Aplikasi Algoritma Runut-balik pada Penyelesaian Teka Teki Mengisi Angka

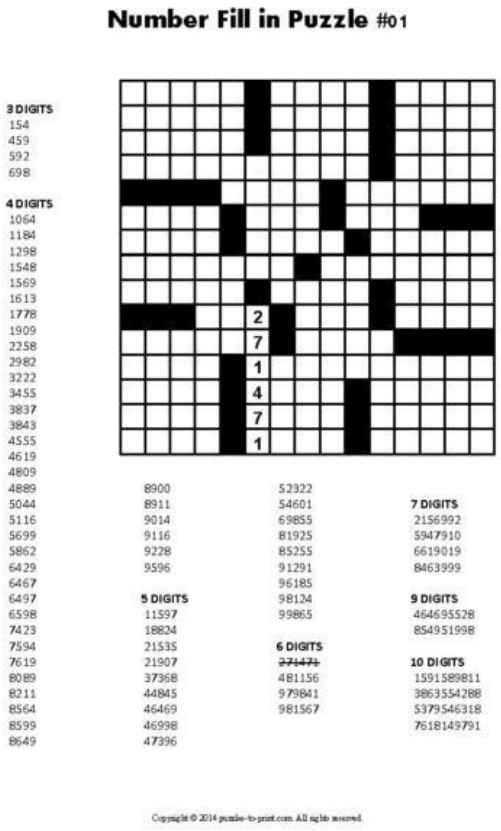
William - 13515144
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl.Ganesha 10 Bandung 40132, Indonesia
13515144@std.stei.itb.ac.id

***Abstrak*—Teka teki mengisi angka, atau "*number fill-in*" merupakan permainan teka-teki dimana disediakan suatu daftar angka dan matriks yang selnya terdiri dari sel hitam dan putih. Tujuan dari permainan ini adalah mengisi sel-sel tersebut dengan seluruh angka yang disediakan, dimana 1 sel hanya diisi oleh 1 digit. Strategi algoritma adalah salah satu materi pada ranah informatika. Salah satu algoritma yang dipelajari adalah algoritma runut balik. Makalah ini akan membahas tentang penyelesaian teka-teki mengisi angka dengan menggunakan pendekatan algoritma runut balik. Algoritma runut balik digunakan saat melakukan pengisian angka-angka pada matriks. Dijelaskan pula tentang komplesitas waktu yang dibutuhkan dan perbandingannya dengan algoritma *brute force*.**

Kata kunci—mengisi angka, kompleksitas, runut-balik

I. PENDAHULUAN

Manusia adalah makhluk yang memiliki banyak aktivitas, seperti bekerja, belajar, dan tugas-tugas lainnya. Semua aktivitas tersebut tentunya dapat membuat manusia menjadi lelah dan penat. Untuk melapaskan kepenatan tersebut, salah satu aktivitas yang dapat dilakukan adalah bermain permainan atau teka-teki. Salah satu teka-teki yang terkenal adalah teka-teki mengisi angka. Berdasarkan referensi [3], teka-teki mengisi angka, atau "number fill-in" adalah salah satu variasi lain dari teka-teki silang. Pada teka-teki silang pada umumnya, yang diberikan adalah klu untuk menebak kata yang tepat. Namun, pada permainan mengisi angka yang disediakan adalah daftar angka-angka jawaban. Teka-teki ini biasanya terdapat pada majalah disamping teka-teki lain seperti mencari kata, kriptografi, Sudoku, teka-teki silang atau teka-teki logika lainnya. Penyelesaian teka-teki ini biasanya membutuhkan percobaan (*trial and error*). Daftar angka yang diberikan terurut berdasarkan banyaknya digit yang membentuknya. Diberikan pula suatu matriks, yang terdiri dari sel hitam dan sel putih. Sel hitam tidak dapat diisi, sedangkan sel putih harus diisi. Tujuan permainan ini adalah mengisi seluruh sel putih pada matriks dengan kata-kata yang telah disediakan. Satu sel putih hanya diisi dengan 1 digit. Penyelesaian teka-teki ini adalah mencari angka dengan panjang yang cocok dengan suatu tempat tersedia. Jika terdapat angka awal yang diberikan, akan berguna sebagai titik awal pencarian. Jika terdapat tempat yang sudah terisi, akan dicari tempat pengisian lain yang beririsan dengan tempat tersebut dimana memiliki digit yang sama. Jika tidak terdapat angka lain dengan panjang yang sesuai yang dapat diisi di tempat tersebut, berarti angka sebelumnya yang telah diisi tidak berada pada tempat yang tepat. Angka dengan digit penyusun sedikit lebih mudah untuk diisikan, tetapi angka yang panjang lebih sulit dan membutuhkan klu angka lain unuk menyambungkannya.



