Pembangkitan Teka – Teki Silang Menggunakan Algoritma Brute Force

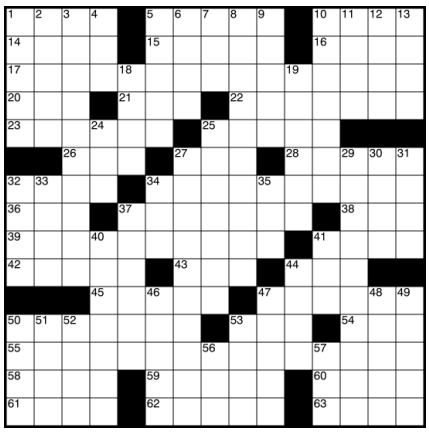
Muhamad Aji Wibisono - 13521095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13521095@std.stei.itb.ac.id

***Abstrak*—Teka – Teki silang atau *crossword puzzle* merupakan permainan yang relatif sederhana. Dengan demikian, jika didekati menggunakan algoritma, permainan teka - teki silang akan dapat dibangkitkan dengan dari kata – kata yang sudah ada. Makalah ini memberikan salah satu algoritma untuk pembuatan permainan tersebut berdasarkan sekumpulan kata yang dimiliki. Selain itu, kualitas teka – teki silang yang dihasilkan juga perlu dianalisis sebagai relevansi algoritma yang digunakan.**

Kata kunci—Teka – Teki Silang; Pembangkitan Soal; Algoritma Brute Force; Heuristik;

I. PENDAHULUAN

Teka – Teki silang atau *crossword puzzle* adalah sebuah teka – teki kata yang umumnya mengambil bentuk kotak atau segi empat yang terdiri atas petak – petak putih dan hitam. Tujuan dari permainan teka teki silang adalah mengisi petak tersebut dengan huruf yang membentuk kata atau kalimat yang saling bersilangan satu sama lain. Pada pengisiannya akan diberikan petunjuk mengenai kata yang terdapat pada rangkaian petak dalam rangkaian mendatar dan menurun dan pemain akan berusaha untuk menerka kata yang dimaksud dengan Panjang yang sesuai. Jika ada petak yang beririsan dengan petak dalam rangkaian lain, maka huruf yang berada pada petak tersebut harus sama dalam kedua jawaban.



Gambar 1.1. Contoh Permainan Teka – Teki Silang
Sumber: https://commons.wikimedia.org/wiki/File:CrosswordUSA.svg
Diakses pada 21 Mei 2022 Pukul 23:24 WIB

Pada pembuatan permainan tersebut terdapat beberapa peraturan yang tidak boleh dilanggar, yaitu tidak boleh ada petak jawaban terhubung yang tidak membentuk kata yang terdapat pada kamus jawaban dan tidak boleh ada rangkaian petak yang tidak berhubungan dengan petak lainnya. Petak – petak yang dibuat tidak boleh terbentuk serangkaian petak mendatar atau menurun yang menghasilkan kata atau huruf acak. Selain itu, jawaban yang dihasilkan perlu saling berkaitan dengan kunci jawaban dari pertanyaan tersebut.

Teka – teki silang di Indonesia dipopulerkan melalui media koran, khususnya harian kompas, dimulai tahun 1968[3]. Saat ini permainan teka – teki silang tersedia pada berbagai media, mulai dari buku khusus teka – teki silang, perangkat lunak, web, dan lain sebagainya. Pembuatan algoritma yang cukup baik untuk membangkitkan teka – teki silang dapat membantu mempercepat dan mempermudah produksi pembuatan teka – teki silang dalam media – media tersebut.

II. DASAR TEORI

1. Algoritma Brute Force

Algoritma Brute Force merupakan pendekatan yang lempang untuk memecahkan suatu persoalan. Algoritma Brute Force didasarkan pada pernyataan pada persoalan dan definisi atau konsep yang dilibatkan. Dalam pemecahan persoalan, Algoritma Brute Force memecahkan persoalan dengan sangat sederhana, langsung, dan cara yang jelas[1]

Contoh – contoh Algoritma Brute Force ada pada penggunaannya dalam memecahkan beberapa permasalahan berdasarkan pernyataan persoalan, seperti:

- Mencari elemen terbesar atau terkecil
- Mencari elemen secara beruntun (*Sequential Search*)

Algoritma Brute Force umumnya tidak cerdas dan tidak mangkus karena membutuhkan volume komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Algoritma Brute Force lebih cocok untuk persoalan yang memiliki ukuran masukan kecil karena sederhana dan kemudahan implementasi dari algoritma. Karena ketidakmangkusannya, Algoritma Brute Force sering digunakan sebagai basis pembanding dengan algoritma lain yang lebih mangkus. Meskipun bukan metode penyelesaian masalah yang mangkus, hampir semua persoalan dapat diselesaikan dengan Algoritma Brute Force dan ada

persoalan yang hanya dapat diselesaikan dengan brute force, seperti mencari elemen terbesar dan terkecil dalam array tidak terurut[1].

