# Penerapan Konsep BackTracking untuk Mencari Solusi pada Permainan 24

Syihabuddin Yahya Muhammad 13519149
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519149@std.stei.itb.ac.id

***Abstract*—Permainan 24 adalah permainan sederhana yang dapat mengasah kemampuan berpikir dan menghitung. Namun melakukan permainan tanpa bantuan mesin dapat menjadi masalah karena jawaban dapat tidak ditemukan ataupun memang tidak ada. Melalui algoritma runut-balik dapat dibuat suatu solusi yang bekerja dengan lebih efisien.**

***Keywords—Runut-balik, permainan 24, pohon***

## I. PENDAHULUAN

Algoritma *Backtracking* atau runut-balik adalah sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis. Algoritma ini dapat dikatakan sebagai metode perbaikan dari *exhaustive search* dengan menggunakan *pruning*. Pada algoritma *exhaustive search*, setiap kemungkinan masalah akan dieksplorasi dan dievaluasi satu persatu. Sedangkan pada algoritma runut-balik, hanya pilihan yang mengarah kepada solusi yang dianalisis lebih lauh. Algoritma runut-balik ini dapat digunakan dalam hampir semua masalah yang kemungkinan solusinya terbatas.

Salah satu permasalahan yang dapat diselesaikan oleh algoritma runut-balik ini adalah pencarian solusi dari permainan 24. Permainan 24 merupakan permainan matematika sederhana dimana pemain disediakan n buah angka. n buah angka tersebut dapat diberikan operasi tambah, kurang, kali dan bagi. Tujuan akhir dari permainan ini adalah mendapatkan angka 24 dengan menggunakan semua angka yang disediakan. Umumnya dalam permainan ini, tidak diperbolehkan adanya pecahan dan angka negatif. Namun terkadang ada juga variasi permainan yang membolehkan pecahan dan bilangan negatif. Permainan ini menjadi agak bermasalah karena terdapat beberapa kombinasi angka yang tidak dapat menjadi 24 hanya dengan operasi tambah, kurang, kali, dan bagi.

## II. LANDASAN TEORI

### A. *Depth First Search*

*Depth First Search* atau sering disebut DFS merupakan algoritma pencarian tanpa informasi yang dilakukan pada sebuah graf. Dalam pemakaian algoritma ini, struktur data yang akan dicari harus diubah menjadi graf terlebih dahulu.

Pada DFS, struktur graf ditelusuri berdasarkan kedalamannya. Penelusuran dimulai dari akar(*root*) kemudian dilanjutkan kepada anaknya. Kemudian penelusuran dilanjutkan kepada anaknya lagi. Hal ini diulang terus menerus hingga simpul yang ditelusurinya sudah tidak memiliki anak atau semua anak simpul tersebut telah ditelusuri. Jika hal ini terjadi, maka penelusuran akan Kembali kepada simpul sebelumnya dan akan ditelusuri anaknya yang lain yang belum pernah ditelusuri.

Dalam implementasinya, pencarian DFS dapat dilakukan dengan bantuan fungsi rekursif dan struktur data stack. Sederhananya algoritma pencarian DFS akan berbentuk seperti ini.

1. Masukkan simpul akar kepada stack.
2. Mulai fungsi rekursif.

2.1. Ambil simpul paling atas yang terdapat pada stack.

2.2. Cek apakah simpul tersebut merupakan solusi.

2.2.1. Jika ya, maka hentikan pencarian dan kembalikan simpul tersebut.

2.3. Cek apakah simpul tersebut memiliki simpul anak.

2.3.1. Jika ya, maka push semua anak simpul tersebut kepada stack

2.4. Cek apakah stack sudah kosong

2.4.1. Jika ya, maka hentikan pencarian, dan kembalikan pesan bahwa nilai yang dicari tidak ditemukan.

2.4.2. Jika tidak, maka ulangi Kembali fungsi rekursif ini.

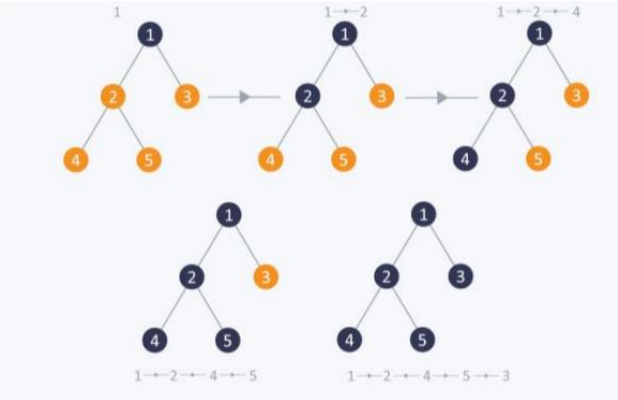
Representasi *pseudocode* dari algoritma pencarian DFS dapat dilihat pada gambar 2.1, sedangkan representasi visual dari pencarian DFS dapat dilihat pada gambar 2.2.