Gambar 1 Ilustrasi Teka-Teki Mengisi Angka
Sumber:
http://cdn.shopify.com/s/files/1/0885/5306/products/printable-number-fill-in-puzzles_Page_2_grande.jpg?v=1433997766
(diakses 15 Mei 2017, Pk.20.43)

II. DASAR TEORI

A. *Pengertian Algortima Runut-balik*

Penjelesan dasar teori tentang algoritma runut-balik didasarkan pada referensi [1]. Algoritma runut balik

(*backtracking*) adalah algortima yang berbasis pada *Depth First Search* (DFS) untuk mencari solusi persoalan yang lebih mangkus. Runut balik, yang merupakan perbaikan dari algortima *brute-force*, secara sistematiks mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Hal ini berimplikasi pada waktu pencarian yang dapat dihemat. Runut balik lebih alami dinyatakan dalam algortima rekursif. Kadang-kadang disebutkan pula bahwa runut balik merupaka tipikal dari algortima rekursif.

Istilah runut-balik pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Selanjutnya, R.J Walker, Golomb, dan Baumer menyajikan uraian umum tentang runut-balik dan penerapannya pada berbagai persoalan. Saat ini algoritma runut-balik banyak diterapkan pada program *games*, seperti permainan *tic-tac-toe*, mencari jalan keluar labirin, catur, dsb. Selain itu, juga sering diterapkan pada bidang kecerdasan buatan (*artificial intelligence*).

B. Properti Umum Metode Runut-balik

Untuk menerapkan runut-balik, terdapat beberapa definisi properti berikut:

1. Solusi persoalan

   Solusi dinyatakan sebagai vector dengan n-tuple

   $X=(x_1,x_2,\ldots,x_n)$, $x_i \in$ himpunan berhingga Si. Mungkin saja $S_1=S_2=\ldots=S_n$.

2. Fungsi pembangkit nilai x_k

   Dinyatakan dengan T(k). T(k) membangkitkan nilai untuk xk, yang merupakan komponen vector solusi.

3. Fungsi pembatas

   Dinyatakan sebagai $B(x_1,x_2,\ldots x_k)$. Fungsi pembatas menentukan apakah (x_1, x_2, x_3, …, x_k) mengarah pada solusi. Jika ya, pembangkitan ilia untuk x_{k+1} dilajutkan, tetapi jika tidak, $(x_1,x_2,x_3,\ldots x_k)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi

C. Pengorgaisasian Solusi

Semua kemungkinan solusi dari persoalan disebut dengan ruang solusi. Seacra formal dapat dinyatakan, bahwa jika $x_i \in$ S, maka

$$S_1 \times S_2 \times \ldots \times S_n$$

disebut ruang solusi. Jumlah aggota di dalam ruang solusi adalah $|S_1| \cdot |S_2| \cdot \ldots \cdot |S_n|$.

Penyelesaian secara *exhaustive search* adalah degan menguji setiap kemungkinan solusi. Setiap kemungkinan solusi diperiksa apakah ia memenuhi kendala (yakni menjadi solusi yang layak). Solusi layak yang memberikan nilai fungsi obyektif.

Algoritma runut-balik memperbaiki pencarian solusi secara *exhaustive search* dengan mencari solusi secara sistematis. Untuk memfasilitasi pencarian ini, maka ruang solusi diorganisasikan ke dalam struktur pohon. Tiap simpul pohon menyatakan status persoalan, sedangkan sisi/cabang merupakan x_i. Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*).