Kelebihan dari Algoritma Brute Force adalah bahwa Algoritma Brute Force dapat diterapkan untuk memecahkan hampir sebagian besar masalah (*wide applicability*). Algoritma Brute Force yang sederhana membuat Algoritma Brute Force mudah dimengerti. Selain itu, Algoritma Brute Forcee menghasilkan algoritma yang layak untuk beberapa masalah penting dan menghasilkan algoritma baku untuk tugas – tugas komputasi[1].

Kelemahan dari Algoritma Brute Force adalah bahwa Algoritma Brute Force jarang menghasilkan algoritma yang mangkus. Ketidakmangkusannya menyebabkan operasi menjadi lambat untuk masukan yang berukuran besar sehingga tidak dapat diterima. Untuk beberapa kasus, Algoritma Brute Force sudah digantikan dengan strategi pemecahan masalah lain yang lebih mangkus, konstruktif, dan kreatif dalam penyelesaiannya[1].

Permasalahan pencarian teka – teki silang merupakan salah satu permasalahan yang dapat diselesaikan menggunakan Algoritma Brute Force. Hal ini disebabkan teka – teki silang memiliki pernyataan persoalan yang sederhana, yaitu bahwa setiap petak – petak yang terisikan huruf akan membentuk kata atau pernyataan jika dihubungkan dengan petak lainnya yang tersambung secara mendatar maupun menurun ke salah satu arah tersebut. Mengingat kegunaan utamanya sebagai permainan , teka – teki silang juga tidak menggunakan ukuran masukan sangat besar.

2. *Exhaustive Search*

Exhaustive Search adalah pendekatan Algoritma Brute Force yang khusus mencari solusi dari objek – objek dengan kriteria tertentu. Langkah – Langkah Algoritma Exhaustive Search yaitu mencoba semua kemungkinan solusi yang ada sehingga solusi terbaik secara pasti akan ditemukan. Namun, kelemahan dari Exhaustive Search adalah kompleksitas waktu yang besar sehingga tidak dapat digunakan untuk menyelesaikan masalah dalam jumlah yang besar[3].

Pada pembangkitan teka – teki silang menggunakan Algoritma Brute Force, pendekatan yang sesuai adalah dengan menggunakan Exhaustive Search. Hal ini disebabkan oleh banyaknya kemungkinan solusi yang dihasilkan dari teka – teki silang. Exhaustive search perlu dilakukan untuk mencari titik yang sesuai untuk meletakkan kata selanjutnya pada petak teka – teki silang. Solusi yang diberikan Exhaustive Search pada kasus ini adalah suatu kata tambahan terhubung dengan satu atau banyak kata lainnya dengan tidak terhubung dengan petak lain sedemikian sehingga setiap petak – petak yang terisikan huruf akan membentuk kata atau pernyataan jika dihubungkan dengan petak lainnya yang tersambung secara mendatar maupun menurun ke salah satu arah tersebut.

3. *Teknik Heuristik*

Untuk mempercepat pencarian solusi dengan Exhaustive Search dapat digunakan salah satu teknik, yaitu teknik heuristic. Dalam exhaustive search, teknik heuristic digunakan untuk mengeliminasi beberapa kemungkinan solusi tanpa harus mengeksplorasi seluruh kemungkinan solusi secara penuh[2] .

Heuristik merupakan teknik yang dirancang untuk memecahkan persoalan dengan mengabaikan apakah teknik tersebut terbukti benar secara matematis. Teknik ini menggunakan pendekatan yang tidak formal, misalnya berdasarkan intuitif, terkaan, atau akal sehat. Contoh dari teknik heuristic adalah program antivirus yang menggunakan pola – pola heuristic untuk mengidentifikasi dokumen yang terkena virus atau malware. Teknik heuristic secara matematis tidak dapat dibuktikan, namun sangat berguna pada penyelesaian permasalahan[2].

Pada pembangkitan soal teka – teki silang menggunakan Algoritma Brute Force, teknik heuristic dapat digunakan untuk mengurangi titik yang perlu diperiksa sebagai kandidat tempat perpotongan kata dengan kata yang akan ditambahkan. Selain itu, heuristik juga digunakan untuk mengoptimalisasi kualitas teka – teki silang yang dihasilkan.

4. *Notasi Big-O*

Notasi big-O adalah notasi yang mendeskripsikan perilaku Batasan dari sebuah fungsi ketika argument mengarah pada angka tertentu atau tak hingga. Big-O digunakan untuk mengklasifikasikan algoritma berdasarkan ruang atau waktu yang diperlukan seiring dengan berkembangnya jumlah input. Notasi big-O umumnya hanya memberikan batasan atas dari laju perkembangan suatu fungsi dalam nilai input sangat besar sehingga koefisien pada notasi big-O tidak dipertimbangkan[5]. Notasi Big-O dapat digunakan untuk menganalisis Algoritma Brute Force yang dihasilkan untuk membangkitkan teka – teki silang.

