Penerapan Algoritma *Backtracking* untuk Solusi *Game Crossword Puzzle*

Muhammad Fadli Gunardi - 13519130
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519130@std.stei.itb.ac.id

Abstrak—*Game Crossword Puzzle* merupakan *game* yang *tricky* dan tidak mudah untuk diselesaikan, bahkan untuk membuat sebuah *Crossword Puzzle* saja tidak semudah yang kita bayangkan. Oleh karena itu, penulis tertarik untuk mengangkat suatu permasalahan mengenai bagaimana cara untuk mempermudah penyelesaian suatu *Crossword Puzzle*. Berdasarkan studi kasus pada makalah ini, ditemukan bahwa apabila kita menggunakan algoritma *Backtracking* solusi yang dihasilkan akan terjamin kebenarannya, serta lebih praktis, dan efisien dibandingkan cara manual. Selain itu, algoritma *Backtracking* pada permsalahan ini juga bertindak sebagai fungsi kelayakan solusi.

Kata kunci—Backtracking; Crossword Puzzle; game

I. PENDAHULUAN

Game Crossword Puzzle atau dalam Bahasa Indonesia disebut teka teki silang (TTS) merupakan suatu permainan dimana kita harus mengisi ruang-ruang kosong yang berbentuk kotak putih dengan huruf-huruf yang membentuk sebuah kata berdasarkan petunjuk yang sudah diberikan. Petunjuk dapat dibagi ke dalam kategori 'mendatar' dan 'menurun' bergantung pada posisi kata-kata yang harus diisi. Jawaban dari suatu petunjuk harus sesuai panjangnya dengan petak yang tersedia. Serta apabila suatu petak jawaban beririsan dengan petak jawaban petunjuk lain maka huruf yang beririsan harus sama. *Crossword Puzzle* mempunyai berbagai tingkat kesulitan yang diukur berdasarkan berbagai parameter, misalkan besar dimensi *Crossword Puzzle* tersebut, tingkat abstraksi sebuah pertanyaan, kombinasi jawaban yang mirip, hingga pemilihan diksi kosa kata jawaban yang terdengar familiar atau asing. Permainan *Crossword Puzzle* juga memiliki beberapa kelebihan, diantaranya dapat meningkatkan imajinasi seseorang serta daya nalar seseorang.

Pembuatan *game Crossword Puzzle* juga tidak semudah yang dibayangkan karena *Crossword Puzzle* memiliki berbagai aturan yang tidak boleh dilanggar, seperti tidak boleh ada petak jawaban yang bersebelahan baik secara horizontal maupun secara vertikal, sebuah pertanyaan juga harus relevan dan saling berkaitan dengan kunci jawaban dari pertanyaan tersebut, sasaran dari *game Crossword Puzzle* ini juga harus diperhatikan karena tidak mungkin anak-anak mengerjakan *Crossword Puzzle* dengan pertanyaan umum selevel orang dewasa. Dapat dibayangkan pembuatan *Crossword Puzzle* saja sudah rumit, bagaimana kita dapat menyelesaikannya?

Penyelesaian *Crossword Puzzle* secara umum adalah dengan menjawab pertanyaan dengan jawaban yang panjangnya sesuai petak yang disediakan, namun cara ini menimbulkan persamasalahan, apabila terdapat jawaban dari dua pertanyaan yang berisisan secara vertikal dan horizontal maka huruf yang beririsan itu harus sesuai. Kasus terburuk terjadi apabila kita sudah mengisi semua jawaban dan tersisa dua pertanyaan yang jawabanya saling berisan secara horizontal dan vertikal ternyata huruf yang beririsan tidak sesuai, maka kita harus mengulangi langkah-langkah sebelumnya dan mencari-cari dimana kesalahan kita. Tentunya cara ini tidak efisien bukan?

Oleh karena itu, seiring berjalannya waktu dan teknologi yang berkembang cepat, pada saat ini penyelesaian *game Crossword Puzzle* dapat lebih mudah dengan bantuan komputer memanfaatkan algoritma *Backtracking*. Algoritma *Backtracking* disini juga menjamin solusi atas *Crossword Puzzle* dengan batasan-batasan dan asumsi tertentu yang akan dibahas selanjutnya pada makalah ini.

II. LANDASAN TEORI

Algoritma *Backtracking* atau runut-balik merupakan suatu metode pemecahan masalah yang mangkus, terstruktur, dan sistematis yang merupakan pengembangan dari algoritma *Depth First Seach* (*DFS*) yang sudah dipangkas (*pruning*) atas simpul-simpul yang tidak mengarah ke solusi. Algoritma *Backtracking* juga merupakan perbaikan dari exhaustive search, dimana pada *exhaustive search* semua kemungkinan solusi dieksplorasi satu per satu, berbeda dengan algoritma *Backtracking* yang melakukan pemangkasan.