D. Prinsip Pencarian Solusi dengan Metode Runut-balik

Akan ditinjau pencarian solusi pada pohon ruang status yang dibangun secara dinamis. Langkah-langkah pencarian solusi adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adakah mengikuti metode pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*expand-node*). Simpul dinomori dari atas ke bawah sesuai urutan kelahiran.
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambag panjag. Jika lintasan yang sedang dibentuk tidak mengarah pada solusi, maka simpul-E tersebut "dibunuh" sehigga enjadi simpul mati. Fungsi yang digunakan untuk membuuh simpul-E adalah dengan menerapkan fungsi pembatas (*bounding function).* Simpul yang sudah mati tidak akan pernah diperluas lagi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membagkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut-balik ke simpul hidup terdekat (simpul orang tua). Selanjutnya simpul ini menjadi simpul-E yang baru, Lintasan baru dibangun bila kita telah menemukan solusi atau tidak ada lagi simpul hiudp untuk runut balik.
4. Pencarian dihentuikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut balik.

E. Skema Umum Algoritma Runut-balik

Di bawah ini disajikan skema umum algoritma runut balik dalam dua versi, yaitu versi rekursif dan iteratif. Skema rekursif lebih tepat digunakan karena algoritma runut balik lebih alami dinyatakan dalam bentuk rekursi. Algoritma di bawah ini akan menghasilkan semua solusi:

1. Versi rekursif

```
procedure RunutBalikR (input k:integer)
{Mencari semua solusi persoalan dengan
metode runut balik dengan skema rekursif.
Masukan: k, yaitu indeks komponen vector
solusi, x[k]
Keluaran: solusi x= (x[1], x[2], …, x[k])}

Algoritma:
for tiap x[k] yang belum dicoba sedemikian
sehingga (x[k]←T(k)) and B(x[1], x[2],…,
x[k])=true do
  if (x[i],x[2],…,x[k]) adalah lintasan
  dari akar ke daun then
    CetakSolusi(x)
  endif
  RunutBalik(k+1)
Endfor
```

Pemanggilan prosedur pertama kali: `RunutBalikR(1)`

2. Versi iteratif

```
procedure RunutBalikI (input k:integer)
{Mencari semua solusi persoalan dengan
metode runut balik dengan skema iteratif.
Masukan: k, yaitu indeks komponen vector
solusi, x[k]
Keluaran: solusi x= (x[1], x[2], …, x[k])}

Algoritma:
k←1
while k>0 do
  if (x[k] belum dicoba sedemikian sehingga
  x[k]←T(k)) and B(x[1], x[2],…, x[k])=true
  then
    if (x[i],x[2],…,x[k]) adalah lintasan
    dari akar ke daun then
      CetakSolusi(x)
    endif
    k←k+1 {indeks anggota tuple berikutnya}
  else {tidak mengarah jalur solusi}
    k←k-1
  endif
endwhile
{k=0}
```

Pemanggilan prosedur pertama kali: `RunutBalikI(k)`

Prosedur cetak solusi adalah sebagai berikut:

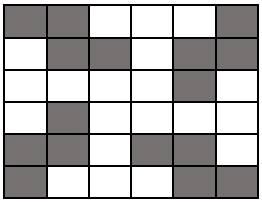
```
Procedure CetakSolusi (input x: TabelInt)
{Mencetak solusi persoalan
Masukan: x[1],x[2],…,x[n]
Keluaran: nilai x[1],x[2],…,x[n] tercetak
pada layar}

Deklarasi
  K:integer

Algoritma
for k←1 to n do
  write x[k]
endfor
```

Setiap simpul dalam pohon ruang status berasosiasi dengan sebuah pemanggilan rekursif. Jika jumlah simpul dalam pohon ruang status adalah 2^n atau n!, maka untuk 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 berderajat n yang menyatakan waktu komputasi setiap simpul.

