n-ary pada figur 10 juga memiliki kekurangan lain pada saat melakukan pencarian, dikarenakan tidak adanya penanda membuat algoritma sulit untuk membedakan kumpulan karakter yang ada di kamus dan tidak ada di kamus. Oleh karena itu penggunaan tanda pada bagian dari *tree* menjadi sangat krusial dalam implementasi *auto-complete*.



Figur 12 Ilustrasi pohon n-ary untuk menyimpan kata
(sumber : Dokumen Penulis)

Dalam mendeteksi akhir dari suatu kata dan sebagai mitigasi

agar suatu kata tidak menjadi rancu maka dibutuhkan suatu tanda yang digunakan untuk mengakhiri suatu kata. Pada kali ini, penulis menggunakan tanda bintang (*asterisk*) sebagai tanda suatu kata berakhir, tanda bintang ini ditambahkan pada waktu pembentukan pohon n-ary dan ditambahkan setiap suatu kata berakhir menjadi suatu simpul daun.



Figur 13 Ilustrasi pohon n-ary untuk menyimpan kata dengan menggunakan asterisk (*) sebagai tanda akhir dari kata
(sumber : Dokumen Penulis)

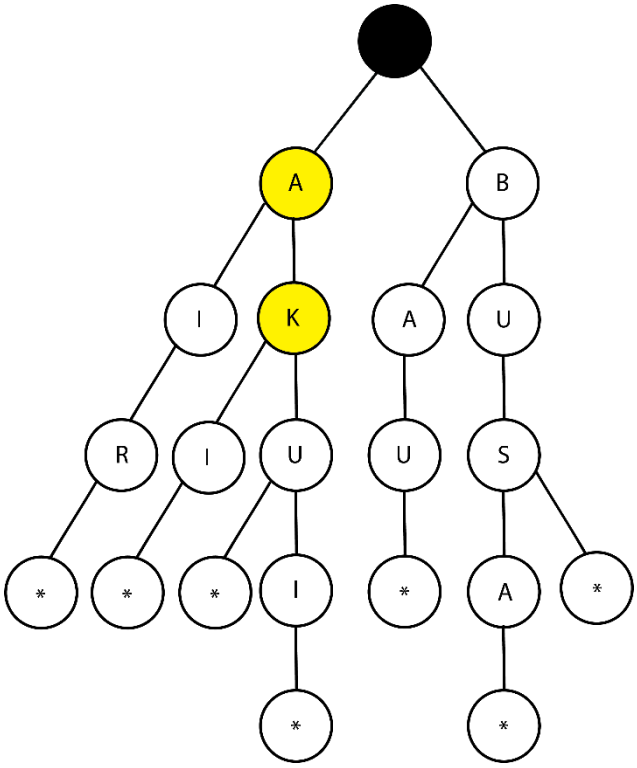
Penambahan simbol *asterisk* juga akan mempermudah algoritma pencarian untuk memunculkan rekomendasi kata, dikarenakan algoritma akan mencari hingga simpul yang berisi tanda asterisk lalu memunculkannya.

C. Algoritma Pencari

Setelah sebuah struktur data dari kamus selesai dibentuk, dibutuhkan suatu algoritma untuk mencari apakah terdapat kata yang sesuai pada *tree* atau tidak. Apabila terdapat kata yang sesuai maka akan menampilkan semua kata yang menjadi kemungkinan dari masukan pengguna dan apabila tidak maka tidak ada yang dimunculkan.

Terdapat beberapa langkah yang digunakan algoritma pencari dalam menemukan suatu kata diantaranya :

1. Mencari karakter pertama dari kata dimulai dari simpul akar
2. Melakukan iterasi hingga karakter terakhir
3. Apabila terdapat simpul yang sesuai maka memunculkan semua kata yang berada di bawah simpul sesuai, langkah ini melakukan iterasi hingga bertemu simpul yang memiliki tanda asterisk.
4. Apabila tidak terdapat simpul sesuai maka tidak memunculkan apapun.



Figur 14 Ilustrasi Auto-complete untuk kata “ak”
(sumber : Dokumen Penulis)

Pada figur 12, user memasukkan kata “ak” dengan menggunakan algoritma pencari pada n-ary tree, maka akan didapatkan simpul “k” yang telah diberi tanda kuning.

```
def search(self, word):
    #definisikan simpul awal dengan
    simpul akar
    cNode = self.root
    for char in word:
        if cNode.children.get(char):
            #lakukan pencarian pada pohon
            cNode = cNode.children[char]
        else:
            #apabila terdapat node yang
            tidak sesuai
            return None
    return cNode
```

Figur 15 Implementasi algoritma untuk mencari node yang sesuai pada bahasa pemrograman python
(sumber : Wengrow, A Common Sense Guide to Data Structures and Algorithm)