1. Solusi persoalan.

   Solusi dinyatakan sebagai vektor dengan n-*tuple*: $X = (x_1 , x_2 , \ldots, x_n)$, $x_i \in S_i$. Umumnya $S_1 = S_2 = \ldots = S_n$.

   Contoh: Pada persoalan 1/0 *knapsack* $S_i = \{0, 1\}$, $x_i =$ 0 atau 1

2. Fungsi Pembangkit nilai x_k

   Dinyatakan sebagai predikat T()

   T(x[1], x[2], …, x[k – 1]) membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.

3. Fungsi Pembatas (*bounding function*)

Dinyatakan sebagai predikat B(x_1 , x_2 , …, x_k)

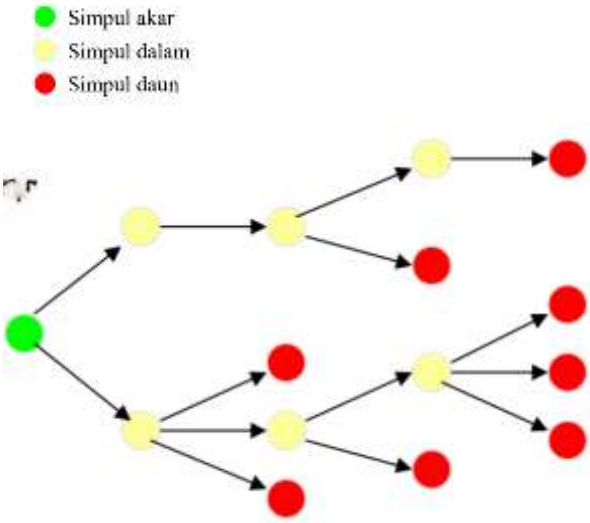
B bernilai *true* jika (x_1 , x_2 , …, x_k) mengarah ke solusi. Mengarah ke solusi artinya tidak melanggar kendala (*constraints*)

Jika *true*, maka pembangkitan nilai untuk xk+1 dilanjutkan, tetapi jika *false*, maka (x_1 , x_2 , …, x_k) dibuang.

4. Pengorganisasian Solusi

Semua kemungkinan solusi dari persoalan disebut ruang solusi (*solution space*). Ruang solusi diorganisasikan ke dalam struktur pohon berakar. simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi (cabang) dilabeli dengan nilai-nilai x_i . Lintasan dari akar ke daun menyatakan solusi yang mungkin.

Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*).



Gambar 1 Pohon Ruang Status
Sumber: www.cis.upenn.edu/.../35-backtracking.pt

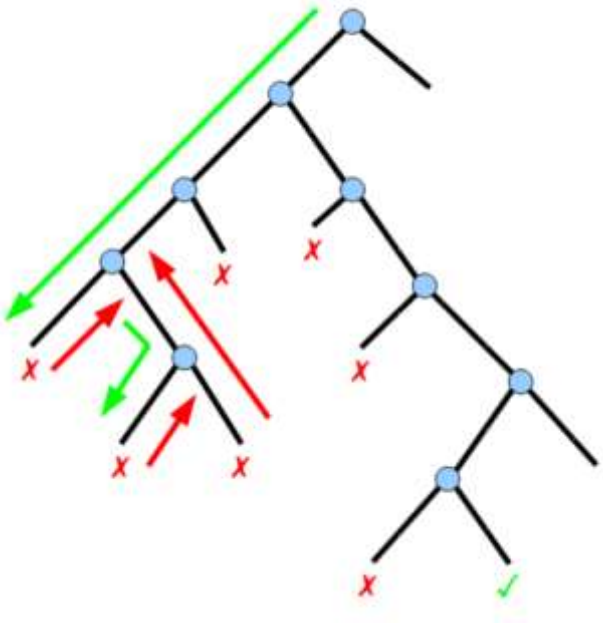
Backtracking dapat dipandang sebagai pencarian di dalam pohon dari akar menuju daun (simpul solusi)

5. Prinsip Pencarian Solusi dengan Algoritma *Backtracking*

Solusi algoritma *Backtracking* dicari dengan membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari akar ke daun. Aturan pembangkitan simpul yang dipakai adalah mengikuti aturan *Depth First Search* (*DFS*). Simpul-simpul yang sudah dibangkitkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*Expand-node*).

Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dimatikan" sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk mematikan simpul-E adalah dengan menerapkan fungsi pembatas (*bounding function*). Ketika sebuah simpul berarti, maka secara implist kita telah memangkas (*pruning*) simpul-simpul anaknya.

Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian *Backtrack* ke simpul pada aras diatasnya. Lalu, teruskan dengan membangkitkan simpul anak yang lainnya. Selanjutnya simpul ini menjadi simpul-E yang baru. Pencarian dihentikan bila kita telah sampai pada *goal node.*



Gambar 2 Contoh Prinsip Pencarian Solusi dengan *Backtracking*

Sumber: http://www.w3.irg/2011/Talks/01-14-steven-phenotype/

6. Skema Umum Algoritma *Bactraking*

Setiap simpul dalam pohon ruang status (kecuali simpul daun) berasosiasi dengan sebuah pemanggilan rekursif. Jika jumlah simpul dalam pohon ruang status adalah 2^n atau n!, maka pada kasus terburuk, algoritma runut-balik membutuhkan waktu dalam $O(p(n)2^n)$ atau O(q(n)n!), dengan p(n) dan q(n) adalah polinom derajat n yang menyatakan waktu komputasi setiap simpul.

```
procedure RunutBalikR(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1], x[2], ..., x[k – 1] sudah
ditentukan nilainya.
Luaran: semua solusi x = (x[1], x[2], ..., x[n])
}
Algoritma:
  for setiap x[k] ∈ T(x[1], x[2], ..., x[k – 1]) do
      if B(x[1], x[2], ... ,x[k]) = true then
          if (x[1], x[2], ... , x[k]) adalah lintasan dari akar ke simpul solusi then
              write(x[1], x[2], ..., x[k])     { cetak solusi }
          endif
          if k < n then
              RunutBalikR(k+1)    { tentukan nilai untuk x[k+1] }
          endif
      endif
  endfor                                   Pemanggilan pertama kali: RunutBalikR(1)
```

Gambar 3 Algoritma *Backtracking*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf

III. PEMBAHASAN

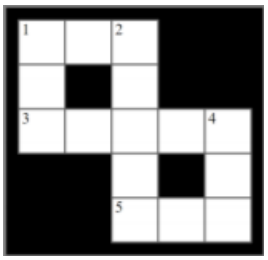
Penggunaan algoritma *Backtracking* untuk penyelesaian *game Crossword Puzzle* harus memenuhi suatu kondisi, yaitu sudah tersediannya kunci jawaban pada tiap soal petunjuk. Penyelesaian *game Crossword Puzzle* secara mutlak dari awal mula permainan tidak dibahas pada pembahasan ini karena tidak mengandung unsur *Backtracking*, serta keberagaman jawaban yang mungkin menjawab pertanyaan-pertanyaan yang tertera pada *game Crossword Puzzle*. Sehingga dapat kita asumsikan kunci jawaban dari *game* tersebut sudah ada, namun dengan catatan kita tidak tahu jawaban ini menjawab pertanyaan yang mana.

Untuk menyelesaikan solusi *game Crossword Puzzle* pertama-tama kita harus membuat suatu tipe data berjenis *key* dan *value* terlebih dahulu. *Key* merepresentasikan panjang karakter suatu kata, sedangkan *value* merepresentasikan sebuah kata. Tipe data ini dapat berbentuk *dictionary*. *Dictionary* dibuat dengan cara iterasi sebuah *loop* pada suatu *list* yang sudah dimasukkan kunci jawaban atas pertanyaan yang ada di game kalkulasi simulai dengan mencari panjang suatu kata indeks pertama list. Misal panjangnya 4, maka masukkan kata tersebut ke *dictionary* dengan *key* 4. Apabila pada iterasi selanjutnya ditemukan panjang suatu kata adalah 4 juga maka masukkan kata tersebut ke dictionary 4, tetapi apabila panjang suatu kata berbeda dengan yang sebelumnya maka buat *dictionary* baru demikian berlaku pada iterasi selanjutnya. Pada suatu kasus, apabila terdapat kunci jawaban dengan 2 kata yang dipisahkan oleh spasi, maka spasi dihitung 1 karakter panjangnya

Kata	Panjang Kata
HVS	3
SALAH	5
BEA	3
WTS	3
ALLOW	5
BUS	3

Tabel 1 Kunci Jawaban

Langkah kedua adalah membuat suatu basis data jawaban dengan atribut jenis pertanyaan, urutan pertanyaan, dan jawaban tersedia. Jenis pertanyaan merupakan urutan suatu bentuk pertanyaan baik itu horizontal maupun vertikal. *Value* dari atribut pertanyaan terurut berdasarkan angka, lalu horizontal terlebih dahulu baru vertikal. dengan penulisan seperti berikut, baris 1, 1 horintontal; baris 2, 1 vertikal; baris 3, 2 vertikal; baris 4, 3 horizontal; dst. Atribut urutan pertanyaan diisi dengan angka 1 hingga sebanyak jumlah pertanyaan yang ada. Sedangkan atribut jawaban tersedia berisi *value* sebuah *dictionary* yang *key* nya sesuai dengan jumlah petak pada petunjuk suatu pertanyaan, misalkan 1 horizontal mebutuhkan 5 petak maka *value* akan diakses dari *dictionary* yang *key*-nya bernilai 5.