III. PENYELESAIAN TEKA-TEKI MENGISI ANGKA DENGAN ALGORITMA RUNUT-BALIK

3.1 Langkah penyelesaian

Dalam pencarian solusi, algoritma runut balik digunakan dalam proses pengisian sel dengan angka-angka yang sesuai. Pada pengisian selanjutnya akan dicari apakah ada angka dari daftar angka yang sesuai pada suatu slot tempat tertentu, Angka dikatakan sesuai apabila panjangnya sama dengan panjang slot dan jika ada dua slot yang bersinggungan (menggunakan sel yang sama) juga menggunakan digit yang sama. Jika ada angka sesuai proses dilanjutkan pengisian slot tempat berikutnya. Jika tidak ada, berarti proses yang telah dijalankan tidak mengarah pada solusi. Sehingga, dilakukan proses *backtracking* dengan mengganti angka terakhir yang diisi sebelumnya. Untuk mengetahui lebih jelas tentang penggunaan algoritma runut-balik ini, penulis menggunakan salah satu contoh teka-teki mengisi angka. Misalnya diberikan persoalan sebagai berikut:

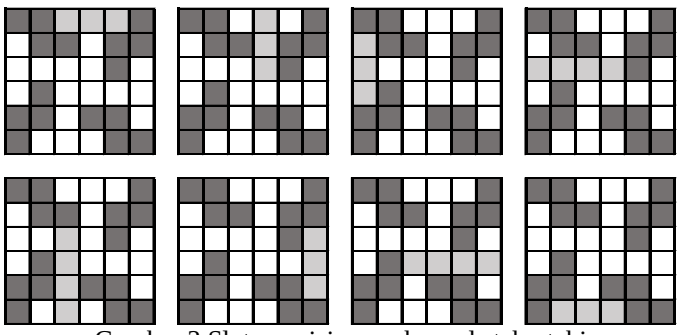


Panjang 3	Panjang 4
283	2981
378	7538
425	8423
617	9892

Gambar 2 Contoh teka-teki mengisi angka

Langkah pertama yang dilakukan adalah membuat *array boolean* yang menunjukkan apakah suatu angka sudah berada pada matriks atau belum. Selanjutnya, akan ditentukan slot tempat pengisian angka. Dilakukan iterasi dari sel (0,0) hingga (5,5) dan dilakukan pengecekan apakah sel tersebut adalah sel awal dari slot pengisian. Terdapat 2 syarat sel dikatakan sel awal slot. Pertama, apabila di sebelah kirinya bukan sel putih (untuk slot mendatar) atau di atasnya bukan sel putih (untuk menurun).Kedua, apabila terdapat minimal 1 sel putih lain di sebelah kanannya (untuk mendatar) atau di bawahnya (untuk menurun). Selanjutnya ditentukan pula jenis arah pengisian slot (mendatar atau menurun) dan panjang slot. Jumlah slot ini akan sama denan jumlah angka yang harus diisikan). Dari soal sebelumnya didapatkan daftar slot adalah sebagai berikut:

Slot	Sel	Arah	Panjang
1	(0,2)	mendatar	3
2	(0,3)	menurun	4
3	(1,0)	menurun	3
4	(2,0)	mendatar	4
5	(2,2)	menurun	4
6	(2,5)	menurun	3
7	(3,2)	mendatar	4
8	(5,1)	mendatar	3



Gambar 3 Slot pengisian angka pada teka-teki

Pengisian angka ke dalam sel yang tersedia dilakukan secara rekursif dengan urutan sesuai pada tabel urutan slot. Pada setiap tahap pengisian dilakukan langkah berikut:

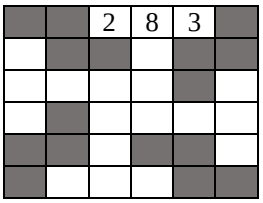
1. Mencari angka yang belum digunakan yang memiliki panjang sama dengan panjang slot

2. Mengecek apakah angka tersebut sesuai dengan angka yang sudah ada sebelumnya. (Sel yang berpotogan menggunakan digit yang sama)
3. Jika ya, isi slot dengan angka tersebut, dan ubah *array boolean* angka tersebut menjadi *true*. lanjutkan proses pengisian untuk slot berikutnya (kembali ke lagkah 1)
4. Jika tidak, akan dilakukan backtracking, dengan mengganti slot sebelumnya yang telah terisi dengan angka lain. Angka pada slot yang dihapus kembali diubah status *array boolean*-nya menjadi *false*.