5. *Parameter Kualitas Teka – Teki Silang*

Mengenai parameter kualitas teka – teki silang yang dihasilkan, tidak ada parameter kualitas formal mengenai teka – teki silang pada masyarakat. Namun, untuk menilai algoritma yang digunakan dan teka – teki silang yang dihasilkan, perlu dirumuskan suatu parameter kualitas yang menunjukkan efektivitas algoritma dalam membuat teka – teki silang.

Mengingat umumnya teka – teki silang pada awalnya dipermainkan dalam kertas, lebih baik jika dalam suatu kertas tersebut tersusun atas teka – teki silang yang memuat kata - kata lebih banyak.Dengan demikian, teka – teki silang yang dihasilkan akan dinilai berdasarkan kepadatan kata dibanding ruang yang digunakan. Rumus kepadatan adalah sebagai berikut:

$$\rho = \frac{jumlah\ huruf}{panjang\ total \times lebar\ total} \quad (1)$$

Dari persamaan tersebut didapatkan sebuah nilai representasi kepadatan yang bernilai $0 \leq \rho \leq 1$ dengan kepadatan 0 melambangkan semua petak yang ada dalam permainan adalah petak yang tidak akan terisi huruf, sementara kepadatan 1 melambangkan semua petak yang ada pada permainan adalah petak yang akan terisi huruf.

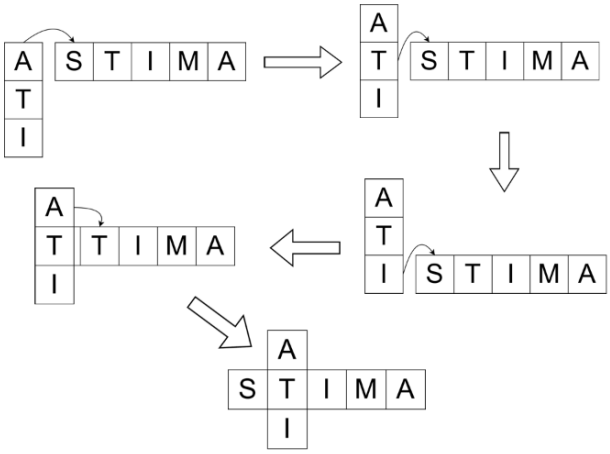
III. PENERAPAN ALGORITMA

1. *Algoritma Solusi*

Permasalahan yang dihadapi pada pembangkitan soal teka – teki silang adalah perihal penambahan kata pada petak. Kata yang ditambahkan perlu diletakkan pada posisi tertentu sedemikian sehingga setiap petak – petak yang terisikan huruf akan membentuk kata atau pernyataan jika dihubungkan dengan petak lainnya yang tersambung secara mendatar maupun menurun ke salah satu arah tersebut. Selain itu diperlukan juga untuk memastikan semua petak yang terhubung memuat kata yang terdapat pada suatu kamus kata yang diberikan dan tidak ada kata asing yang terbentuk secara tidak sengaja.

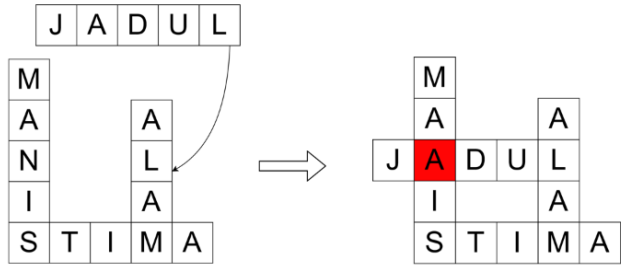
Dalam penyelesaiannya, algoritma membutuhkan sebelumnya sebuah kamus kosakata yang akan digunakan dalam teka – teki silang yang dihasilkan. Setelah itu akan diambil sebuah kata acak sebagai basis dari teka – teki silang yang akan dibuat.

Penambahan kata – kata selanjutnya akan dilakukan dengan membandingkan huruf kata yang belum diletakkan dengan huruf yang yang sudah diletakkan pada petak secara Exhaustive Search. Algoritma akan memeriksa seluruh huruf dari kata yang belum diletakkan dengan seluruh huruf yang telah diletakkan pada petak



Gambar 3.1.1. Ilustrasi persamaan huruf untuk penambahan kata pada pembangkitan teka – teki silang

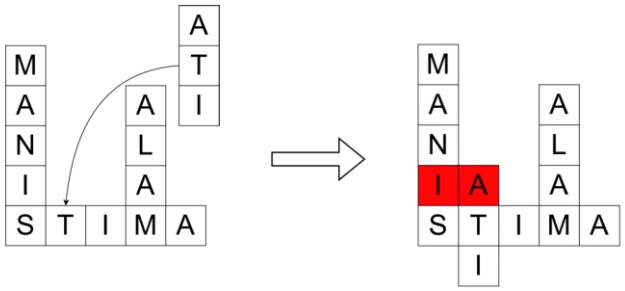
Permasalahan yang perlu dihadapi algoritma adalah memastikan bahwa penambahan kata pada posisi tersebut tidak beririsan secara salah dengan huruf dari kata lain yang sudah diletakkan.