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}
Deklarasi
   w : integer

Algoritma:
   write(v)
   dikunjungi[v]←true
   for w←1 to n do
      if A[v,w]=1 then {simpul v dan simpul w bertetangga }
         if not dikunjungi[w] then
            DFS(w)
         endif
      endif
   endfor
```

Gambar 2.1 *Pseudocode* DFS [4]



Gambar 2.2 Visualisasi pencarian DFS [1]

## B. *BackTracking*

*Backtracking* atau runut-balik adalah sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis. Runut-balik merupakan algoritma yang dikembangkan dari algoritma *Depth First Search*(DFS) dengan mengaplikasikan *pruning* atau pemangkasan., yaitu menghapus kandidat solusi yang dianggap tidak akan mengarah kepada solusi. Oleh karena itu, algoritma ini juga sering disebut sebagai perbaikan dari *exhaustive search* yang menelusuri semua kandidat solusi tanpa pertimbangan lebih jauh.

1. Properti Umum Algoritma Runut-balik

   a. Solusi Persoalan

      Solusi dari sebuah pencarian runut-balik dinyatakan sebagai n-tuple dimana setiap elemennya merupakan anggota dari semestanya masing-masing. Notasi Matematika dari solusi ini adalah:

      $X = (x_1, x_2, \ldots, x_n), x_i \in S_i$

      Umumnya semesta dari setiap elemen sama, yang dalam notasi matematikanya adalah:

      $S_1 = S_2 = S_3 = \ldots = S_n$

      Berikut contoh dari solusi persoalan pada *Knapsack's Problem* dengan n benda:

      $X = (x_1, x_2, \ldots, x_n), S_i = \{0, 1\}$

      Yang artinya semua elemen dari tuple solusi akan bernilai 0 atau 1.

   b. Fungsi Pembangkit

      Fungsi pembangkit pada algoritma runut-balik bertugas untuk memunculkan langkah selanjutnya. Dalam notasi Matematika, fungsi pembangkit $T(x_1, x_2, \ldots, x_{k-1})$ akan membangkitkan nilai $x_k$ yang akan menjadi komponen vektor solusi juga.

   c. Fungsi Pembatas

      Fungsi Pembatas pada algoritma runut-balik bertugas untuk menentukan apakah suatu simpul mengarah pada solusi. Umumnya fungsi ini akan mengembalikan nilai suatu Boolean *true* atau *false*. Jika fungsi pembatas pada suatu simpul mengebalikan nilai *false*, maka simpul tersebut akan dibuang, sedangkan jika nilainya *true*, akan dibangkitkan Langkah selanjutnya dengan menggunakan fungsi pembangkit.

2. Pengorganisasian Solusi

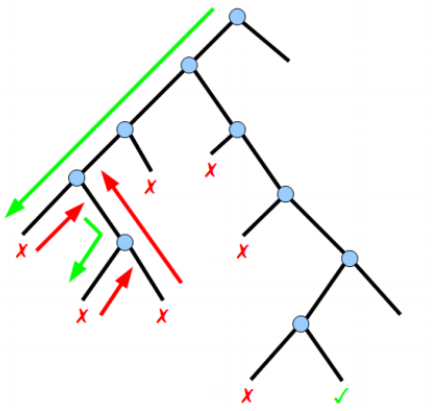
   Semua kemungkinan solusi dari persoalan disebut dengan ruang solusi. Dalam penyajiannya, sebuah ruang solusi dapat disajikan dalam struktur data *tree* atau pohon. Setiap simpul pada pohon menyatakan status dari persoalan, sedangkan setiap sisi melambangkan/dilabeli dengan nilai $x_i$.

   Lintasan dari akan ke daun menyatakan solusi yang mungkin, sedangkan seluruh lintasan dari akan ke daun akan membentuk ruang solusi.

   Pohon yang terbentuk sering kali disebut sebagai pohon ruang status.

3. Prinsip Pencarian Solusi