Jika dilihat dari pohon ruang status, simpul akar adalah teka-teki soal, yaitu matriks yang kosong, dan setiap melakukan pengisian akan terbetuk simpul baru.

Berikutnya akan ditunjukkan implementasi langkah pengisian pada contoh teka-teki mengisi angka yang telah diberikan:

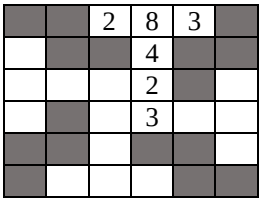
1. Slot pertama memiliki panjang 3, dan angka pertama yang memiliki panjang 3 adalah "283". Isikan angka 283 pada slot 1.



Panjang 3	Panjang 4
~~283~~	2981
378	7538
425	8423
617	9892

Gambar 4 Pembangkitan simpul 2

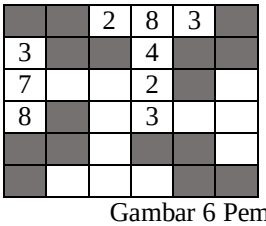
2. Slot kedua memiliki panjang 4. Angka pertama yang memiliki panjang 4 adalah 2981. Namun 2981 tidak cocok karena digit pertamanya berpotongan dengan digit kedua slot 1 yaitu 8. Sehingga dicari angka berikutnya yang sesuai, yang digit pertamanya adalah 8, yaitu "8423". Isikan "8243" pada slot 2.



Panjang 3	Panjang 4
~~283~~	2981
378	7538
425	~~8423~~
617	9892

Gambar 5 Pembangkitan simpul 3

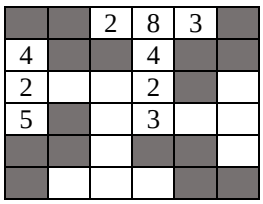
3. Berikutnya, slot 3 memiliki panjang 3. Angka pertama yang sesuai adalah 378. Isikan 378 pada slot 3.



Panjang 3	Panjang 4
~~283~~	2981
~~378~~	7538
425	~~8423~~
617	9892

Gambar 6 Pembangkitan simpul 4

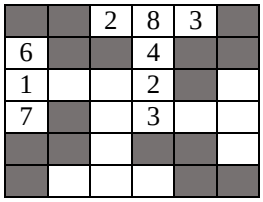
4. Pada pengisian slot 4 (berpanjang 4), tidak ada angka yang panjangnya 4 yang sesuai dengan slot tersebut (digit pertama 7, digit keempat 2), sehingga dilakukan runut-balik. Runut-balik dilakukan dengan mengganti angka yang digunakan pada slot sebelumnya (yaitu slot 3) dengan angka lain. Angka lain hasil penggantian yang sesuai adalah 425.



Panjang 3	Panjang 4
~~283~~	2981
378	7538
~~425~~	~~8423~~
617	9892

Gambar 7 Pembangkitan simpul 5

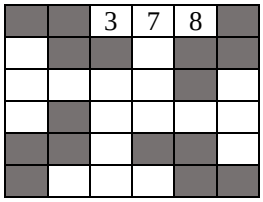
5. Ternyata, masih tidak ada angka yang sesuai pada slot 4. Sehingga, slot 3 kembali dilakukan pergantian angka. Angka berikutnya yang dicoba 617.



Panjang 3	Panjang 4
~~283~~	2981
378	7538
425	~~8423~~
~~617~~	9892

Gambar 8 Pembangkitan simpul 6

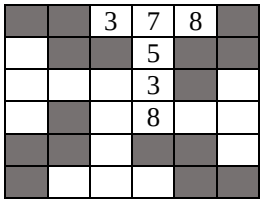
6. Kembali tidak ditemukan angka yang sesuai untuk slot 4, namun juga tidak ada lagi angka yang yang belum dicoba untuk mengisi slot 3. Sehingga, dilakukan runut-balik dengan menghapus slot 3. Namun, tidak ada lagi angka yang dapat digunakan untuk mengisi slot 2. Sehingga, kembali dilakukan runut-balik dengan mengganti angka pada slot 1. Slot 1 diganti dengan angka berikutnya, yaitu 378.