Gambar 3.1.2. Ilustrasi ketika huruf untuk penambahan kata beririsan secara salah

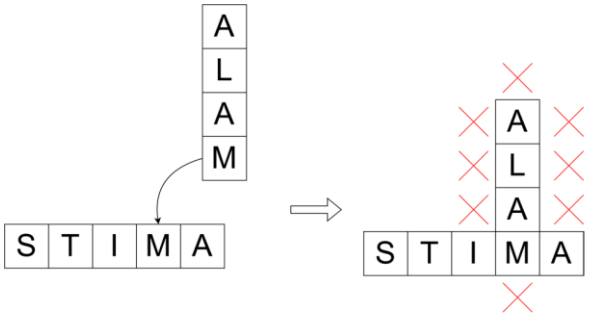
Untuk memastikan bahwa penambahan kata pada posisi tersebut tidak beririsan secara salah dengan huruf dari kata lain, maka petak beserta huruf yang sudah mengisinya perlu disimpan pada memori algoritma. Setelah itu akan dilakukan pengecekan untuk tiap huruf yang akan ditambahkan, huruf yang berada pada petak tujuanya harus sesuai, jika tidak, maka kata tidak bisa ditambahkan dan algoritma akan mencari petak lain untuk meletakkan kata tersebut.

Permasalahan lain yang perlu dihadapi algoritma adalah memastikan bahwa penambahan kata pada posisi tersebut tidak akan membuat rangkaian kata yang tidak diinginkan setelah kata tersebut diletakkan.



Gambar 3.1.3. Ilustrasi ketika penambahan kata menyebabkan rangkaian kata yang tidak diinginkan

Untuk memastikan bahwa penambahan kata pada posisi tersebut tidak membentuk kata yang tidak diinginkan ketika kata tersebut diletakkan, pendekatan diambil oleh algoritma adalah memastikan bahwa petak di sekeliling huruf yang tidak beririsan dengan kata lain adalah kosong kecuali huruf sebelum dan selanjutnya dan terdapat satu petak kosong di awal dan akhir kata.



Gambar 3.1.3. Ilustrasi pemastian posisi kosong untuk penambahan kata pada pembangkitan teka – teki silang

Perbandingan akan diulangi untuk setiap huruf dari kata yang sudah terletak pada petak. Jika tidak ada posisi yang memadai untuk kata tersebut, maka algoritma akan melanjutkan ke kata selanjutnya sampai didapatkan kata yang dapat diletakkan di suatu posisi.

Algoritma akan diulangi seterusnya sampai setiap kata pada kamus sudah terletak dengan benar pada petak atau tidak ada lagi kata yang dapat diletakkan. Dari algoritma tersebut dapat dihasilkan suatu *pseudocode* sebagai berikut.

```
While (possible & notEmpty(Wordlist)) do
  Possible <- false
  For(word in wordlist)
   For(placedLetter in possibleLocations)
    For(letter in word)
     If(letter = placedLetter)
      Add <- true
      For(otherLetter in word)
       If(letterOnSide not null)
        Add <- false
        break
       If(letterBelow != otherLetter)
        Add <- false
        break
      If(next not null or before not null)
       Add <- false
       break
      if(Add)
       break
    if(Add)
     break
  If(Add)
   placeWord(word)
   possible <- true
   break
```

Gambar 3.1.4 Pseudocode penambahan kata pada pembangkitan teka – teki silang

Penambahan akan dilakukan secara selang seling antara penambahan mendatar dan menurun untuk memungkinkan huruf lain agar ditambahkan. Jika penambahan secara mendatar dari semua huruf tidak memungkinkan, maka akan dicoba penambahan secara menurun. Berlaku juga sebaliknya jika penambahan secara menurun dari semua huruf tidak memungkinkan.

Perlu diketahui bahwa algoritma yang dihasilkan cukup sederhana dan masih memiliki Batasan. Salah satu batasannya adalah tidak memungkinkan untuk dua soal bersebelahan walaupun semua huruf yang bersebelahan membentuk kata yang terdapat pada kamus. Batasan ini menyebabkan kurangnya kepadatan teka – teki silang yang dihasilkan. Batasan lain adalah bahwa untuk suatu kata yang merupakan substring dari kata yang lebih panjang, kata yang lebih panjang dapat menimpa kata yang lebih pendek jika diletakkan setelahnya.

2. Pengembangan Algoritma menggunakan Heuristik

Algoritma yang dihasilkan masih dapat ditingkatkan secara kinerja menggunakan pendekatan heuristik. Heuristik pada penerapan algoritma dapat dibagi menjadi dua kriteria, yaitu teknik heuristik yang meningkatkan kinerja algoritma dari segi kecepatan dan teknik heuristic yang meningkatkan kinerja algoritma dari segi kualitas teka – teki silang.

Pendekatan heuristik yang dapat digunakan untuk meningkatkan kecepatan antara lain:

- Karena sudah dipastikan pada penambahan kata di sekeliling huruf yang tidak beririsan dengan kata lain adalah kosong kecuali huruf sebelum dan selanjutnya, pemilihan kandidat titik untuk penambahan mendatar dan menurun dapat dipisahkan. Tidak mungkin kandidat titik yang didapat dari penambahan mendatar akan berpetak kosong pada penambahan mendatar pada titik tersebut, sama halnya dengan menurun.

