Penerapan Algoritma Backtracking dalam Permainan Mainarizumu

Ruhiyah Faradishi Widiaputri/ 13519034
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519034@std.stei.itb.ac.id

Abstrak*—algoritma *backtracking* merupakan algoritma yang merupakan perbaikan dari *exhaustive search*. Dengan menggunakan algoritma *backtracking* banyak permasalahan yang dapat diselesaikan dengan lebih efektif, khususnya *constraint satisfaction problem* seperti pada permainan *logic puzzle.* Makalah ini akan membahas mengenai bagaimana penerapan algoritma *backtracking* untuk memecahkan persoalan permainan *puzzle* asal Jepang yang bernama *Mainarizumu.

***Keywords—backtracking, mainarizumu,* batasan, pohon ruang status, baris, kolom**

I. PENDAHULUAN

Permainan *puzzle* merupakan salah satu permainan yang disukai banyak orang dari segala rentang usia. Permainan ini cocok dimainkan di waktu senggang. Tidak hanya menyenangkan, bermain *puzzle* juga memiliki banyak manfaat seperti meningkatkan fungsi kognitif, meningkatkan daya ingat, meredakan stress, dan meningkatkan kemampuan berpikir matematis. Salah satu jenis *puzzle* yang terkenal adalah puzzle logika (*logic puzzle*). *Logic puzzle* biasanya terdiri dari kisi-kisi yang harus diselesaikan dengan mengikuti aturan tertentu. Contoh dari *puzzle* jenis ini adalah sudoku, nonogram, dan mainarizumu.

Mainarizumu adalah sebuah *puzzle* logika dari Jepang yang diselesaikan dengan mengisikan angka ke dalam kisi-kisi sesuai dengan aturan tertentu. Aturan dalam permainan *mainarizumu* mirip seperti pada permainan *futoshiki* namun pada *mainarizumu* terdapat aturan tambahan yaitu mengenai selisih antara dua nilai sel yang berdekatan.

Ada banyak cara menyelesaikan permainan *mainarizumu*, salah satunya adalah dengan mencoba semua kemungkinan angka (dengan *exhaustive search*). Namun cara ini kurang efektif dan membutuhkan waktu yang lebih lama. Cara yang lebih efektif dalam menyelesaikan permainan ini adalah dengan menguunakan algoritma *backtracking*. Algoritma ini adalah perbaikan dari metode *exhaustive search.* Dengan skema umum algoritma *backtracking* permainan *puzzle* dapat diselesaikan dengan lebih efisien, baik oleh manusia maupun oleh komputer.

II. LANDASAN TEORI

A. Algoritma Backtracking

Backtracking dapat dilihat sebagai sebuah fase di dalam algoritma traversal DFS maupun sebagai metode pemecahan masalah yang mangkus dan sistematis dengan pendekatan yang merupakan perbaikan dari *brute force*. Perbedaan algoritma runut-balik ini dengan *brute force* adalah pada algoritma ini hanya pilihan yang mengarah ke solusi yang dieksplorasi. Algoritma backtracking pertama kali ditemukan oleh D.H. Lehmer pada tahun 1950. Terdapat tiga tipe masalah yang dipecahkan dengan algoritma *backtracking*, yaitu masalah keputusan (*decision problem* – mencari solusi yang mungkin), masalah optimisasi (mencari solusi terbaik), dan masalah enumerasi (mencari semua solusi yang mungkin). Algoritma *backtracking* umumnya digunakan untuk memecahkan *constraint satisfaction problem*

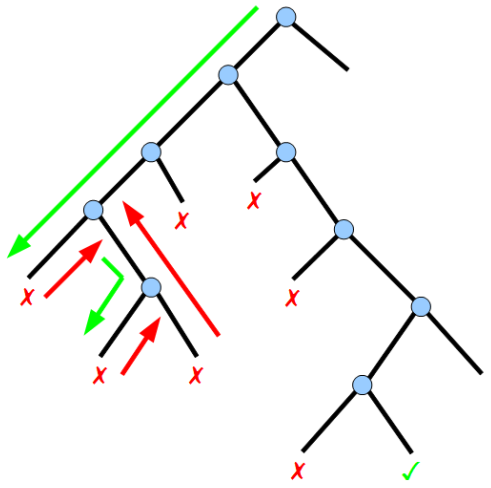
Ada tiga properti umum dalam algoritma runut-balik (*backtracking*) sebagai berikut.

- Solusi persoalan yang dinyatakan sebagai vector n-*tuple* $X = (x_1, x_2, x_3, \ldots, x_n)$ dengan $x_i \in S_i$
- Fungsi pembangkit, yaitu predikat $T(x_1, x_2, x_3, \ldots, x_{k-1})$ yang membangkitkan nilai untuk x_k yang merupakan komponen vektor solusi.
- Fungsi pembatas, yaitu predikat $B(x_1, x_2, x_3, \ldots, x_k)$ yang akan bernilai benar jika $(x_1, x_2, x_3, \ldots, x_k)$ mengarah ke solusi. Pembangkitan nilai untuk x_{k+1} akan dilanjutkan jika nilai $B(x_1, x_2, x_3, \ldots, x_k)$ benar.

Semua kemungkinan solusi persoalan (ruang solusi) diorganisasikan ke dalam struktur pohon berakar yang disebut pohon ruang status (*state space tree*). Tiap simpul pohon menyatakan status persoalan sedangkan sisi dilabeli lambang x_i.

Aturan pembangkitan simpul pada algoritma runut-balik adalah mengikuti aturan DFS (*Depth Search First*). Simpul hidup adalah simpul-simpul yang sudah dibangkitkan. Adapun simpul ekspan adalah simpul hidup yang sedang diperluas. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi simpul ekspan akan dimatikan dan proses pencarian *backtrack* ke simpul pada aras di atasnya. Ketika suatu simpul dimatikan maka simpul-simpul anaknya dalam hal ini secara mplisit telah

dipangkas. Pemangkasan inilah yang membedakan algoritma runut-balik dengan *exhaustive search*.



Gambar 1 Prinsip Pencarian Solusi dengan Algoritma Runut-balik

Sumber: https://www.w3.org/2011/Talks/01-14-steven-phenotype/

Algoritma runut balik dapat diselesaikan secara rekursif maupun iteratif. Skema umum algoritma runut-balik secara rekursif adalah sebagai berikut.

```
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 2 Skema Umum Algoritma Runut Balik Versi Rekursif

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

Adapun secara iteratif algoritma runut-balik memiliki skema umum sebagai berikut.

