Penerapan Algoritma Runut-balik pada Permainan *Math Maze*

Angela Lynn - 13513032
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
angelynz95@students.itb.ac.id

Abstrak*—Makalah ini berisi sedikit penjelasan tentang teori algoritma runut-balik, permainan Math Maze, dan penerapan algoritma runut-balik untuk memperoleh solusi permainan Math Maze. Algoritma adalah urutan langkah-langkah untuk memecahkan suatu persoalan dengan memproses masukan menjadi keluaran. Beberapa strategi algoritma, antara lain algoritma Brute-Force, algoritma Greedy, algoritma Divide and Conquer, algoritma Decrease and Conquer, algoritma Runut-balik, algoritma Branch and Bound, dan Dynamic Programming. Algoritma runut-balik merupakan perbaikan dari exhaustive search. Solusi permainan Math Maze bisa diperoleh dengan memanfaatkan algoritma runut-balik dan solusi tersebut pasti benar.

***Kata Kunci*—Algoritma Runut-balik, *Maze Math*, Solusi Persoalan, Fungsi Pembangkit, Fungsi Pembatas, Pohon Ruang Status.**

I. PENDAHULUAN

Permainan *Math Maze* adalah suatu permainan yang menerapkan konsep permainan labirin, yaitu mencari jalan keluar dari labirin yang ada di permainan tersebut. Permainan ini cukup unik karena tidak seperti permainan labirin lainnya. Pada *Math Maze*, pemain harus menjawab soal matematika yang diberikan supaya bisa mendapat jalur penyelesaian yang tepat.

Permainan seperti *Math Maze* tentunya sangat bagus untuk dimainkan, karena selain melatih logika, juga dapat melatih kemampuan matematika. Permainan ini bisa juga digunakan untuk melatih orang-orang mengerjakan soal matematika dengan cara yang lebih menyenangkan dan tidak membosankan. Oleh karena itu, penulis tertarik untuk menerapkan algoritma runut-balik pada permainan *Math Maze* ini dan akan menjelaskannya melalui makalah ini.

II. TEORI DASAR

A. *Persoalan*

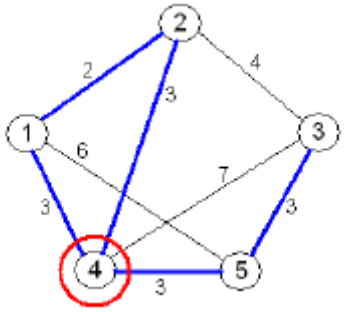
Persoalan adalah pertanyaan atau tugas yang kita cari jawabannya. Contoh-contoh persoalan seperti persoalan pengurutan dan persoalan pencarian. Instansiasi persoalan adalah parameter nilai yang diasosiasikan di dalam persoalan. Jawaban terhadap instansiasi persoalan disebut solusi.

Contoh: Selesaikan persoalan pengurutan untuk
S = [15, 4, 8, 11, 2, 10, 19] n = 7

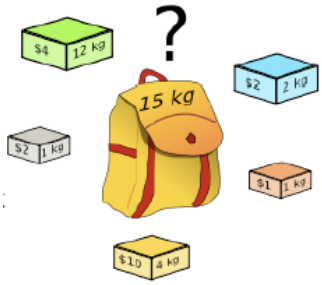
Solusi: S = [2, 4, 8, 10, 11, 15, 19].

Beberapa contoh persoalan klasik:

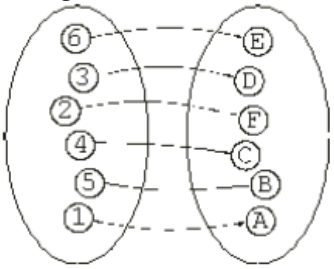
1. *Travelling Salesperson Problem (TSP).*



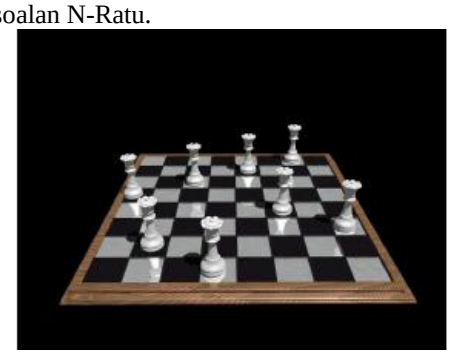
2. *Integer Knapsack Problem.*



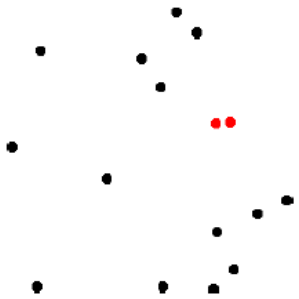
3. Persoalan Penugasan.