  Mengetahui hal tersebut, setiap huruf untuk kata yang ditambahkan secara mendatar akan menjadi kandidat tempat penambahan menurun dan setiap kata yang ditambahkan secara menurun akan menjadi kandidat tempat penambahan secara horizontal.

  Dalam pencarian, penambahan mendatar hanya akan mencari kandidat mendatar dan penambahan menurun hanya akan mencari kandidat menurun. Dengan demikian, jumlah titik pencarian berkurang dengan cukup drastis, yaitu setengahnya.

  Selain mempercepat waktu eksekusi program, teknik ini juga mengatasi penimpaan substring oleh string yang lebih panjang.

- Penambahan pada perpotongan tidak mungkin menghasilkan kata tanpa mengubah kata yang sudah terdapat pada perpotongan tersebut sebelumnya. Sehingga dapat dipastikan bahwa penambahan kata pada perpotongan akan selalu tidak sesuai. Titik yang merupakan perpotongan dapat dihilangkan dalam

pencarian titik penambahan, mengurangi jumlah titik pencarian.

Pendekatan heuristik yang dapat digunakan untuk meningkatkan kecepatan antara lain:

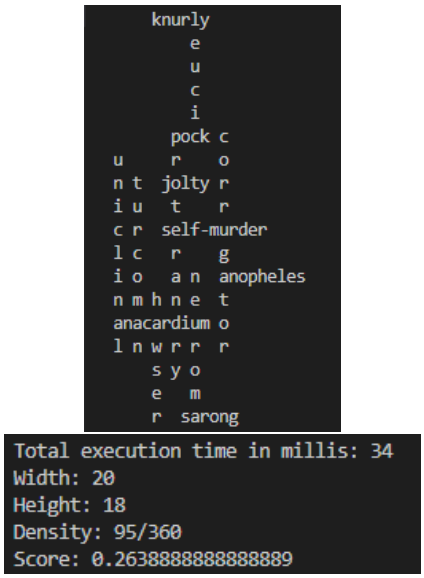
- Mengurutkan kata berdasarkan panjang dan memasukkan kata yang lebih panjang terlebih dahulu karena kata yang panjang memiliki kemungkinan untuk beririsan dengan huruf lain lebih tinggi daripada kata yang pendek. Dengan banyaknya irisan, teka – teki yang dihasilkan akan menjadi lebih kecil dan padat.
- Melakukan pencarian huruf yang sama dari tengah. Dengan hal itu, irisan yang berada pada tengah kata akan ditemukan terlebih dahulu dan kata diletakkan di tengah irisan dan tidak diujung. Hal ini dapat mengecilkan ukuran teka – teki yang dihasilkan dengan memadatkan peletakkan kata.
- Mendahulukan pengecekan dari titik kandidat yang ditambahkan lebih dahulu. Titik kandidat yang ditambahkan dari kata terlebih dahulu akan cenderung berposisi lebih tengah dari titik kandidat yang ditambahkan dari kata selanjutnya. Dengan demikian, jika pengecekan dilakukan dari titik kandidat yang ditambbhahkan lebih awal, kata akan diletakkan lebih tengah.
- Mendahulukan penambahan kata yang tidak memperbesar jumlah ruang petak keseluruhan. Hal ini akan memperlambat operasi karena menambah jumlah pencarian pertambahan kata sampai didapatkan penambahan kata yang tidak memperbesar ukuran teka – teki silang. Akan tetapi, pendahuluan kata tersebut dapat mengecilkan teka – teki silang yang dhasilkan secara cukup signifikan.

Perlu diingat bahwa beberapa pendekatan heuristik yang meningkatkan kualitas teka – teki yang dihasilkan akan memperlambat kecepatan operasi. Dengan pengetahuan tersebut. pengaplikasian heuristic untuk meningkatkan kualitas teka – teki silang yang dihasilkan adalah bersifat opsional.

3. Implementasi Menggunakan Bahasa Java

Penulis telah mengimplementasikan algoritma yang telah dibahas sebelumnya untuk eksperimen. Pada implementasi menggunakan bahasa java, dapat digunakan Map dengan hash berkunci lokasi petak dan bernilai karakter untuk menyimpan informasi petak. Penggunaan hash dapat mempercepat operasi pencarian lokasi penambahan kata.

Pada pengimplementasiannya, penulis mengambil kamus bahasa inggris berformat .json dari internet lalu mengembangkan algoritma menggunakan kata – kata yang terdapat pada kamus tersebut. Program hanya membangkitkan teka – teki silang berupa hasilnya dan analisis penilaiannya saja untuk keperluan analisis. Untuk pembuatan teka – teki silang sebenarnya dengan petak kosong beserta petunjuknya diperlukan pengembangan lebih lanjut.