```
procedure RunutBalikI(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema iteratif
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:
  while k ≠ 0 do
     if terdapat nilai x[k] yang belum dicoba sedemikian sehingga x[k] ∈ T(x[1], x[2], …, x[k – 1]) and
        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
        k ← k + 1      { tentukan nilai x[k] selanjutnya }
     else
        k ← k – 1
     endif
  endwhile
```

Pemanggilan pertama kali: RunutBalikI(1)

Gambar 3 Skema Umum Algoritma Runut Balik Versi Iteratif

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

Beberapa contoh permasalahan yang diselesaikan oleh algoritma *backtracking* adalah persoalan N-ratu (*the N-queens problem*), *integer knapsack problem*, *sum of subsets problem*, pewarnaan graf, permasalahan labirin, dan sirkuit hamilton.

B. Constraint Satisfaction Problem

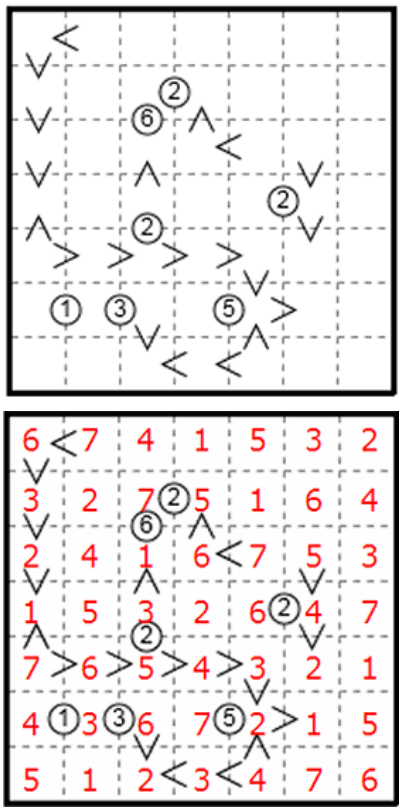
Constraint Satisfaction Problem (CSP) adalah proses menemukan solusi untuk suatu set masalah di mana suatu *state* harus memenuhi persyaratan atau kriteria tertentu. CSP terdiri dari satu set variabel, domain nilai untuk setiap variabel, dan satu set kendala. CSP bertujuan untuk memberikan nilai untuk setiap variabel sehingga semua batasannya terpenuhi.

Jenis paling sederhana dari CSP melibatkan variabel diskrit dan domain yang berhingga, contohnya masalah pewarnaan graf dan masalah N-ratu. Beberapa contoh CSP lainnya adalah permasalahan *cryptarithmetics*, *sudoku*, *crossword*,dan permasalahan-permasalahan *logic puzzle* yang lain.

Logic puzzle adalah permainan teka-teki yang biasanya terdiri dari kisi-kisi yang harus diselesaikan dengan mengikuti aturan tertentu. Contoh dari *puzzle* jenis ini adalah *nonogram*, *hutoshiki*, dan *mainarizumu*.

C. Mainarizumu

Mainarizumu(マイナリズム) adalah sebuah permainan teka-teki yang berasal dari Jepang. Permainan ini diperkenalkan oleh Nikoli. *Mainarizumu* dimainkan pada papan persegi yang berukuran 7 x 7 atau lebih kecil.



Gambar 4 Papan Permainan *Mainarizumu* sebelum dan sesudah dimainkan

Sumber: https://www.janko.at/Raetsel/Mainarizumu/002.a.htm

Pada permainan *Mainarizumu* pemain diminta untuk memasukkan angka ke setiap sel yang ada di papan. Angka yang dimasukkan adalah pada rentang 1 hingga N untuk papan yang berukuran N x N sedemikian sehingga tidak ada angka yang sama pada kolom maupun baris yang sama. Selain itu pada papan permainan *Mainarizumu* juga terdapat tanda ketidaksamaan '>', '<', dan sebuah angka. Jika ditemukan tanda ketidaksamaan di antara 2 sel maka kedua angka pada sel yang bersangkutan harus mematuhi tanda ketidaksamaan tersebut. Misalkan terdapat sel A, sel B, dan tanda '>' di antara kedua sel tersebut. Ini berarti bahwa sel A harus diisikan angka yang nilainya lebih besar daripada angka di sel B. Jika di antara 2 sel A dan B terdapat angka maka kedua angka pada sel A dan B harus memiliki selisih sebesar angka yang terdapat di antara kedua sel tersebut.

III. IMPLEMENTASI

A. *Properti Umum Algoritma Runut Balik pada Permainan Mainarizumu*

Seperti yang telah dijelaskan sebelumnya terdapat tiga properti umum algoritma runut balik, yaitu solusi persoalan, fungsi pembangkit, dan fungsi pembatas (*bounding function*). Berdasarkan aturan permainan *Mainarizumu* maka dapat diambil properti algoritma runut-balik penyelesaiannya yaitu sebagai berikut.

- Solusi umum

  Solusi umum permainan *Mainarizumu* berukuran N x N adalah berupa vektor n-*tuple* X = (x_1, x_2, x_3, …, x_n) dengan n = banyak sel (n = N x N) dan $x_i \in \{1, 2, …, N\}$. Hubungan antara nilai x_i dan sel-sel pada papan permainan *manarizumu* adalah seperti pada gambar di bawah ini.

x[1]	x[2]	. . .	x[N]
x[1*N+1]	x[1*N+2]	. . .	x[2*N]
. . .	. . .	. . .	. . .
x[(N-1)N+1]	x[(N-1)N+2]	. . .	x[N^2]

Gambar 5 Vektor Solusi Umum pada Kisi-kisi Papan Permainan *Mainarizumu*

Sumber: Dokumentasi Pribadi

- Fungsi pembangkit

  Berdasarkan aturan permainan *Mainarizumu* nilai yang dibangkitkan untuk setiap x_i adalah bilangan bulat dari 1 hingga N sehingga dalam penyelesaian permainan *mainarizumu* fungsi pembangkitnya akan mengembalikan nilai dari 1 hingga N ($T(x_1, x_2, x_3, …, x_{k-1}) = \{1,2,…,N\}$). Dengan demikian bagian skema umum algoritma *backtracking* pada gambar 2 yaitu:

```
for setiap x[k] ∈ T(x[1], x[2], …, x[k – 1])
do
    . . .
endfor
```

Dapat juga dilihat sebagai:

```
i traversal [1..N] do
    . . .
```

- Fungsi pembatas

  Dalam menyelesaikan permainan terdapat batasan-batasan yang harus diperhatikan sebagai berikut.

  - Tinjau dua posisi sel dengan angka yang sama (i,j) dan (m,n).
    - Kedua sel tidak berada pada kolom yang sama

      $$j \neq n$$

    - Kedua sel tidak berada pada baris yang sama

      $$i \neq m$$

  - Jika sebuah sel berada di posisi (i,j) maka perlu diperiksa tanda ketidaksamaan atau angka yang ada di dekat sel tersebut. Jika ada maka harus diperiksa apakah sel tersebut memenuhi tanda ketidaksamaan atau angka tersebut.

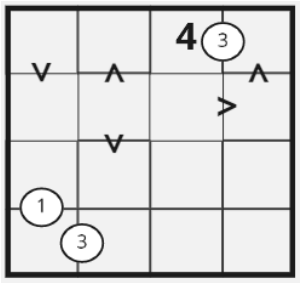
  Berdasarkan batasan-batasan di atas maka fungsi pembatas pada permainan ini secara garis besar adalah sebagai berikut.