4. Persoalan N-Ratu.



5. Mencari Pasangan Titik Terdekat

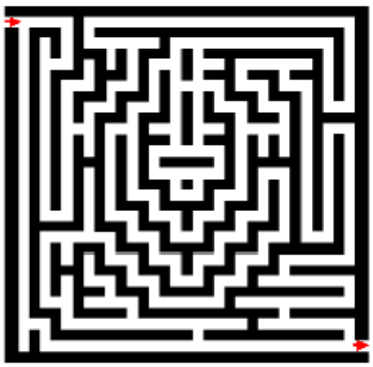


6. Permainan15-Puzzle.

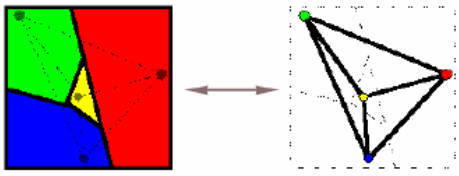
(a) Susunan awal

(b) Susunan akhir

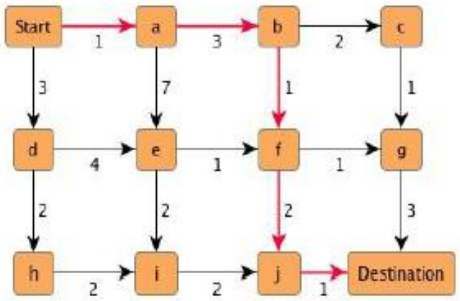
7. *Maze Problem.*



8. Pewarnaan Graf



9. Lintasan Terpendek



B. Algoritma

Algoritma adalah urutan langkah-langkah untuk memecahkan suatu persoalan, dengan memproses masukan menjadi keluaran. Untuk persoalan dengan instansiasi yang besar, solusinya menjadi lebih sulit ditentukan, sehingga biasanya menggunakan algoritma untuk membantu menyelesaikannya.

Dalam suatu algoritma, biasanya dilakukan analisis. Tujuannya adalah untuk mengukur kinerja algoritma dari segi kemangkusannya.

Alat ukur kemangkusan algoritma:

1. Kompleksitas waktu, T(n), yaitu jumlah tahap komputasi yang dilakukan untuk menjalankan sebuah algoritma sebagai fungsi dari ukuran masukan n.
2. Kompleksitas ruang, S(n), yaitu ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n.

n adalah ukuran masukan yang diproses oleh algoritma.

C. Strategi Algoritma

Strategi algoritma yang ada, antara lain:

1. Algoritma *Brute-Force.*
2. Algoritma *Greedy.*
3. Algoritma *Divide and Conquer.*
4. Algoritma *Decrease and Conquer.*
5. Algoritma Runut-balik.
6. Algoritma *Branch and Bound.*
7. *Dynamic Programming.*

Menurut Levitin, ada dua alasan mengapa perlu mempelajari strategi algoritma, yaitu:

1. Memberikan panduan untuk merancang algoritma bagi persoalan baru.
2. Dapat mengklasifikasikan algoritma berdasarkan gagasan perancangan yang mendasarinya.

Klasifikasi strategi algoritma:

1. Strategi solusi langsung, contohnya algoritma *Brute Force* dan algoritma *Greedy.*
2. Strategi berbasis pencarian pada ruang status, contohnya algoritma runut-balik dan algoritma *Branch and Bound.*
3. Strategi solusi atas-bawah, contohnya algoritma *Divide and Conquer* dan algoritma *Decrease and Conquer.*
4. Strategi solusi bawah-atas, contohnya *Dynamic Programming.*

Pada makalah ini, hanya algoritma runut-balik yang akan dijelaskan lebih rinci.