Gambar 3.3.1 Contoh hasil dan informasi kualitas teka teki silang yang dibangkitkan dari kamus menggunakan 15 kata acak

Adapula penulis tidak menuliskan ataupun membahas kode secara lengkap pada makalah ini karena hal tersebut akan membuat makalah ini terlalu panjang. Kode lengkap dapat diakses melalui tautan github yang terdapat pada bagian – bagian selanjutnya.

Penulis mengimplementasikan beberapa kelas yang menunjang implementasi algoritma utama. Untuk algoritma utama, penulis membuat tiga buah file implementasi algoritma yang dibedakan oleh heuristik yang digunakannya dalam menyelesaikan masalah. Tiga file implementasi algoritma adalah sebagai berikut:

- CrosswordGeneratorPlain.java (tanpa heuristik)
- CrosswordGeneratorSpeed.java (heuristik kecepatan)
- CrosswordGeneratorComplete.java (semua heuristik)

Percobaan lalu dilakukan menggunakan file Main.java yang akan menjalankan algoritma. Algoritma diubah dari baris 5 yang berisikan kelas algoritma yang digunakan untuk menyelesaikan masalah. Sementara itu, kamus diubah dari parameter String yang diterima oleh kelas tersebut. String yang diterima berupa nama file kamus .json. File kamus harus terletak pada folder src/main/resource.

IV. ANALISIS

1. Analisis Kompleksitas

Berdasarkan *pseudocode* yang sudah dihasilkan, kasus terburuk adalah ketika untuk semua pengecekan berhasil pada pengecekan terakhir. Analisis kompleksitas dilakukan menganggap semua kata yang dimasukkan memiliki panjang yang sama agar mempermudah analisis. Selain itu, posisi memungkinkan dianggap ditambahkan sejumlah panjang kata ketika suatu kata tersebut diletakkan

Loop paling luar akan melakukan pengulangan sejumlah n kali sampai didapatkan semua kata terletak pada suatu petak dan

loop yang kedua akan melakukan pengulangan sejumlah kata yang tersisa. Kedua loop ini akan menghasilkan kompleksitas gabungan berupa deret yang dimulai dari n hingga 1.

$$n + (n-1) + \cdots + 1$$

Untuk setiap pengulangan tersebut terdapat pengulangan lagi, yaitu untuk memeriksa setiap kandidat yang terdapat pada petak. Dalam setiap penambahan, jumlah kandidat akan bertambah sejumlah panjang kata yang dimasukkan, yaitu m. Jumlah kandidat akan terus bertambah sampai berjumlah $(n-1)m$ pada pertambahan kata terakhir.

$$nm + (n-1)2m + \cdots + 1(n-1)m$$

Untuk setiap pertambahan, dilakukan dua kali looping sejumlah huruf dalam kata. Looping pertama untuk memeriksa kecocokan dan looping kedua untuk memeriksa kesesuaian yang sudah dibahas pada bagian III.

$$(nm + (n-1)2m + \cdots + 1(n-1)m)m^2$$

Dengan demikian, *pseudocode* tersebut, menganggap semua kata tertambahkan pada akhirnya, kompleksitas 5 buah loop akan menghasilkan kompleksitas total dengan notasi Big-O sebagai berikut:

$$O(n) = (nm + (n-1)2m + \cdots + 1(n-1)m)m^2$$

$$O(n) = m^3(n*1 + (n-1)2 + \cdots + 1*(n-1))$$

$$O(n) = m^3 \sum_{i=1}^{n} \left[\left(n - (i-1)\right) i \right]$$

$$O(n) = m^3 \sum_{i=1}^{n} (in - i^2 + i)$$

$$O(n) = m^3(n+1)\left(\sum_{i=1}^{n} i - \sum_{i=1}^{n} i^2\right)$$

$$O(n) = m^3(n+1)\frac{(n)(n+1)}{2} - \frac{(n)(n+1)(2n+1)}{6}$$

$$O(n) = m^3 \frac{n(n+1)(n+2)}{6}$$

$$O(n) = m^3 n^3$$

Dengan n adalah jumlah kata dan m adalah panjang huruf dari kata tersebut.

Dapat terlihat bahwa algoritma yang diimplementasikan memiliki kompleksitas yang tinggi, yaitu dengan orde jumlah pangkat 6. Walaupun begitu, kasus terburuk jarang untuk ditemukan dan algoritma bergantung kepada jumlah alfabet yang terdapat dalam bahasa yang digunakan. Semakin sedikit jumlah alfabet, maka irisan akan lebih sering ditemukan.

Penggunaan heuristik dalam mempercepat operasi bekerja dengan membagi dan mengurangi jumlah pengecekan dengan suatu jumlah tetap. Mengingat notasi big-O tidak memperhitungkan koefisien, penggunaan heuristik tidak mengubah kompleksitas big-O pada algoritma.

2. Analisis Kualitas Teka – Teki Silang

Seperti yang sudah dijelaskan pada bagian II, analisis kualitas teka – teki silang dilakukan dengan melihat kepadatan teka – teki silang yang dihasilkan. Ada pula kepadatan teka – teki silang yang dihasilkan bergantung kepada jumlah kata dan huruf yang terdapat pada kata tersebut sehingga tidak dapat ditentukan secara tepat efektivitas algoritma dari segi kualitas teka – teki silang yang dihasilkan.