```
function B(i: int, j: int, num: int)→
boolean
{i adalah baris sel yang akan dimasukkan
nilainya dengan num, j adalah kolom sel yang
akan dimasukkan nilainya dengan num}
    → ((tidak ada nilai pada kolom j yang
nilainya sama dengan num) and
          (tidak ada nilai pada baris i yang
nilainya sama dengan num) and
          (num memenuhi tanda ketidaksamaan
yang ada di kanan, kiri, atas, atau
bawahnya) and
          (num mematuhi ketentuan selisih
mutlak yang ada di sebelah, kanan, kiri,
bawah, atau atasnya))
```

B. *Penyelesaian Permainan Mainarizumu dengan Algoritma Backtracking*

Dengan peraturan yang telah dirumuskan ke dalam bentuk solusi umum, fungsi pembangkit, dan fungsi pembatas maka permainan *Mainarizumu* sudah siap diselesaikan dengan menggunakan algoritma *backtracking*. Berikut ini adalah beberapa contoh persoalan permainan *Mainarizumu* berikut penyelesaiannya dengan algoritma *backtracking*.

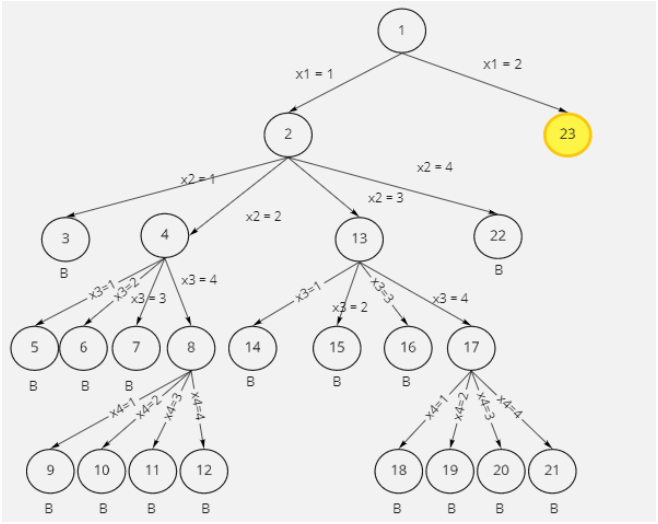
- Persoalan 1



Gambar 6 Contoh Persoalan 1 Permainan *Mainarizumu*

Sumber: Dokumentasi Pribadi

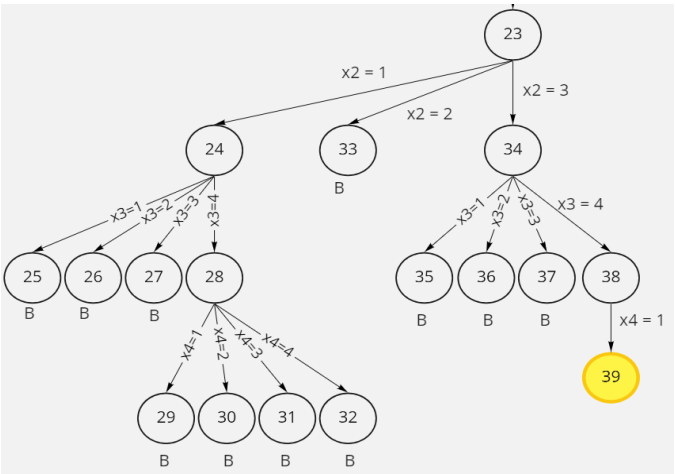
Langkah pertama yang dilakukan untuk menyelesaikan persoalan di atas adalah dengan membangkitkan nilai untuk x_1. Pertama kali x_1 diberi nilai 1. Nilai 1 untuk x_1 saat diperksa dengan *bounding function* tidak melanggar *constraint* sehingga pembangkitan nilai dilanjutkan untuk x_2. Pertama kali x_2 diberi nilai 1. Namun saat diperiksa dengan *bounding function* ternyata x_2 tidak dapat diberi nilai 1 karena dapat menyebabkan 2 buah nilai 1 pada baris pertama sehingga simpul pada pohon statusnya dimatikan (di sini diberi tanda B). x_2 kemudian diberi nilai 2. Karena saat diperiksa pemberian nilai 2 untuk x_2 masih menghasilkan *bounding function* yang bernilai benar maka pembangkitan nilai dilanjutkan untuk x_3. Untuk nilai 1,2, dan 3 yang diberikan kepada x_3 ketiga nilai tersebut menghasilkan keluaran *bounding function* menjadi *False* karena nilainya sudah ditetapkan di dalam pertanyaan menjadi 4. Untuk $x_3 = 4$ selanjutnya dilakukan lagi pemberian nilai untuk x_4. Namun setelah diperiksa ternyata tidak ada satupun pemberian nilai 1, 2, 3, atau 4 yang memberikan nilai keluaran *bounding function* yang benar (semua nilai di S_4 melanggar *constraint*) sehingga dilakukanlah *backtrack* untuk nilai x_3. Karena semua nilai di S_3 melanggar *constraint* juga maka dilakukan *backtrack* untuk nilai x_2. Nilai 2 kemudian diberi nilai 3 berdasarkan fungsi pembangkit (dalam hal ini nilainya diberi oleh *looping* terurut). Saat diperiksa ternyata pemberian nilai $x_2 = 3$ juga tidak melanggar *constraint* sehingga pemberian nilai dilanjut untuk x_3. Pemberian nilai dilanjutkan sama seperti sebelumnya dan pada akhirnya ditemukan bahwa untuk $x_{1\ =\ 1}$ tidak ada nilai yang memenuhi sehingga dilakukan *backtrack* untuk nilai x_1. Berdasarkan fungsi pembangkit (*looping* terurut) maka x_1 sekarang diberi nilai 2. Sejauh ini pohon solusi yang kita miliki adalah seperti pada gambar di bawah ini. Nilai di dalam simpul menunjukkan urutan pembangkitan dan simpul berwarna kuning adalah simpul ekspan hingga tahap ini.

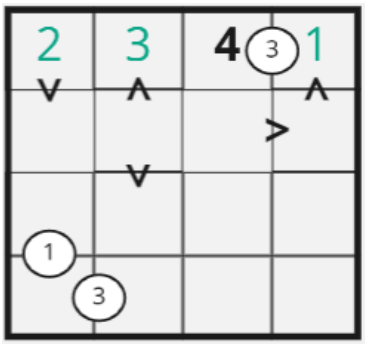


Gambar 7 Pohon Ruang Status Tahap 1

Sumber: Dokumentasi Pribadi