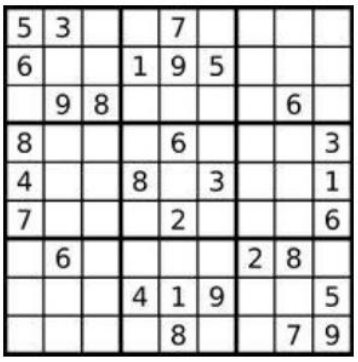
D. Algoritma Runut-balik

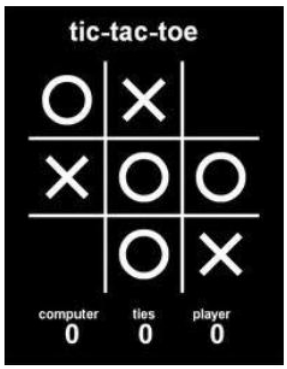
Algoritma runut-balik pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950. Kemudian R. J. Walker, Golomb, dan Baumert menyajikan uraian umum tentang algoritma runut-balik.

Runut-balik dapat dipandang sebagai salah satu dari dua hal berikut:

1. Sebagi sebuah fase di dalam algoritma traversal DFS.
2. Sebagai sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis.

Runut-balik banyak diterapkan untuk program *games*, seperti permainan *tic-tac-toe*, menemukan jalan keluar dalam sebuah labirin, catur, *crossword puzzle*, *sudoku*, dan masalah-masalah pada bidang kecerdasan buatan.





Ciri – ciri permasalahan yang tepat untuk diselesaikan dengan algoritma runut-balik:

1. Membuat rangkaian keputusan di antara beberapa pilihan.
2. Tidak mempunyai cukup informasi untuk mengetahui apa yang akan dipilih.
3. Tiap keputusan mengarah pada sekumpulan pilihan baru.
4. Beberapa sekuens pilihan mungkin merupakan solusi persoalan.

Dalam algoritma runut-balik, ada dua solusi untuk mengetahui langkah mana yang perlu dijejaki kembali, yaitu:

1. Simpan semua langkah yang pernah dilakukan.
2. Menggunakan rekursi (yang secara implisit menyimpan semua langkah).

Jika kita menggambarkan sekuens pilihan yang kita lakukan, maka diagram akan berbentuk seperti pohon. Simpul daun bisa merupakan titik runut-balik atau simpul *goal.* Pada titik runut-balik, simpul tersebut menjadi mati (tidak bisa diekspansi lagi). Dalam pembentukan simpulnya, runut-balik menerapkan DFS.

Algoritma runut-balik merupakan perbaikan dari *exhaustive search.* Pada *exhaustive search*, semua kemungkinan solusi dieksplorasi satu per satu, sedangkan pada runut-balik, hanya pilihan yang mengarah ke solusi yang akan dieksplorasi. Pilihan yang tidak mengarah ke solusi tidak dipertimbangkan lagi. Runut-balik akan memangkas simpul-simpul yang tidak mengarah ke solusi.

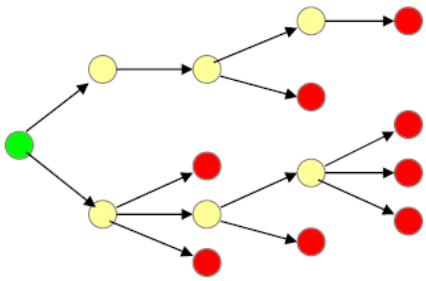
Properti umum metode runut-balik ada tiga, yaitu:

1. Solusi persoalan.
   Solusi dinyatakan sebagai vector dengan *n-tuple*: X = (x_1, x_2, …, x_n), dimana x_i merupakan elemen bagian S_i.
2. Fungsi pembangkit.
   Untuk membangkitkan nilai x_k. Dinyatakan sebagai predikat: T(k). T(k) membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.
3. Fungsi pembatas.
   Dinyatakan sebagai predikat B(x_1, x_2, …, x_k). B bernilai true jika (x_1, x_2, …, x_k) mengarah ke solusi. Jika true, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika false, maka (x_1, x_2, …, x_k) dibuang.

Semua kemungkinan solusi dari persoalan disebut ruang solusi. Dalam algoritma runut-balik, ruang solusi diorganisasikan ke dalam struktur pohon. Tiap simpul pohon menyatakan status persoalan, sedangkan cabang dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun akan membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status.