Pada percobaan analisis, penulis mencoba mengaplikasikan algoritma tersebut kepada 15, 100, dan 1000 kata acak. Tanpa menggunakan heuristic apapun, algoritma menghasilkan teka – teki silang sebagai berikut:

Gambar 4.2.1 Hasil teka – teki silang yang dibangkitkan tanpa heuristik untuk 15, 100, dan 1000 kata acak.

Pada percobaan dapat terlihat bahwa teka – teki silang yang dihasilkan cenderung berbentuk lingkaran dan meninggalkan banyak ruang kosong di sudut – sudut ruang. Selain itu terlihat juga bahwa jumlah kata berbanding lurus terhadap kepadatan teka – teki silang yang dihasilkan. Hal ini didukung oleh perhitungan kepadatan, yang menunjukkan nilai kepadatan teka – teki silang tersebut masing – masing adalah 0.142, 0.206, dan 0.291. Ada pula waktu yang dibutuhkan pada perangkat penulis dalam milisekon untuk masing – masing adalah 18, 51, dan 732

Penulis juga mencoba sebuah test case yang berisi kata acak sejumlah 10000 kata. Percobaan ini menghasilkan teka – teki silang yang terlalu besar sehingga tidak ditampilkan pada layar. Sementara itu, kepadatan dari teka – teki silang yang dihasilkan adalah 0.377 dan waktu yang dibutuhkan untuk menghasilkan adalah 65433 milisekon, kepadatan yang meningkat mendukung hasil analisis yang sebelumnya.

3. Efektivitas Teknik Heuristik

Algoritma dicoba untuk ditingkatkan kualitasnya menggunakan teknik Heuristik. Dalam analisis keefektifan, teknik Heuristik dibagi menjadi heuristik yang meningkatkan kecepatan dari eksekusi dan heuristik yang meningkatkan kualitas teka – teki silang yang dihasilkan. Pada heuristik yang meningkatkan kualitas teka – teki silang yang dihasilkan, heuristik peningkatan kecepatan juga telah diimplementasikan.

Perbedaan waktu dan kepadatan yang dihasilkan dari ketiga algoritma adalah sebagai berikut dengan ρ adalah nilai kepadatan dan t adalah waktu yang diperlukan untuk membangkitkan teka – teki silang tersebut dalam milisekon:

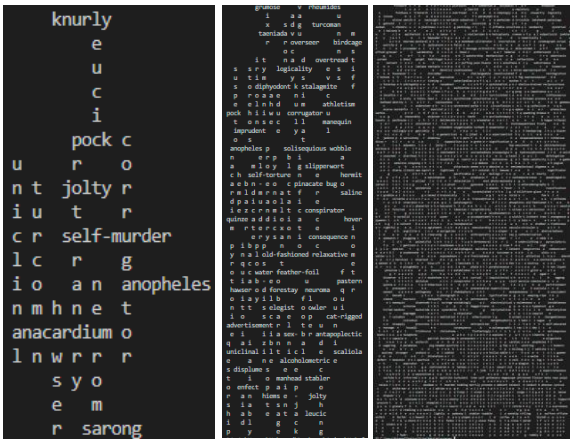
Tabel 4.3.1. Data yang diperoleh algoritma untuk tiap dataset

Algoritma	Dataset 15		Dataset 100		Dataset 1000		Dataset 10000	
	ρ	t	ρ	t	ρ	t	ρ	t
Tanpa Heuristik	0.142	18	0.206	51	0.291	732	0.377	65433
Heuristik Kecepatan	0.142	19	0.206	39	0.291	552	0.382	30580
Semua Heuristik	0.264	28	0.400	155	0.477	28130	X	X

ρ = Tingkat kepadatan
t = waktu (ms)
X = Tidak selesai

Untuk heuristik kecepatan, pada dataset 15, 100, dan 1000, teka – teki silang yang dihasilkan sama sama seperti tanpa heuristik. Namun, untuk dataset 10000 terdapat perbedaan dalam kepadatan. Hal ini dapat disebabkan oleh dipisahkannya titik mendatar dan menurun sehingga urutan pengecekkan dapat berbeda dibandingkan yang tidak menggunakan heuristic. Penghilangan perpotongan juga dapat menyebabkan perbedaan karena mengatasi penimpaan substring yang dapat terjadi Selain itu, untuk semua dataset terdapat peningkatan dari kecepatan.

Sementara itu, untuk heuristik kualitas, waktu yang dibutuhkan untuk menghasilkan teka – teki silang lebih lama. Waktu yang dibutuhkan heuristik kualitas lebih lama daripada tidak menggunakan heuristik. Heuristik kualitas juga membutuhkan waktu yang terlalu lama untuk dataset 10000 dan dianggap tidak selesai. Di sisi lain, kepadatan yang dihasilkan meningkat signifikan dan menghasilkan teka – teki silang seperti berikut:



Gambar 4.3.1 Hasil teka – teki silang yang dibangkitkan dengan heuristik untuk 15, 100, dan 1000 kata acak.