Setelah menemukan simpul yang sesuai, maka dibutuhkan suatu algoritma yang mencari secara rekursif hingga menemukan tanda *asterisk*. Hal ini diperlukan untuk menampikan semua kata yang menjadi kemungkinan masukan pengguna.

```
def dispWord(self, node=None, word="",
words=[]):
```

```
    cNode = node or self.root
    for key, childNode in
cNode.children.items():
        if key == "*":
            words.append(word)
        else:
            self.collectAllWords(cNode,
word + key, words)
    return words
```

Figur 16 Implementasi algoritma untuk himpunan kemungkinan kata pada bahasa pemrograman python *(sumber : Wengrow, A Common Sense Guide to Data Structures and Algorithm)*

Setelah mendapatkan kemungkinan kata yang menjadi masukan pengguna maka program akan menampilkan kata tersebut. Pada figur 12, apabila input dari pengguna adalah "ak" maka himpunan kata yang menjadi kemungkinan masukan pengguna adalah {"aki","aku","akui"}.

Algoritma pencari ini memiliki kompleksitas O(N) dengan N merupakan jumlah karakter pada tiap kata. Algoritma ini juga memiliki *best case* O(1) ketika cNode atau simpul yang bersesuaian merupakan simpul yang diakhiri dengan tanda asterisk atau tidak memiliki anak.

IV. Beberapa Kesalahan Umum

Implementasi dari auto-complete membutuhkan banyak himpunan kata yang terdiri dari berbagai bahasa. Oleh karena itu pembentukan pohon n-ary untuk merepresentasikan kamus akan memakan waktu yang cukup lama dikarenakan banyaknya kata yang menjadi masukan. Selain itu, terdapat kekurangan pada saat menampilkan rekomendasi jika algoritma menemukan banyak kata yang menjadi rekomendasi. Terdapat ruang pengembangan untuk memberikan sebuah peringkat dalam mengurutkan kata – kata yang sering dipanggil oleh pengguna sehingga kata – kata yang muncul di rekomendasi merupakan kata yang umum.

V. Simpulan

Penerapan *auto-complete* pada *search engine* akan sangat membantu pengguna dalam melakukan pencarian dan juga menghemat waktu. Pengguna tinggal memasukkan beberapa awalan dari kata kunci dan algoritma akan melakukan pencarian pada pohon n-ary yang berisi himpunan kata lalu memunculkan kemungkinan kata dari input pengguna. Selain itu, dilihat dari notasi O-besar atau *big-O* dari algoritma pencarian ini juga tergolong cukup sangkil dikarenakan O(N) tergolong dalam kelompok algoritma linier. Kompleksitas linier ini juga dapat membantu penghematan waktu dalam menjalankan algoritma sehingga dapat bekerja secara real-time ketika pengguna sedang memasukkan kata kunci.

VI. Ucapan Terima Kasih

Penulis mengucapkan rasa syukur dan terima kasih kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya makalah ini dapat selesai. Ucapan terima kasih penulis sampaikan kepada orang tua dan keluarga yang telah mendukung penulis dari segi moral maupun material.

Tak lupa penulis juga mengucapkan terima kasih kepada seluruh Bapak dan Ibu dosen pengampu mata kuliah Matematika Diskrit Institut Teknologi Bandung yang senantiasa menjadi pembimbing dalam mengajarkan ilmu – ilmu matematika diskrit sehingga makalah berjudul "Penerapan N-ary Tree Dalam Fitur Auto-complete Pada Search Engine" dapat selesai dengan baik. Penulis juga mengucapkan terima kasih kepada teman – teman yang telah mendukung selama proses pengerjaan makalah ini.

Daftar Pustaka

[1] Munir, Rinaldi. (2016). Pohon. *Matematika Diskrit Revisi Ketujuh*, pp 443-491, Bandung, Indonesia.

[2] Munir, Rinaldi. (2022). Pohon. *Slide Kuliah IF2120 Matematika Diskrit*, Bandung, Indonesia.

[3] Wengrow, Jay. (2020). Tries. *A Common Sense Guide to Data Structures and Algorithm*, pp 305-329 , Raleigh, North Carolina.

[4] Shah, Jinit. *AutoComplete Using Tries*. Diakses pada 3 Desember 2022 dari Medium : https://medium.com/analytics-vidhya/autocomplete-using-tries-42aadd875d72

[5] *Auto-complete feature using Trie*. Diakses pada 3 Desember 2022 dari Geeks for Geeks : https://www.geeksforgeeks.org/auto-complete-feature-using-trie/

[6] *Trie | (Insert and Search),* Diakses pada 3 Desember 2022 dari Geeks for Geeks : https://www.geeksforgeeks.org/trie-insert-and-search/.

[7] *Introduction to Trie – Data Structure and Algorithm Tutorials*. Diakses pada 3 Desember 2022 dari Geeks for Geeks : https://www.geeksforgeeks.org/introduction-to-trie-data-structure-and-algorithm-tutorials/

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2022

Henry Anand Septian Radityo 13521004