   Pada algoritma runut-balik, pencarian solusi dilakukan dengan membangkitkan simpul-simpul status, sehingga terbentuk suatu lintasan dari akar ke daun. Pembangkitan simpul dilakukan dengan mengikuti aturan DFS. Simpul-simpul yang telah dibangkitkan dinamakan dengan simpul hidup. Simpul yang sedang diperluas dinamakan dengan simpul *expand*. Suatu simpul akan di *expand* sesuai dengan urutan pada algoritma DFS. Setiap kali suatu simpul diperluas, lintasan yang dibangunnya akan bertambah panjang. Jika suatu simpul tidak mengarah kepada solusi, simpul tersebut akan dimatikan menjadi simpul mati. Sebuah simpul mati tidak akan di *expand*, sehingga pencarian menjadi lebih efisien.



Gambar 2.3 Contoh pencarian menggunakan algoritma runut-balik [2]

## C. *Permainan 24*

Permainan 24 adalah permainan matematika sederhana yang dapat membantu mengasah kecepatan berpikir dan menghitung seseorang. Pada permainan ini pemain akan diberikan n buah

angka positif yang tidak harus berbeda. Umumnya jumlah angka yang diberikan ada sebanyak empat, namun jumlah angka ini dapat berubah sesuai dengan kesepakatan pemain.

Permainan biasanya dijalankan menggunakan sistem ronde. Pada setiap ronde, pemain harus mencari sebuah solusi dengan menerapkan operasi tambah, kurang, kali dan bagi untuk mendapatkan nilai 24 dengan menggunakan semua angka yang disediakan. Setiap angka yang disediakan harus digunakan tepat satu kali. Pemain yang mendapatkan solusi paling cepat akan memenangkan ronde tersebut. Alternatif lainnya, pemain yang paling lambat menemukan solusi kalah pada ronde tersebut.

Pada alternatif pertama, pemain yang berhasil mendapatkan solusi paling cepat akan mendapatkan poin. Pemenang dari pertandingan adalah pemain yang memiliki poin paling besar. Pada alternatif kedua, pemain yang mendapatkan solusi paling terakhir akan mendapatkan poin, akan tetapi pemenang pertandingan adalah pemain yang poinnya paling sedikit. Pada alternatif ini, permainan juga dapat dibuat menjadi lebih menarik dengan membolehkan pemain untuk berbohong. Seorang pemain dapat menyatakan dirinya telah menemukan solusi padahal sebenarnya belum. Pada akhir ronde, pemain yang kalah dapat menanyakan solusinya pada seorang pemain lain. Jika pemain tersebut tidak dapat menjawab, maka pemain yang kalah ganti menjadi pemain tersebut.

Jika permainan dilakukan secara offline, umumnya angka yang diberikan akan dipilih secara acak dari objek yang memiliki angka seperti kartu remi ataupun kartu/batu mahjong. Dikarenakan sistemnya yang sangat acak, jika angka dipilih dari kartu remi, kartu As biasanya dapat bernilai 1 atau 11, sehingga solusi yang ada menjadi lebih banyak dan permainan menjadi lebih mudah. Hal ini juga dilakukan untuk mengurangi kombinasi kartu yang tidak memiliki solusi. Kartu dengan angka biasanya bernilai angka itu sendiri sedangkan kartu royal(Jack, Queen, King) bernilai 11.

Gambar 2.4 permainan 24 menggunakan kartu remi [3]

Selain dilakukan secara offline, permainan 24 dapat juga dilaksanakan secara online/digital. Hingga saat ini sudah tersedia banyak aplikasi yang menyajikan permainan 24 dengan berbagai variasi. Beberapa dari variasi yang tersedia adalah

1. Multi player