Runut-balik dapat dipandang sebagai pencarian di dalam pohon menuju simpul daun (*goal)* tertentu. Ada tiga macam simpul, yaitu:

1. Simpul akar (berwarna hijau).
2. Simpul dalam (berwarna krem).
3. Simpul daun (berwarna merah).



Prinsip pencarian solusi dengan metode runut-balik:

- Solusi dicari dengan membentuk lintasan dari akar ke daun.
- Aturan pembentukan yang dipakai adalah mengikuti aturan DFS.
- Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup.
- Simpul hidup yang sedang diperluas dinamakan simpul-E.
- Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang.
- Jika lintasan yag sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi simpul mati.
- Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas.
- Simpul yang sudah mati tidak akan pernah diperluas lagi.
- Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian runut-balik ke simpul aras di atasnya.
- Lalu, teruskan dengan membangkitkan simpul anak yang lain. Selanjutnya simpul ini menjadi simpul-E yang baru.
- Pencarian dihentikan bila telah mencapai simpul tujuan.

Skema Umum Algoritma Runut-Balik (versi rekursif)

```
procedure RunutBalikR(input k:integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema
rekursif
 Masukan: k, yaitu indeks komponen vektor solusi, x[k]
 Keluaran: solusi x = (x[1], x[2], …, x[n])
}
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[1], x[2], ... ,x[k]) adalah lintasan dari akar ke daun
      then
        CetakSolusi(x)
      endif
      RunutBalikR(k+1)     { tentukan nilai untuk x[k+1]}
   endfor
```

Jika jumlah simpul dalam pohon ruang status adalah $2n$ atau $n!$, maka untuk kasus terburuk, algoritma runut-balik membutuhkan waktu dalam $O(p(n)2n)$ atau $O(q(n)n!)$, dengan $p(n)$ dan $q(n)$ adalah polinom derajatn yang menyatakan waktu komputasi setiap simpul.

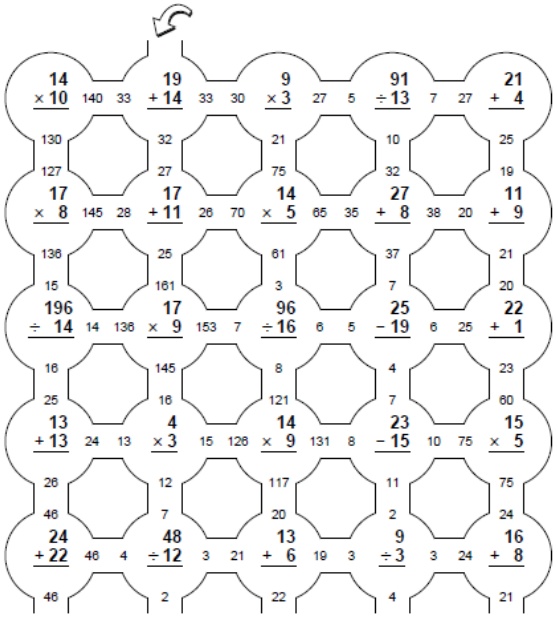
Penerapan lain oleh algoritma runut-balik:

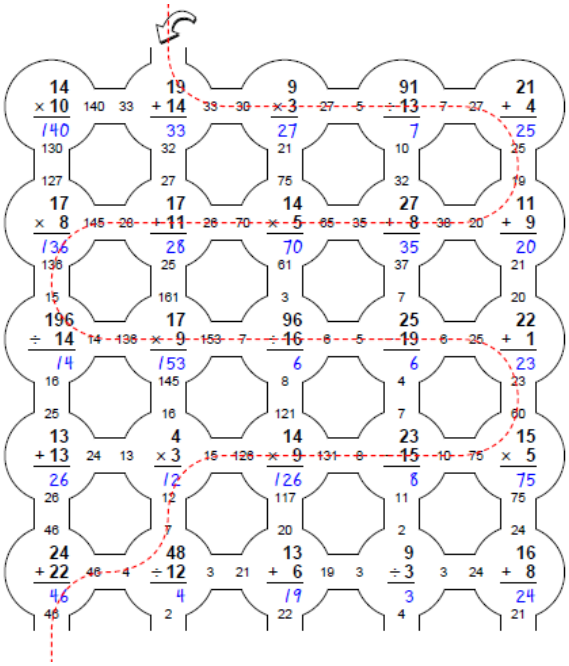
- Persoalan N-Ratu.
- Pewarnaan Graph.