Panjang 3	Panjang 4
283	2981
~~378~~	7538
425	8423
617	9892

Gambar 9 Pembangkitan simpul 7

7. Slot 2 memiliki panjang 4, angka yang sesuai dengan slot tersebut adalah 7538.

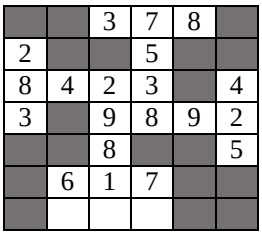


Panjang 3	Panjang 4
283	2981
~~378~~	~~7538~~
425	8423
617	9892

Gambar 10 Pembangkitan simpul 8

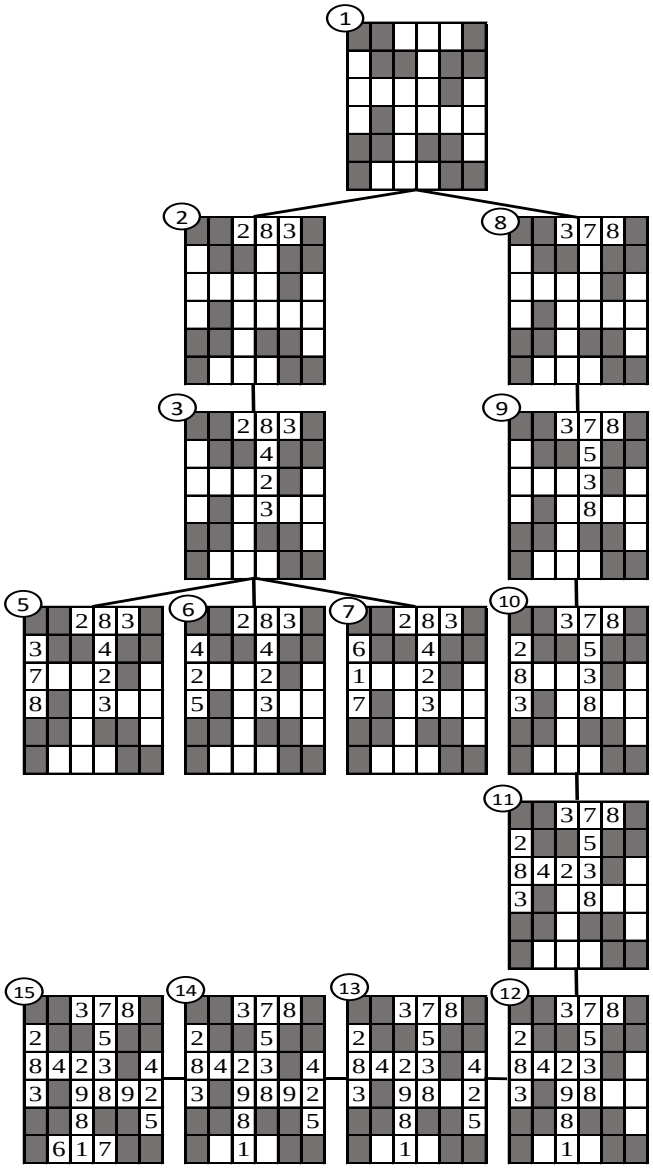
8. Langkah pengisian dan runut-balik ini terus dilakukan hingga seluruh slot / matriks terisi dengan semua angka pada daftar.