Dari simpul ke-23 kemudian dibangkitkan nilai untuk x_2, dimulai dari 1. Saat dilakukan pemeriksaan ternyata pemberian nilai 1 untuk x_2 menghasilkan keluaran *bounding function* yang bernilai True sehingga selanjutnya dilakukan pembangkitan nilai untuk x_3. Pemberian nilai 1, 2, dan 3 untuk x_3 melanggar *constraint*. Dengan x_3 yang diberi nilai 4 dilanjutkan pembangkitan nilai untuk x_4. Ternyata semua nilai S_4 melanggar *constraint* sehingga dilakukan *backtrack* untuk nilai x_3. Karena semua nilai S_3 kini melanggar *constraint* maka dilakukan *backtrack* untuk nilai x_2.. x_2 selanjutnya diberi nilai 2 tetapi ini melanggar *constraint* sebab menyebabkan 2 buah nilai 2 di paris pertama sehingga simpul ekspan dimatikan dan dibangkitkan simpul ekspan yang baru, yaitu untuk $x_2 = 3$. Kemudian dilakukan pembangkitan nilai untuk x_3. Untuk nilai 1,2, dan 3 yang diberikan kepada x_3 ketiga nilai tersebut menghasilkan keluaran *bounding function* menjadi *False* karena nilainya sudah ditetapkan di dalam pertanyaan menjadi 4. Untuk $x_3 = 4$ selanjutnya dilakukan lagi pemberian nilai untuk x_4. Pertama nilai x_4 adalah 1. Dengan nilai $x_4 = 1$ *bounding function* masih bernilai True sehingga selanjutnya akan dibangkitkan nilai untuk x_5. Pohon ruang status dari simpul ke-23 hingga saat ini beserta nilai sel-sel papan permainan sejauh ini adalah sebagai berikut.

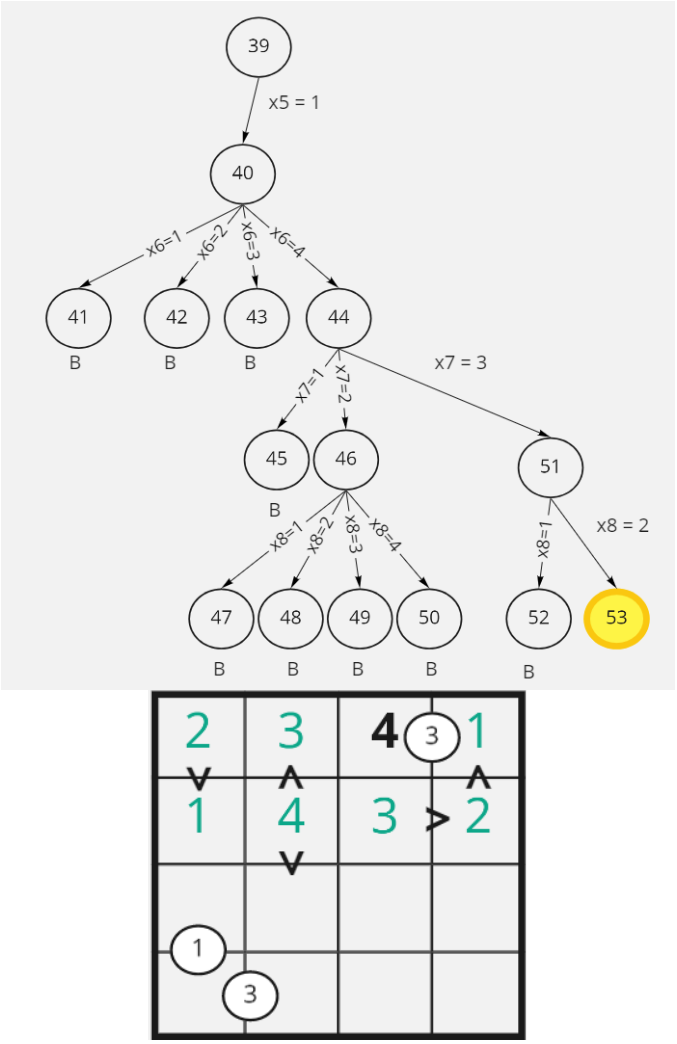




Gambar 8 Pohon Ruang Status Tahap 2 dan Sel-Sel Papan Permainan hingga Tahap 2

Sumber: Dokumentasi Pribadi

Dari simpul ke-39 kemudian dibangkitkan nilai untuk x_5, dimulai dari 1. Untuk nilai $x_5 = 1$ diperoleh hasil *bounding function* yang bernilai True sehingga pembangkitan nilai dilakukan untuk x_6. Pemberian nilai 1 untuk x_6 menghasilkan keluaran *bounding function* yang bernilai False karena menyebabkan ada 2 angka 1 di baris kedua sehingga dibangkitkan nilai lainnya untuk x_6.. $x_6 = 2$ juga melanggar *constraint* karena tidak memenuhi tanda ketidaksamaan (menyebabkan nilai $3 < 2$ yang tidak benar) sehingga dibangkitkan nilai yang lain. Untuk x_6 yang diberi nilai 3 juga diperoleh *bounding function* yang mengembalikan nilai Falsen karena tidak memenuhi tanda ketidaksamaan sehingga sekarang x_6 diberi nilai 4. Karena $x_6 = 4$ tidak melanggar *constraint* maka selanjutnya dilakukan pembangkitan nilai untuk x_7. Untuk nilai 1 tidak bisa diberikan kepada x_7 karena menyebabkan terdapat 2 buah nilai 1 di baris kedua. Untuk nilai selanjutnya yang diberikan kepada x_7 berdasarkan *looping* terurut yaitu 2 yang setelah diperiksa tidak melanggar *constraint* selanjutnya dilakukan pembangkitan nilai untuk x_8. Karena setelah diperiksa tidak ada satupun S_8 yang memenuhi untuk diberikan kepada x_8 maka simpul ekspan dimatikan dan dilakukan *backtrack* untuk x_7. Nilai x_7 selanjutnya adalah 3 dan dibangkitkan simpul ekspan yang baru. Nilai $x_7 = 3$ tidak melanggar *constraint* sehingga selanjutnya dilakukan pembangkitan simpul untuk nilai x_8. Untuk $x_8 = 1$ saat dilakukan pemeriksaan *bounding function* akan mengembalikan nilai False karena pemberian nilai $x_8 = 1$ akan menyebabkan 2 buah nilai 1 di kolom keempat sehingga simpul ekspan dimatikan dan dibangkitkan simpul ekspan baru untuk pemberian nilai 2 pada x_8. $X_8 = 2$ memenuhi semua *constraint* sehingga selanjutnya akan dibankitkan simpul untuk pemberian nilai x_9. Pohon ruang status dari simpul ke-39 hingga saat ini beserta nilai sel-sel papan permainan sejauh ini adalah sebagai berikut.

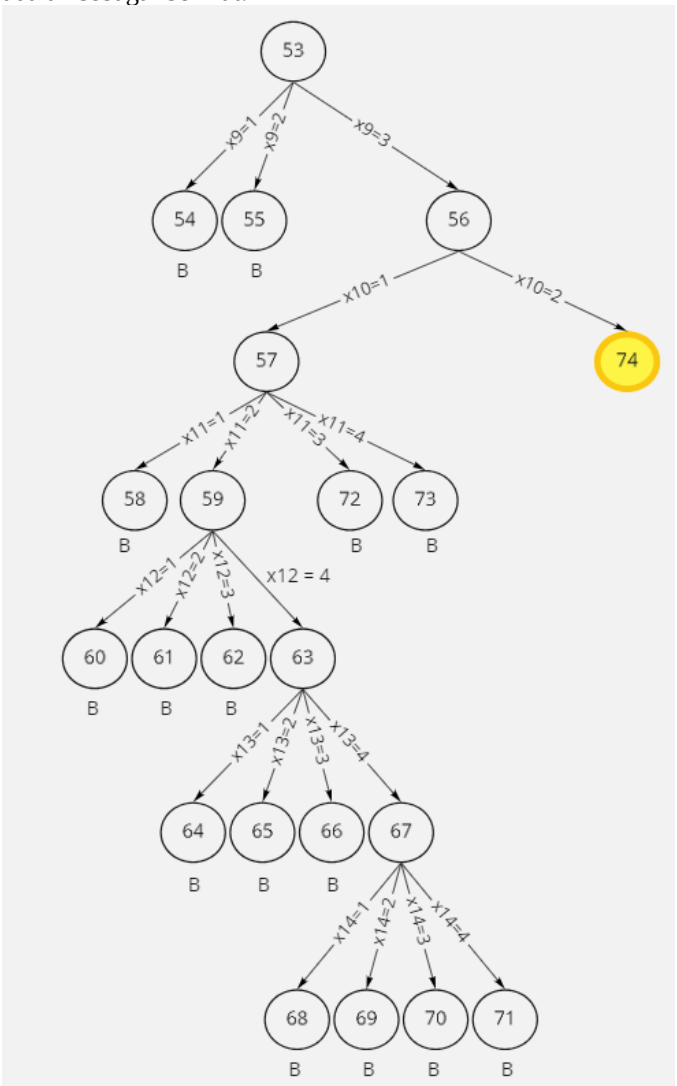


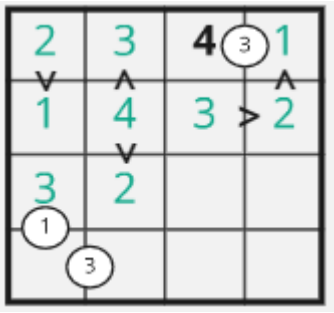
Gambar 9 Pohon Ruang Status Tahap 3 dan Sel-Sel Papan Permainan hingga Tahap 3

Sumber: Dokumentasi Pribadi