III. *MATH MAZE*

Math Maze adalah salah satu permainan yang menerapkan konsep labirin. Permainan ini dibuat oleh perusahaan Amerika bernama MathMaze USA, dengan target anak-anak dari TK hingga kelas 3 SMA. Tujuan dibuatnya permainan ini adalah supaya meningkatkan ketertarikan anak-anak terhadap matematika dan meningkatkan kemampuan mereka di bidang tersebut.

Permainan ini memiliki ciri khas yang membedakan dengan permainan labirin lainnya. Dalam mencari jalur penyelesaian, pemain harus mengerjakan soal matematika yang disediakan supaya bisa mengetahui jalur yang tepat. Setiap jalur memiliki nilai yang bisa saja berupa jawaban dari soal matematika tersebut.





Math Maze berhasil diselesaikan (diperoleh solusinya).

Berikut garis besar algoritma runut-balik untuk permainan *Math Maze:*

```
while belum mencapai jalan keluar do
    kerjakan soal matematika
    if  jalur memiliki jawaban yang benar dan
jalur tersebut belum pernah dikunjungi then
      masuk ke jalur tersebut
    else
      lakukan runut-balik
    endif
endwhile
```

Berikut *pseudo-code* algoritma runut-balik untuk permainan *Math Maze* secara rekursif:

```
procedure solveMathMaze(input maze : Maze)
  { Mencari solusi permainan Math Maze }

  Deklarasi
    { Variabel }
    direction : integer  { atas = 1, bawah = 2,
kiri = 3, kanan = 4}
    answer : real

    { Fungsi }
    function solveMathQuestion() -> real
    { Mengembalikan jawaban dari soal matematika
}
    procedure move(input direction : integer)
    { Bergerak sesuai dengan direction }
    procedure printSolution()
    { Menampilkan solusi Math Maze }

  Algoritma
    jawaban <- solveMathQuestion()
    if sudah mencapai solusi then
      printSolution()
    else
      for tiap direction do
        if direction memiliki value jawaban dan
belum pernah dikunjungi then
          move(direction)
          solveMathMaze(maze)
        endif
      endfor
  endif
```

IV. Algoritma Runut-balik *Math Maze*

Algoritma runut-balik untuk menyelesaikan permainan *Math Maze* adalah sebagai berikut:

1. Selesaikan soal matematika yang disediakan.
2. Ikutilah jalur yang memiliki jawaban yang sesuai dengan soal matematika di percabangan tersebut dan belum pernah dikunjungi.
3. Apabila ada lebih dari satu jalur yang memiliki jawaban benar dan belum pernah dikunjungi, pilihlah salah satu jalur.
4. Jika mencapai jalan buntu atau jalur yang memiliki jawaban benar sudah dikunjungi, lakukan runut-balik. Kemudian kembali ke langkah 2.
5. Jika sudah mencapai jalan keluar, maka permainan

V. Kesimpulan

Berdasarkan ilmu dan pengetahuan yang penulis dapatkan dari referensi-referensi yang penulis gunakan untuk menyelesaikan makalah ini, maka penulis bisa menarik kesimpulan bahwa:

1. Algoritma runut-balik bisa diterapkan dalam permainan *Math Maze.*
2. Algoritma runut-balik bisa memberikan solusi yang tepat untuk permainan *Math Maze.*

VI. Ungkapan Terima Kasih

Penulis memanjatkan syukur kepada Tuhan Yang Maha

Esa. Atas berkat dan rahmat-Nya, penulis dapat menyelesaikan makalah ini dengan baik. Penulis mengucapkan terima kasih kepada dosen IF2211 Strategi Algoritma, yaitu Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi atas ilmu dan didikan yang diberikan oleh beliau, sehingga penulis bisa mendapatkan inspirasi selama pembuatan makalah. Penulis juga mengucapkan terima kasih kepada semua pihak yang sudah membantu penulis memberikan ide dalam pengerjaan makalah ini, baik secara langsung maupun tidak langsung.

REFERENSI

[1] Munir, Rinaldi, "Algoritma Runut-balik (2015)".
[2] http://www.mathmaze.us, diakses pada tanggal 4 Mei 2015 pukul 17.00.
[3] http://www.worksheetworks.com/puzzles/math-maze.html, diakses pada tanggal 4 Mei 2015 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, 4 Mei 2015

Angela Lynn - 13513032