Setelah proses tersebut dilakukan hingga seluruh slot/matriks terisi, akan didapatkan solusi sebagai berikut:



Gambar 11 Solusi contoh teka-teki mengisi angka

Jika dilihat dari rohon ruang solusi, akan dibangkitkan sebanyak total 15 simpul hingga ditemukan solusi. Pohon ruang simpul yang terbentuk ditunjukkan pada gambar berikut:



Gambar 12 Pohon ruang status untuk teka-teki mengisi angka

3.2 Penghitungan kompleksitas

Penghitungan kompleksitas waktu didapatkan dari jumlah pemeriksaan terhadap himpunan solusi yang telah ada. Dapat pula dilihat dari jumlah simpul yang dibangkitkan pada pohon ruang status. Jika pengerjaan dilakukan dengan algoritma *brute force* dengan *exhaustive search*, akan dicari semua kombinasi pengisian angka pada slot. Sehingga kompleksitasnya adalah O(n!), dimana n adalah jumlah angka yang harus diisi. Sehingga, kompleksitas waktu pada teka-teki contoh adalah 8! = 40320. Cara lain untuk memotong waktu algoritma exhaustive *bruteforce* adalah dengan hanya menghitung kombinasi angka di slot yang memang panjangnya sesuai, sehingga kompleksitasya adalah

$m_i \times m_{i+1} \times \ldots$

dengan m_i adalah banyaknya angka dengan i digit. Sehingga kompleksitas pada contoh adalah banyaknya angka 3 digit x banyaknya angka 4 digit = 4! X 4! = 576.

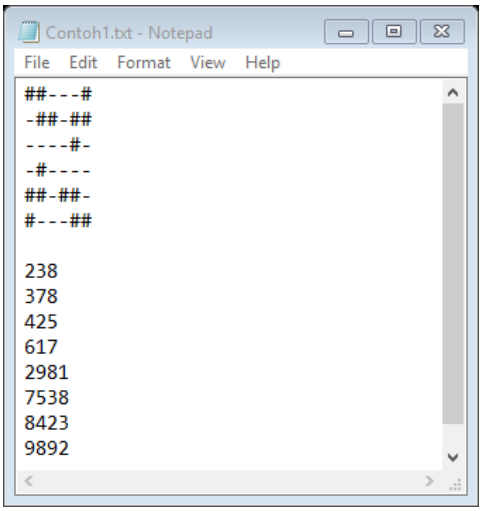
Sedangkan, jika menggunakan algoritma runut-balik, dilihat dari pohon ruang status, hanya diperlukan pembangkitan 15 simpul. Hal ini disebabkan karena pada algoritma *backtracking*, tidak semua kemungkinan pengisian angka dicoba. Hanya kombinasi yang mengarah pada solusi yang dilanjutkan. Dengan kompleksitas waktu 15, berarti algoritma runut-balik ini lebih cepat sekitar 576/15 ≈ 38 kali. Namun kasus terburuk dari algoritma runut-balik ini dapat terjadi jika solusi baru ditemukan setelah mencoba semua kemungkinan. Untuk kasus terburuk, simpul yang terbentuk yaitu banyaknya angka 3 digit dikalikan banyaknya angka 4 digit = 4! X 4! = 576.

3.3 Uji coba pada program

Berdasarkan langkah-langkah yang telah dijelaskan sebelumnya, penulis mencoba untuk membuat program untuk menyelesaikan teka-teki mengisi angka ini. Program diimplementasikan dalam bahasa Java, dan input dibaca dari file eksternal dengan format .txt yang berisi matriks pengisian dan daftar angka yang harus ditempatkan pada matriks. Struktur data yang digunakan dalam penyusnan program ini antara lain:

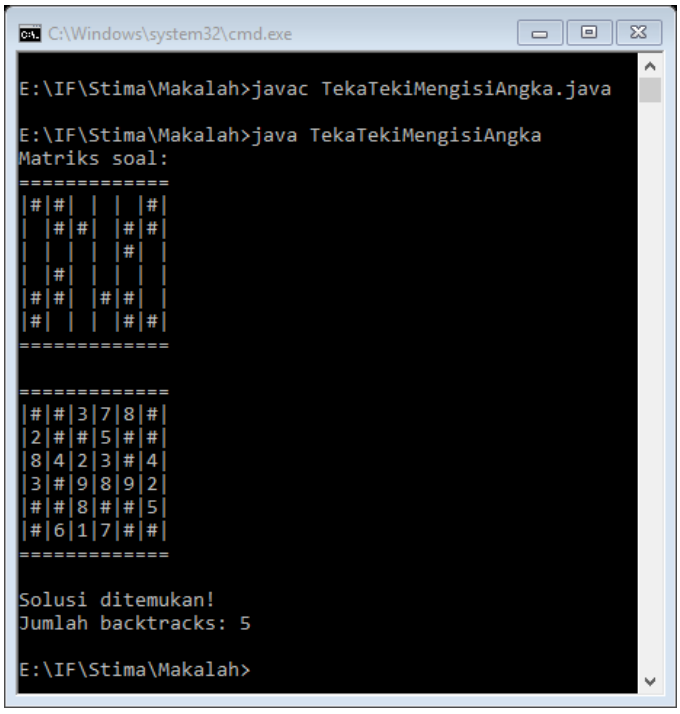
1. *Matrix of char*, yang digunakan sebagai tempat mengisi angka-angka
2. Tipe bentukan *DaftarAngka* yang memuat *array of integer* berisi daftar angka, dan *array of integer* berisi panjang dari setiap kata tersebut
3. Tipe bentukan *slot* yang menyimpan posisi awal slot, arah, dan panjang slot. Slot ini disatukan dalam sebuah *array of slot.*

Pada *file* masukan, karakter '#' melambangkan sel hitam dan karakter '-' melambangkan sel putih. Berikut adalah masukan dari program:



Gambar 13 Masukan program teka-teki mengisi angka

Program akan menampilkan matriks soal dan solusi dari persoalan tersebut. Selain itu akan menampilkan pesan apakah solusi ditemukan atau tidak dan juga jumlah runut-balik yang dilakukan. Berikut adalah keluaran dari program:



Gambar 14 Keluaran program teka-teki mengisi angka

IV. KESIMPULAN

Penyelesaian suatu persoalan dapat diselesaikan dengan berbagai agoritma untuk menyelesaikannya. Salah satunya adalah algoritma runut balik. Salah satu penyelesaian yag dapat memanfaatkan algorima ini adalah penyelesaian teka-teki megisi angka. Jika dibandingkan dengan algoritma *brute force*, algoritma *backtracking* ini lebih mangkus untuk meyelesaikan teka-teki mengisi angka ini. Hal ini disebabkan karena tidak semua kombinasi solusi akan dicoba seperti pada *exhaustive search*. Pada algoritma runut-balik, hanya kombinasi yang megarah pada solusilah yang dilanjutkan. Jika suatu kombinasi tidak mengarah pada solusi, pencarian akan dihentikan dan dilakukan runut balik. Walaupun belum mencoba algoritma lain, penulis berpendapat bahwa algoritma runut-balik ini sudah cukup mangkus untuk menyelesaikan teka-teki mengisi angka ini. Karena keefektifannya, algoritma runut-balik banyak diterapkan pada penbuatan *game* atau permasalahan yang berhubungan dengan kecerdasan buatan (*artificial intelligence*).

V. UCAPAN TERIMA KASIH

Ucapan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, atas rahmat dan penyertaan-Nya makalah ini dapat diselesaikan. Penulis juga berterima kasih kepada Bapak Rinaldi Munir, Ibu Masayu Leyla Khodra, dan Ibu Nur Ulfa Maulidevi selaku dosen mata kuliah IF2211 Strategi Algortima untuk waktu dan ilmu yang diberikan. Tidak lupa penulis berterima kasih kepada orang tua dan teman-teman atas bantuan dan dukungannya.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2007. Diktat Kuliah IF2251. Strategi Algoritmik. Teknik Informatika ITB : Bandung

[2] Noam M.Shazeer, Michael L.Littman, Greg A.Keim. 1999. “Solving Crossword Puzzles as Probabilistic Constraint Satisfication”. Duke University, Durham

(diakses dari www.aaai.org/Papers/AAAI/1999/AAAI99-023.pdf?origin pada 17 Mei 2017 pukul 20.15)

[3] http://www.bigopolis.com/BigOp/fill-it-ins/xrisxros_help.html

(diakses pada 17 Mei 2017 pukul 17.00)

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, 18 Mei 2017

William / 13515144