Selain kepadatan yang meningkat, dapat terlihat juga penggunaan teknik heuristik menyebabkan teka – teki silang untuk berbentuk segi empat. Selain itu, untuk jumlah kata yang sedikit kata – kata lebih terkonsentrasi ke suatu sudut dan untuk jumlah kata yang banyak setiap sudut lebih terisi dibandingkan tidak menggunakan heuristik.

Penambahan waktu yang banyak utamanya disebabkan oleh penambahan batasan untuk mendahulukan kata yang tidak memperbesar ukuran petak, sehingga lebih sering mencapai kasus terburuk ketika tidak ditemukan kata yang tidak memperbesar ukuran petak. Selain itu, terdapat juga overhead dari sorting yang dilakukan untuk kamus. Pengecekan titik dari yang ditambah lebih awal juga memiliki kemungkinan untuk tidak valid lebih besar daripada mengambil secara acak. Semua peningkatan waktu adalah penambahan berupa koefisien atau penambahan oleh orde yang lebih renah, seperti $nLog(n)$ untuk sorting sehingga notasi big-O tidak berubah.

V. KESIMPULAN

Dari analisis dan implementasi yang telah dilakukan pada bab sebelumnya terlihat bahwa memungkinkan untuk dibangkitkannya sebuah teka – teki silang menggunakan Algoritma Brute Force. Algoritma dasar yang dihasilkan memilki kepadatan dan kecepatan yang dapat ditingkatkan menggunakan heuristik. Ada pula algoritma yang dihasilkan memilki kompleksitas dengan notasi big-O, yaitu:

$$O(n) = m^3 n^3$$

Heuristik yang diimplementasikan dapat meningkatkan performa algoritma dari segi kecepatan, maupun dari segi kualitas teka – teki silang yang dihasilkan. Namun, heuristik yang meningkatkan kualitas teka – teki silang menyebabkan performa kecepatan menjadi lambat dan tidak dapat digunakan untuk kasus yang berukuran cukup besar.

Kesimpulan lain yang dapat diambil adalah bahwa walaupun pembangkitan teka – teki silang menggunakan brute force memungkinkan, algoritma yang digunakan memiliki kompleksitas yang cukup tinggi. Penggunaan heuristik dapat meningkatkan kecepatan dari algoritma yang digunakan, namun peningkatan kecepatan tersebut tidak menyebabkan perubahan notasi big-O sehingga tidak efektif untuk input yang berukuran besar. Dengan demikian, akan lebih baik jika ditemukan algoritma yang dapat membangkitkannya melalui pendekatan yang lain.

Repository Github

https://github.com/MuhamadAjiW/CrosswordGenerator

Video Youtube

https://youtu.be/RIMGg-_kQ6k

Ucapan Terima Kasih

Penulis berterima kasih kepada Tuhan yang Maha Esa karena telah memberikan kelancaran dalam pembuatan makalah berjudul "Pembangkitan Soal Teka – Teki Silang Menggunakan Algoritma Brute Force". Penulis juga ingin berterima kasih kepada Dr. Ir. Rinaldi Munir, M.T. selaku dosen dan pembimbing penulis dalam mata kuliah IF2211 Strategi Algoritma tahun ajaran 2022/2023. Selain itu, penulis juga ingin berterima kasih kepada keluarga, teman – teman, serta pihak-pihak lain atas segala kontribusi dalam penyelesaian makalah ini. Semoga Tuhan yang Maha Esa membalas semua kebaikan dengan kebaikan yang lebih berlipat ganda.

Referensi

[1] Munir, R. (2022). Bahan Kuliah IF2211 Strategi Algoritma: *Algoritma Brute Force* (Bagian 1). Program Studi Informatika Sekolah Elektro dan Informatika, ITB. Retrieved Mei 21, 2023, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[2] Munir, R. (2022). Bahan Kuliah IF2211 Strategi Algoritma: *Algoritma Brute Force* (Bagian 2). Program Studi Informatika Sekolah Elektro dan Informatika, ITB. Retrieved Mei 21, 2023, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf

[3] Wulandari, M. P., Raharjo, D., & Lutfi, H. (n.d.). Algoritma Exhaustive Search Sebagai Pencari Solusi Terbaik. Retrieved Mei 21, 2023, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/Makalah/MakalahStmik20.pdf

[4] Kustiani, R. (2015, April 18). Hari ini 91 tahun lalu, tts resmi dibukukan. *Tempo.Co*, pp. 1–1. Retrieved Mei 21, 2023, from https://tekno.tempo.co/read/658722/hari-ini-91-tahun-lalu-tts-resmi-dibukukan

[5] Mohr, A. (2007). Quantum Computing in Complexity Theory and Theory of Computation. Retrieved Mei 21, 2023, from http://www.austinmohr.com/Work_files/complexity.pdf

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, 21 Mei 2023

Muhamad Aji Wibisono – 13521095