Dari simpul ke-53 dibangkitkan nilai untuk x_9, mulai dari 1. Nilai $x_9 = 1$ dan 2 melanggar *constraint* karena menyebabkan berturut-turut 2 nilai 1 dan 2 di kolom pertama sehingga dibangkitkan nilai yang baru untuk x_9 yaitu 3. $X_9 = 3$ masih menghasilkan keluaran *bounding function* True sehingga selanjutnya dibangkitkan nilai untuk x_{10}, mulai dari 1. Untuk $x_{10} = 1$ saat diperiksa dengan *bounding function* masih benar sehingga selanjutnya dilakukan pembangkitan untuk nilai x_{11}, mulai dari 1. Pemberian nilai $x_{11} = 1$ melanggar constraint karena menyebabkan ada 2 angka 1 di baris ketiga sehingga simpul ekspan dimatikan dan dibangkitkan simpul ekspan baru untuk pemberian nilai x_{11} yang lain yaitu 2. $X_{11} = 2$ tidak melanggar *constraint* sehingga dibangkitkan nilai untuk x_{12}. Untuk nilai 1,2, dan 3 saat diberikan kepada x_{12} menghasilkan keluaran *bounding function* False berturut-turut karena menyebabkan 2 angka 1 di kolom keempat, menyebabkan 2 buanh angka 2 di kolom keempat, dan menyebabkan 2 angka 3 di baris ketiga, tetapi kemudian saat $x_{12} = 4$ *bounding function* bernilai True sehingga selanjutnya dibangkitkan nilai untuk x_{13}. Untuk setiap nilai yang

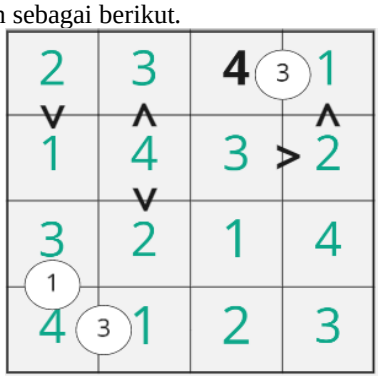
dibangkitkan untuk x_{13} hanya nilai 4 yang tidak melanggar *constraint* sehingga selanjutnya dibangkitkan nilai untuk x_{14}. Ternyata pemberian nilai 1 untuk x_{14} melanggar *constraint* karena menyebabkan 2 angka 1 di kolom kedua. Nilai 2, 3, dan 4 untuk x_{14} juga melanggar *constraint* karena menyalahi tanda ketidaksamaan yang terdapat di sebelah kiri sel yang bersesuaian dengan x_{14}. Karena tidak ada nilai di S_{14} yang memenuhi *bounding function*, maka dilakukan *backtrack* untuk nilai x_{13}. Karena tidak ada lagi nilai di S_{13} yang memenuhi *bounding function* maka dilakukan *backtrack* untuk x_{12}. Karena tidak ada lagi juga nilai di S_{12} yang memenuhi *bounding function* maka dilakukan *backtrack* untuk x_{11}. Setelah diperiksa untuk nilai-nilai selanjutnya yang diberikan kepada x_{11} yaitu 3 dan 4 tidak ada yang memenuhi *constraint.* Sama seperti sebelumnya maka simpul ekspan dimatikan dan dilakukan *backtrack* untuk x_{10}, artinya nilai x_{10} sekarang adalah 2. Untuk $x_{10} = 2$ saat dilakukan pengecekan terhadap *bounding function* didapatkan bahwa pemberian nilai 2 untuk x_{10} tidak melanggar *constraint* sehingga selanjutnya akan dibangkitkan simpul untuk nilai x_{11}. Pohon ruang status dari simpul ke-53 hingga saat ini beserta nilai sel-sel papan permainan sejauh ini adalah sebagai berikut.

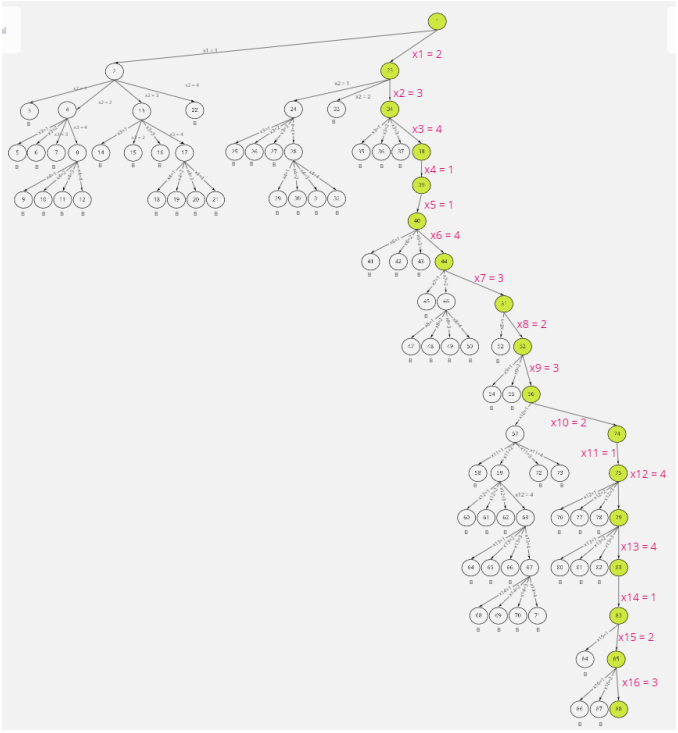




Gambar 10 Pohon Ruang Status Tahap 4 dan Sel-Sel Papan Permainan hingga Tahap 4

Dari simpul ke-74 dibangkitkan nilai untuk x_{11}, mulai dari 1. Pemberian nilai 1 untuk x_{11} tidak melanggar *constraint* sehingga selanjutnya dilakukan pembangkitan untuk x_{12}. Nilai 1, 2, dan 3 tidak dapat diberikan pada x_{12} karena pada baris ketiga sudah ada angka 1,2, dan 3 sehingga x_{12} diberikan nilai 4. Selanjutnya dilakukan pembangkitan nilai untuk x_{13}. Nilai 1, 2, dan 3 tidak dapat diberikan pada x_{13} karena pada kolom pertama sudah ada angka 1,2, dan 3 sehingga x_{13} diberikan nilai 4. Pemberian nilai $x_{13} = 4$ masih menghasilkan keluaran *bounding function* True sehingga pembangkitan nilai dilanjutkan untuk x_{14}. Untuk nilai 1 yang diberikan pada x_{14} diperoleh keluaran *bounding function* True sehingga pembangkitan nilai dilanjutkan untuk x_{15}. Nilai 1 tidak bisa