Gambar 4 Contoh *Game Crossword Puzzle*

Sumber: https://iopscience.iop.org/article/10.1088/1742-6596/1363/1/012075/pdf

Jenis Pertanyaan	Urutan Pertanyaan	Jawaban Tersedia
1 horizontal	1	HVS, BEA, WTS, BUS
1 vertikal	2	HVS, BEA, WTS, BUS
2 vertikal	3	SALAH, ALLOW
3 horizontal	4	SALAH, ALLOW
4 vertikal	5	HVS, BEA, WTS, BUS
5 horizontal	6	HVS, BEA, WTS, BUS

Tabel 2 Basis Data Jawaban

Langkah selanjutnya adalah mendefinisikan aturan-aturan yang ada pada sebuah *game Crossword Puzzle*. Aturan-aturan yang berlaku disini bentuknya tentatif tidak *fix* pada setiap permainan crossword puzzle, tetapi bergantung pada bentuk *Crossword Puzzle* itu sendiri. Aturan-aturan itu terdefinisi apabila terdapat petak yang beririsan satu dengan yang lainnya. Aturan ini dibuat dengan prioritas urutan jenis pertanyaan yang berhimpitan lebih kecil didahulukan, misalkan 1 horizontal dengan 2 vertikal dibandingkan dengan 1 vertikal 3 horizontal, maka didahulukan 1 horizontal dengan 2 vertikal. Oleh karena itu, aturan-aturannya terdefinisi sebagai berikut:

1. Huruf ke-1 angka 1 horizontal = huruf ke-1 angka 1 vertikal
2. Huruf ke-3 angka 1 horizontal = huruf ke-1 angka 2 vertikal

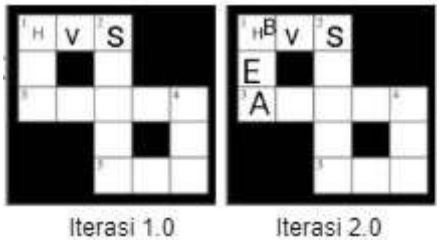
3. Huruf ke-1 angka 2 vertikal = huruf ke-3 angka 1 horizontal
4. Huruf ke-1 angka 3 horizontal = huruf ke-3 angka 1 vertikal
5. Huruf ke- 3 angka 3 horizontal = huruf ke-3 angka 2 vertikal
6. Huruf ke-5 angka 3 horizontal = huruf ke-1angka 4 vertikal
7. Huruf ke-1 angka 5 horizontal = huruf ke-5 angka 2 vertikal
8. Huruf ke-3 angka 5 horizontal = huruf ke-3 angka 4 vertikal

Aturan di atas merupakan langkah-langkah dalam algoritma *Backtracking* karena apabila suatu poin tidak terpenuhi maka pengecekan akan mundur satu indeks dan maju lagi, apabila tidak terpenuhi lagi maka pengecekan akan mundur 2 indeks, proses ini berlaku seterusnya dalam *Backtracking*. Dalam pengambilan suatu kata dari basis data jawaban dilakukan secara iteratif mulai dari paling kiri hingga paling kanan. Contoh:

- Iterasi pertama pada 1 horizontal, terdapat jawaban tersedia berupa HVS, BEA, WTS, BUS, maka akan diambil kata HVS.
- Iterasi kedua akan mengambil 1 vertikal, terdapat jawaban berupa HVS, BEA, WTS, BUS. Akan tetapi, apabila suatu kata sudah diambil maka kata yang sama tidak boleh diambil lagi, misalkan pada iterasi pertama akan mengambil kata HVS maka iterasi kedua tidak boleh mengambil kata HVS harus kata selanjutnya yaitu BEA.

Proses langkah iteratif algoritma *Backtracking* akan dimulai dari aturan pertama pada *game Crossword Puzzle*. Proses *Backtracking* akan mencocokan kata dengan aturan-aturan yang sudah terdefinisi sebelumnya, apabila kata tidak memenuhi poin aturan 1 maka akan *Backtracking* ke iterasi kedua, apabila iterasi kedua sudah mengganti dengan semua kata yang tersedia di basis data jawaban atribut jawaban tersedia masih belum ditemukan kata yang cocok, maka akan dilanjutkan dengan *Backtracking* ke proses iterasi 1 dan seterusnya. Begitupun sebaliknya, apabila suatu kata ditemukan tidak cocok pada poin aturan 2, maka kata tersebut akan dibuang pada iterasi tersebut dan akan digeser ke kanan sejauh 1 indeks pada kata yang tersedia di basis data jawaban yang barisnya sesuai dengan jenis pertanyaan tersebut. Apabila kata yang tersedia pada baris tersebut tidak memenuhi poin aturan tersebut maka akan dilakukan proses *Backtracking* ke poin aturan sebelumnya dengan mengganti kata pada iterasi sebelumnya.

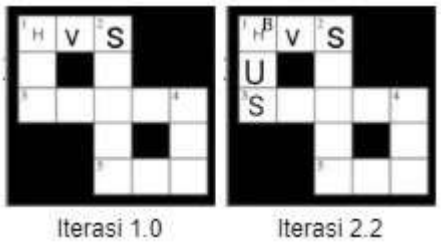
Berikur ilustrasi penyelesaian *game* crossword puzzle menggunakan algoritma *Backtracking*:



Gambar 5 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

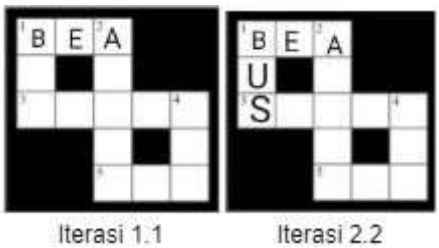
Gambar di atas merupakan iterasi kesatu dan iterasi kedua pada algoritma *c* game crossword ini. Pada iterasi 2 didapatkan bahwa jawaban kata kunci melanggar poin peraturan 1, yaitu huruf pertama 1 horizontal dengan huruf pertama 1 vertikal harus sama. Pada gambar di atas yang dimaksud dengan iterasi 1.0 adalah iterasi ke-1 dengan 0 menandakan indeks kata ke-0 di baris yang sesuai pada jawaban tersedia di basis data jawaban.

Karena pelanggaran peraturan poin pertama iterasi kedua harus menggeser kata pada jawaban tersedia di basis data jawaban sebanyak 1 indeks ke kanan hingga didapatkan kata yang sesuai.



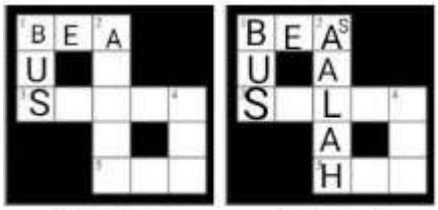
Gambar 6 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi 2.2 kata tersebut masih melanggar poin peraturan pertama, yaitu huruf pertama 1 horizontal dengan huruf pertama 1 vertikal harus sama. Oleh karena itu, harus dilakukan proses *Backtracking* Kembali ke iterasi pertama dengan catatan kata pada iterasi pertama harus dibuang dan digantikan dengan kata yang indeksnya bergeser 1 ke kanan pada jawaban tersedia di basis data jawaban.



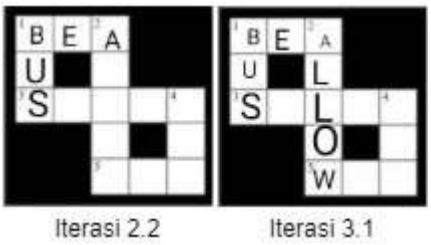
Gambar 7 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Setelah iterasi 1.1 dilanjutkan dengan iterasi 2.0 akan tetapi iterasi 2.1 dilewatkan karena tidak boleh ada jawaban kata yang sama. Pada akhirnya pada iterasi 2.2 ditemukan kata yang cocok pada poin peraturan pertama sehingga iterasi dilanjutkan ke indeks selanjutnya.



Gambar 8 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi indeks 3.0 melanggar poin peraturan kedua, yaitu huruf pertama 2 vertikal harus sama dengan huruf ketiga 1 horizontal. Oleh karena itu, proses iterasi 3.0 akan dilanjutkan dengan menggeser kata pada jawaban tersedia di basis data jawaban sebanyak 1 indeks ke kanan hingga didapatkan kata yang sesuai.



Gambar 9 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi 3.1 didapatkan kata yang sesuai dengan poin perturan 2 sehngga iterasi dapat dilanjutkan ke indeks selanjutnya.

Gambar 10 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Iterasi 3.1 hingga iterasi selanjutnya tidak ditemukan permasalahan kata yang melanggar poin peraturan sehingga pada iterasi 5.0 didapatkan solusi yang sesuai.

Fungsi pembatas atau *bounding function* pada algoritma ini terletak pada, apabila pada proses iteratif tidak memenuhi syarat pada suatu aturan yang tertera pada suatu *game Crossword Puzzle*, maka kata tersebut akan dibuang dan dilakukan proses *Backtracking* ke kata selanjutnya. Proses *Backtracking* ini akan berlangsung secara rekursif hingga ditemukannya sebuah solusi.