diberikan untuk x_{15} karena akan menyebabkan ada 2 angka 1 pada kolom ketiga. Nilai 2 bisa diberikan kepada x_{15} karena tidak melanggar *constraint* sehingga pembangkitan nilai akhirnya dilanjutkan ke sel terakhir yaitu untuk x_{16}, mulai dari 1. Nilai 1 dan 2 tidak bisa diberikan kepada x_{16} karena mengakibatkan banyak angka 1 dan 2 di baris keempat lebih dari 1 sehingga dibangkitkan nilai yang lain yaitu 3. Untuk nilai 3 diperoleh keluaran bounding function True sehingga memenuhi. Mengingat sekarang vektor solusi sudah lengkap maka pencarian solusi permasalahan 1 permainan *Mainarizumu* selesai. Hasil yang diperoleh dan pohon ruang status yang dibentuk dalam pencarian solusi permasalahan 1 ini dan adalah sebagai berikut.

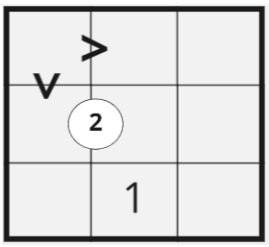




Gambar 11 Pohon Ruang Status dan Hasil Penyelesaian Persoalan 1 Permainan *Mainarizumu*

Sumber: Dokumentasi Pribadi

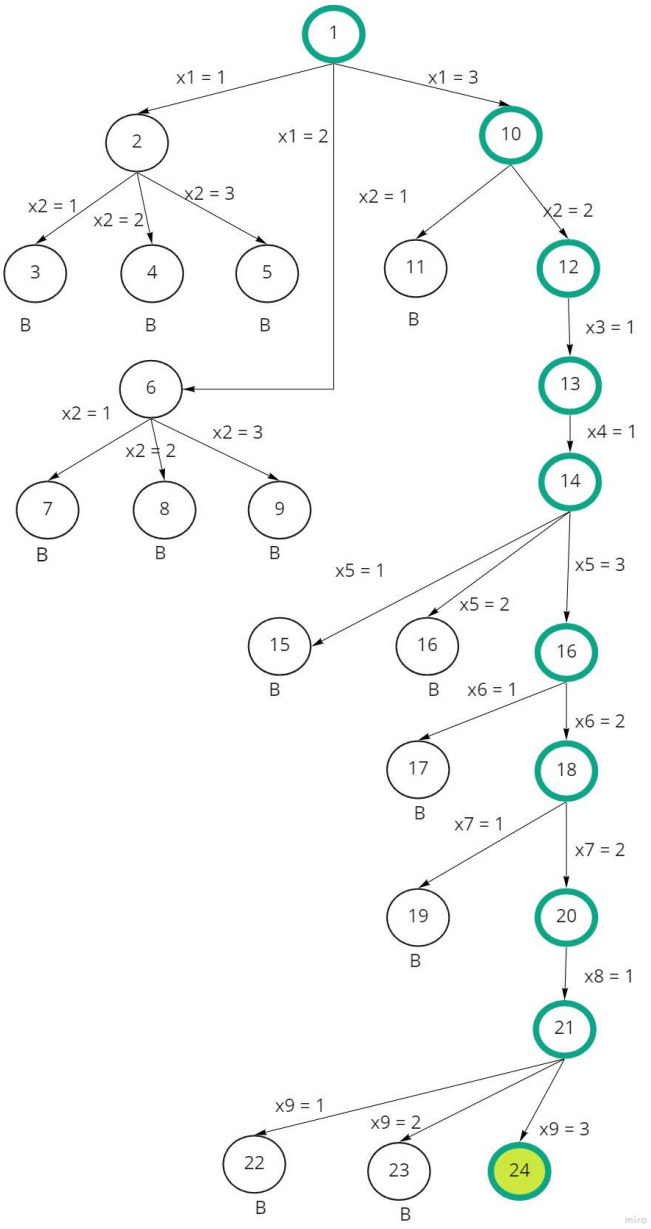
- Persoalan 2



Gambar 3 Contoh Persoalan 2 Permainan *Mainarizumu*

Sumber: Dokumentasi Pribadi

Pada pencarian solusi persoalan 2 tersebut pertama-tama dilakukan pencarian nilai untuk x_1, yaitu nilai pada sel di baris 1 dan kolom 1. Untuk setiap nilai yang dibangkitkan yaitu 1, 2, dan 3 secara dilakukan pengecekan apakah memenuhi fungsi pembatas atau tidak, dimulai dari nilai 1. Karena nilai 1 memenuhi maka dibangkitkan nilai untuk x_2. Ternyata untuk setiap nilai 1, 2, 3 tidak ada nilai yang memenuhi fungsi pembatas (nilai 1 tidak memenuhi karena akan menyebabkan dua buah nilai 1 pada baris pertama, nilai 2 dan 3 tidak memenuhi karena tidak memenuhi syarat $x_1 > x_2$) sehingga dilakukan *backtrack* untuk mencari kemungkinan nilai x_1 yang lain. Karena sebelumnya nilai x_1 adalah 1 maka saat *backtrack* dilakukan pengecekan untuk nilai 2. Ternyata untuk setiap nilai 1, 2, 3 juga tidak ada nilai yang memenuhi fungsi pembatas (nilai 1 tidak memenuhi karena akan menyebabkan dua buah nilai 1 pada kolom kedua, nilai 2 tidak memenuhi karena akan menyebabkan dua buah nilai 2 pada baris pertama, dan 3 tidak memenuhi karena tidak memenuhi syarat $x_1 > x_2$) sehingga dilakukan *backtrack* untuk mencari kemungkinan nilai x_1 yang lain. Karena sebelumnya nilai x_1 adalah 2 maka saat *backtrack* dilakukan pengecekan untuk nilai 3.

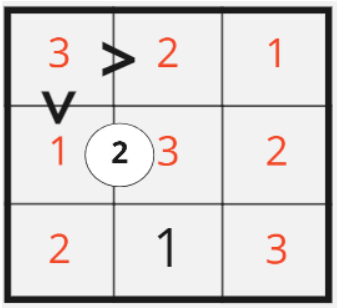


Gambar 12 Penyelesaian Persoalan 2 Permainan Mainarizumu

Sumber: Dokumentasi Pribadi

Selanjutnya dicari x_2 yang memenuhi fungsi pembatas. Nilai 1 tidak bisa menjadi nilai untuk x_2 karena akan menyebabkan terdapat dua buah nilai 1 pada kolom kedua. Jika nilai $x_2 = 2$ saat pemeriksaan *bounding function* akan dihasilkan nilai benar (artinya tidak melanggar aturan permainan) sehingga pencarian nilai dilakukan untuk x_3. Pencarian solusi pada pohon ruang status dilakukan seperti ilustrasi sebelumnya secara terus-menerus hingga ditemukan vektor n-*tuple* $X = (x_1, x_2, x_3, \ldots, x_n)$ yang memenuhi. Vektor inilah yang menjadi penyelesaian permainan sehingga solusi

permasalahan 2 permainan *Mainarizumu* adalah sebagai berikut.



C. *Pengujian Algoritma Backtracking dalam Menyelesaikan Permainan Mainarimuzu dengan Program*

Penyelesaian permainan *Mainarimuzu* dengan memanfaatkan algoritma runut-balik dapat ditranslasikan ke dalam kode program.

Dengan kode program yang telah dibuat dapat dilakukan pengujian terhadap pohon ruang status yang telah dibentuk terhadap persoalan I dan persoalan II.

- Pengujian persoalan 1
  Masukan dari file txt:

```
4
0-0-430
>|<|-|<
0-0-0>0
-|>|-|-
0-0-0-0
1|-|-|-
030<0-0
```

Hasil program:

```
PS C:\Z\mainarizumu> python -u "c:\Z\mainarizumu\baca.py"
Hasil perhitungan (hanya matriks angka, diperoleh dari vektor N-tuple X):
2 3 4 1
1 4 3 2
3 2 1 4
4 1 2 3
------------------------------------------------------------------------
Hasil perhitungan (beserta tanda ketidaksamaan dan angka selisih):
2 - 3 - 4 3 1
> | < | - | <
1 - 4 - 3 > 2
- | > | - | -
3 - 2 - 1 - 4
1 | - | - | -
4 3 1 < 2 - 3
```

Dari pengujian yang dilakukan ditemukan hasil yang sama dengan proses *backtracking*.

- Pengujian persoalan 2
  Masukan dari file txt:

```
3
0>0-0
>|-|-
020-0
-|-|-
0-1-0
```

Hasil program:

```
PS C:\Z\mainarizumu> python -u "c:\Z\mainarizumu\baca.py"
Hasil perhitungan (hanya matriks angka, diperoleh dari vektor N-tuple X):
3 2 1
1 3 2
2 1 3
------------------------------------------------------------------------
Hasil perhitungan (beserta tanda ketidaksamaan dan angka selisih):
3 > 2 - 1
> | - | -
1 2 3 - 2
- | - | -
2 - 1 - 3
```

Dari pengujian yang dilakukan ditemukan hasil yang sama dengan proses *backtracking*.

IV. KESIMPULAN

Algoritma runut-balik (*backtracking*) adalah algoritma yang merupakan perbaikan dari *exhaustive search*. Pada algoritma ini terdapat tiga properti umum yaitu solusi umum, fungsi pembangkit, dan fungsi pembatas. Semua kemungkinan solusi persoalan (ruang solusi) diorganisasikan ke dalam pohon ruang status dengan aturan pembangkitan simpul seperti pada *Depth Fisrt Search* (DFS). Jika lintasan yang sedang dibentuk tidak mengarah ke solusi simpul ekspan akan dimatikan dan proses pencarian *backtrack* ke simpul di atasnya. Algoritma *backtracking* umumnya digunakan untuk memecahkan *constraint satisfaction problem*, misalnya pada *logic puzzle* seperti permainan *Mainarizumu*.

Mainarizumu dimainkan pada papan persegi yang berukuran 7 x 7 atau lebih kecil. Tujuan permainan ini adalah mengisi setiap sel dengan angka 1 sampai N dengan syarat tidak ada angka yang sama pada baris atau kolom yang sama serta semua susunan angka tersebut mematuhi tanda ketidaksamaan dan selisih antarsel.

Dalam menyelesaikan permainan *mainarizumu* dengan menggunakan algoritma *backtracking* terlebih dahulu perlu ditentukan bentuk solusi umum, fungsi pembangkit, dan fungsi pembatasnya. Solusi umum permainan *mainarizumu* adalah vektor n-*tuple* $X = (x_1, x_2, x_3, \ldots, x_n)$ dengan n = banyak sel (n = N x N) dan $x_i \in \{1, 2, \ldots, N\}$. Nilai yang dibangkitkan untuk setiap komponen vektor X adalah bilangan bulat dari 1 hingga N. Fungsi pembatas dalam penyelesaian fungsi ini harus memastikan bahwa tidak ada angka yang sama pada baris atau kolom yang sama dan semua sel mematuhi tanda ketidaksamaan dan selisih antarsel.

VIDEO LINK AT YOUTUBE

Penulis juga mempersiapkan sebuah video terkait penerapan algoritma *backtracking* dalam permainan *mainarizumu* yang dapat dilihat pada link berikut.

https://www.youtube.com/watch?v=VI9SRXR5KAw

Untuk melihat kode program untuk pengecekan bisa dilihat melalui link berikut.

https://github.com/ruhiyahfw/Mainarizumu-Solution

UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan puji dan syukur kepada Tuhan Yang Maha Esa karena atas rahmat dan berkah-Nya penulis dapat menyelesaikan makalah ini dengan baik.

Selanjutnya penulis mngucapkan terima kasih kepada orang tua penulis yang sudah memberikan dukungan terbaik kepada penulis dalam menuntut ilmu.

Kemudian penulis mengucapkan terima kasih kepada Bapak dan Ibu dosen pengajar IF2210 Strategi Algoritma yang telah membimbing dan memberikan ilmu kepada penulis, khususnya ilmu mengenai strategi algoritma yang digunakan dalam pembuatan makalah ini.

DAFTAR PUSTAKA

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf, diakses tanggal 6 Mei 2021 pukul 13.00

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian2.pdf, diakses tanggal 6 Mei 2021 pukul 13.00

[3] https://www.geeksforgeeks.org/backtracking-introduction/, diakses tanggal 8 Mei 2021 pukul 20:00

[4] https://www.programiz.com/dsa/backtracking-algorithm, diakses 8 Mei 2020 pukul 20:00

[5] https://www.janko.at/Raetsel/Mainarizumu/index.htm, diakses tanggal 9 Mei 2021 pukul 21.00

[6] https://www.owlapps.net/owlapps_apps/articles?id=52079043&lang=en, diakses tanggal 10 Mei 2021 pukul 6.30

[7] https://clontz.org/puzzles/types/, diakses 11 Mei 2021 pukul 17.30

[8] 8 Surprising Benefits of Playing Puzzle Games Daily | Ascend Healthy, diakses 11 Mei 2021 pukul 17.30

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, 26 April 2021

Ruhiyah Faradishi Widiaputri
13519034