Selain sebagai penjamin terselesaikanya solusi dari suatu game *Crossword Puzzle* algoritma *Backtracking* disini akan menjadikan proses lebih efisien dibandingkan kita mencocokan secara manual (tanpa melakukan algoritma) karena pada *worst case*-nya, apabila tinggal 1 petak yang diisi. Akan tetapi, kata yang akan diisi hurufnya tidak beririsan dengan petak yang seharusnya beririsan, maka kita akan mencari-cari kesalahan kita, "dimana kita melakukan kesalahan itu?" atau "pada pertanyaan berapa kita melakukan kesalahan?" sehingga akan berujung pada keputusasaan. Solusi manual juga tidak terjamin terselesaikanya solusi karena ada faktor dimana daya pikir atau daya nalar suatu individu sudah mencapai batasnya sehingga tidak mempunyai ide lagi sehingga bertanya-tanya "pertanyaan ini harus dijawab dengan jawaban yang mana, yaa?".

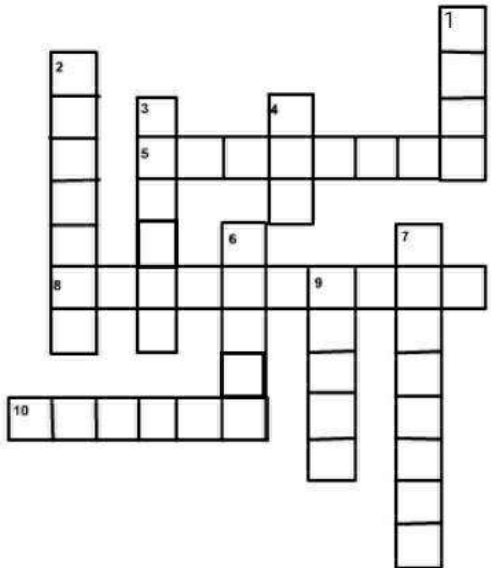
Algoritma *Backtracking* di atas dapat di optimalkan, agar waktu eksekusi yang dibutuhkan lebih cepat dengan cara memasukkan jawaban tersedia ke basis data jawaban secara terurut berdassarkan urutan *alphabet*. Alasannya adalah apabila terdapat petak yang beririsan, maka tentu petak yang beririsan tersebut harus diisi dengan huruf yang sama. Oleh karena itu, apabila ditemukan pada sebuah iterasi huruf tidak sama pada petak tersebut, tidak perlu menggeser kata secara berlebihan pada pengambilan kata di basis data karena terjamin bahwa huruf pertama yang sama akan terletak bersebalahan, sebagai contoh berikut:

panjang kata 3 terurut = BEA, BUS, DAD, DIG, HVS, WTS

panjang kata 3 tidak terurut = HVS, BEA, WTS, DAD, BUS, DIG

Apabila pada sebuah iterasi algoritma *Backtracking Crossword Puzzle* huruf yang sesuai adalah DAD dan DIG, Apabila dilakukan pengurutan kata, maka kita cukup melakukan 2+2 = 4 kali penggeseran kata, sedangkan apabila kata tidak dilakukan pengurutan iterasi yang diambil pertama adalah penggeseran ke-4 sedangkan iterasi kedua yang diambil adalah penggeseran ke-6, tentunya ini tidak efisien karerena kita melakukan penggeseran sebanyak 3+4 = 7 kali. Oleh karena itu dengan bentuk seperti ini akan terjamin bahwa huruf pertama yang karakternya sama terletak bersebelahan sehingga tidak perlu menggeser kata terlalu jauh.

IV. STUDI KASUS



Gambar 11 Studi Kasus *Game Crossword Puzzle* Lain

Kata	Panjang Kata
ONE	3
STAR	4
REINDEER	8
ELEVEN	6
FROSTY	6
WESTWARD	8
CUPID	5
ROCKING	7
NUTCRACKER	10
GREEN	5

Tabel 3 Tabel Kunci Jawaban

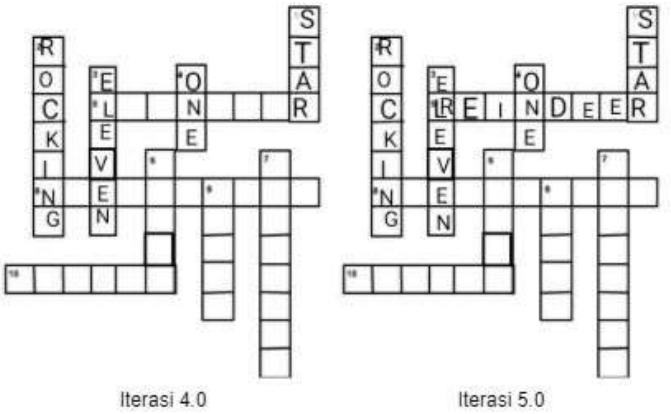
Setelah mengelompokkan tiap kata berdasarkan panjangnya dan dimasukkan ke sebuah dictionary berdasarkan panjang katanya, lalu buat basis data jawaban dari *Crossword Puzzle* tersebut. Untuk melakukan optimasi pada algoritma *Backtracking* hendaknya urutan kata yang dimasukkan ke dalam basis data terurut berdasarkan *alphabet*.

Jenis Pertanyaan	Urutan Pertanyaan	Jawaban Tersedia
1 vertikal	1	STAR
2 vertikal	2	ROCKING
3 Vertikal	3	ELEVEN, FROSTY
4 vertikal	4	ONE
5 horizontal	6	REINDEER, WESTWARD
6 vertikal	7	CUPID, GREEN
7 vertikal	8	REINDEER, WESTWARD
8 horizontal	9	NUTCRACKER
9 vertikal	10	CUPID, GREEN
10 horizontal	11	ELEVEN, FROSTY

Gambar 14 Basis Data Jawaban

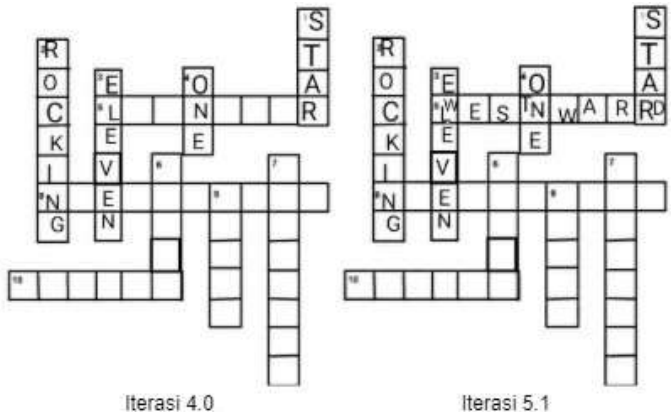
Langkah selanjutnya adalah mendefinisikan aturan pada *Crossword Puzzle* tersebut, yaitu:

1. Huruf ke-4 angka 1 vertikal = huruf terakhir angka 5 horizontal
2. Huruf ke-2 angka 3 vertikal = huruf ke-1 angka 5 horizontal
3. Huruf ke-2 angka 4 vertikal = huruf ke-4 angka 5 horizontal
4. Huruf ke-6 angka 2 vertikal = huruf ke-1 angka 8 horizontal
5. Huruf ke-2 angka 3 vertikal = huruf ke-3 angka 8 horizontal
6. Huruf ke-2 angka 6 vertikal = huruf ke-5 angka 8 horizontal
7. Huruf ke-2 angka 7 vertikal = huruf ke-8 angka 8 horizontal
8. Huruf ke-1 angka 9 vertikal = huruf ke-6 angka 8 horizontal
9. Huruf ke-5 angka 6 vertikal = huruf ke-6 angka 10 horizontal



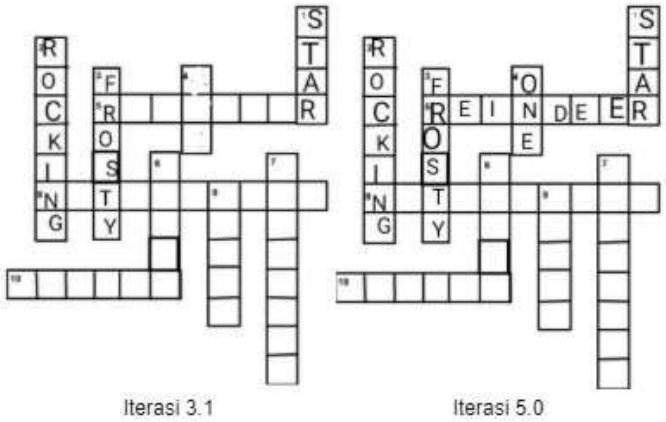
Gambar 12 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi 5.0 ditemukan pelanggaran perturan pada poin ke-2 sehingga iterasi 5.0 akan digeser sebanyak satu kata ke kanan pada basis data jawaban



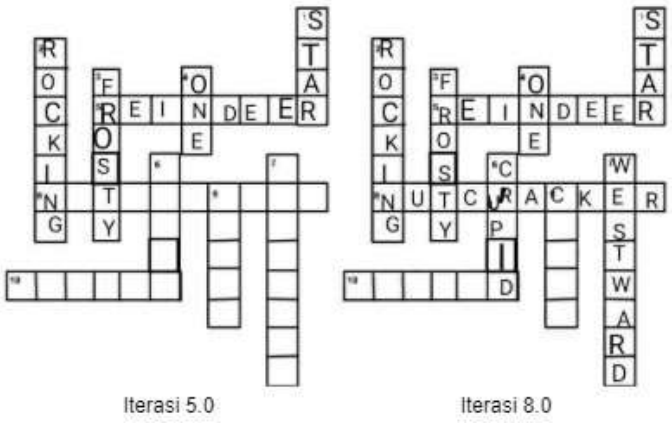
Gambar 13 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi 5.1 masih ditemukan pelanggaran pada peraturan poin ke-2 sehingga akan dilakukan *Backtracking* ke iterasi sebelumnya. *Backtracking* dilakukan hingga iterasi 3 karena pada iterasi 4 tidak tersedia kata yang lain pada basis data jawaban



Gambar 14 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi ke-3 karena terdapat kata lain yang tersedia pada basis data jawaban, maka kata tersebut dibuang dan digantikan dengan kata lain yang tersedia pada basis data jawaban dengan cara menggeser sebanyak 1 indeks ke kanan dari kata sebelumnya. Lalu iterasi akan dilanjutkan Kembali. Pada iterasi 5 tidak ditemukan pelanggaran peraturan sehingga iterasi dapat dilanjutkan.



Gambar 15 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi 8.0 ditemukan pelanggaran peraturan poin ke-6. Akan tetapi pada iterasi 8 tidak ditemukan kata lain yang tersedia pada basis data jawaban sehingga akan dilakukan *Backtracking* ke iterasi sebelumnya. Beigutpun dengan iterasi ke-7 tidak ditemukan kata lain yang tersedia pada basis data jawaban sehingga dilakukan *Backtracking* ke iterasi sebelumya.

Gambar 16 Langkah Penyelesaian Solusi Menggunakan Algoritma *Backtracking*

Pada iterasi ke-6 karena terdapat kata lain yang tersedia pada basis data jawaban, maka kata tersebut dibuang dan digantikan dengan kata lain yang tersedia pada basis data jawaban dengan cara menggeser sebanyak 1 indeks ke kanan dari kata sebelumnya. Setelah ittu, iterasi dilanjutkan hinggga akhirnya didapatkan sebuah solusi.

KESIMPULAN

Algoritma *Backtracking* dapat digunakan untuk mencari solusi sebuah game *Crossword Puzzle*. Algoritma *Backtracking* juga menjamin adanya solusi dan langkah-langkah penyelesain yang efisien. Akan tetapi, dengan kondisi-kondisi yang harus dipenuhi sebelumnya, yaitu adanya basis data yang menampung kunci jawaban suatu game *Crossword Puzzle*, basis data jawaban juga perlu dukungan dengan kalkulasi banyaknya huruf di tiap kata pada kunci jawaban, serta terdefinisinya aturan-aturan yang menyesuaikan sesuai bentuk petak pada suatu permainan *Crossword Puzzle*.

Oleh karena itu, algoritma *Backtracking* untuk menyelesaikan sebuah *game Crossword Puzzle* tidak terdefinisi mutlak untuk semua jenis game *Crossword Puzzle* karena jawaban-jawaban yang ada pada tiap jenis *Crossword Puzzle* berbeda-beda dan jenis petak tiap *Crossword Puzzle* berbeda-beda. Akan tetapi, langkah-langkah penyelesaian algoritma backtracking pada *game Crossword Puzzle* berlaku untuk semua jenis *Crossword Puzzle*.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada Allah SWT dengan izinnya penulis dapat menuntaskan tugas makalah ini. Ucapan terima kasih juga ingin disampaikan oleh penulis kepada dosen mata kuliah Strategi Algoritma, IF 2211 terutama kepada Bapak Harlili selaku dosen pengajar kelas K03, atas ilmunya yang telah disampaikan dapat berguna bagi penulis untuk menerapkanya dalam makalah ini.

REFRENSI

[1] Munir, Rinaldi. Slide Kuliah: Algoritma Runut-balik (Backtracking), Bandung, 2021.

[2] Wulan, Suwatra, Jampel (2019). Pegembangan Media Permainan Edukatif Teka-Teki Silang Berorientasi Pendidikan Karakter pada Mata Pelajaran IPS. Singaraja.
[3] Nurdin et al 2019 J. Phys.: Conf. Ser. 1363 01207
[4] https://fatkhan.web.id/model-pembelajaran-crossword-puzzel-teka-teki-silang/ diakses pada tanggal 10 Mei 2021

LINK YOUTUBE

https://youtu.be/DngklfXlGUg

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, 10 Mei 2021

Muhammad Fadli Gunardi