   Pada variasi *multiplayer*, permainan 24 akan terasa sangat mirip dengan permainan yang dilakukan secara offline. Pemain akan bertanding dengan pemain lain menggunakan aturan yang sama dengan permainan offline.

2. Mini Game

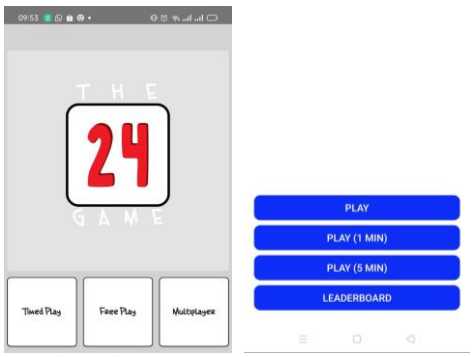
   pada variasi *Mini Game*, jalannya permainan sangat mirip dengan permainan *multiplayer*. Hanya saja pada variasi ini, disediakan juga permainan lain sehingga permainan 24 hanyalah sebagian kecil dari permainan yang disediakan. dengan strategi ini, pertandingan dapat berjalan dengan lebih menarik.

3. Arcade

   Pada variasi *Arcade*, permainan disajikan dalam bentuk berbagai level. Pemain dapat menyelesaikan kombinasi angka yang disediakan pada suatu level untuk membuka level selanjutnya. Biasanya dalam variasi ini disediakan juga opsi petunjuk untuk membantu pemain yang kesulitan. Biasanya terdapat juga sistem skor, dimana skor pemain akan lebih tinggi jika pemain tidak melakukan kesalahan perhitungan ataupun memakai petunjuk.

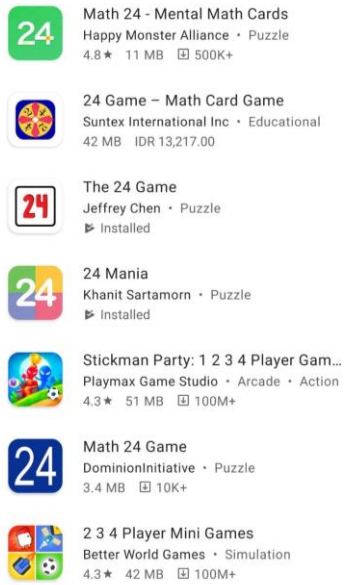
4. Highscore

   Pada variasi *highscore*, permainan disajikan secara terus. Permainan dapat berupa menyelesaikan kombinasi angka terbanyak dalam batas waktu yang ditentukan, menyelesaikan kombinasi angka sejumlah yang ditentukan dalam waktu secepat-cepatnya., ataupun tantangan-tantangan lainnya.




Gambar 2.5 permainan 24 dalam mode online/digital

Selain 4 variasi tersebut, masih banyak juga variasi lain yang tersebar di internet. Permainan tersebut dapat diakses melalui internet seperti google, Appstore, ataupun iostore.




Gambar 2.6 hasil pencarian permainan 24 di playstore

## III. IMPLEMENTASI

Pada Implementasi kali ini, penulis menggunakan Bahasa Java dengan pendekatan paradigma berbasis objek. Program dibuat dengan membuat objek *Node* yang melambangkan setiap simpul pada pohon ruang status. Setiap simpul memiliki fungsi pembangkit dan pembatasnya sendiri sehingga program dapat berjalan dengan lebih efektif.

### A. *Banyak kemungkinan solusi*

Misal pada permainan 24 terdapat n angka yang disediakan. Pada setiap Langkah, akan dipilih 2 angka berbeda yang tersedia, lalu kemudian akan dipilih operasi antar kedua angka tersebut. Setelah operasi dijalankan kedua angka yang telah dipilih akan hilang karena sudah terpakai, akan tetapi jumlah angka juga akan bertambah satu, karena hasil operasi kedua angka tersebut akan menjadi sebuah angka yang baru. Dengan demikian setelah suatu langkah dijalankan, jumlah angka yang tersedia akan berkurang satu. Langkah tersebut akan terus dilakukan hingga hanya tersisa satu angka saja. Jika angka yang tersisa adalah 24, maka Langkah-langkah yang diambil merupakan solusi. Jika tidak, maka Langkah tersebut bukanlah solusi.

Pada setiap langkah, jika terdapat n angka yang tersedia, total pilihan yang dapat dilakukan ada sebanyak nP2 * 4. nP2 didapat dari cara memilih 2 angka yang akan dioperasikan. Digunakan Permutasi karena urutan angka pada yang akan dioperasikan berpengaruh terhadap hasil akhirnya(contohnya a-b belum tentu sama dengan b-a). angka 4 didapat dari 4 pilihan operasi yang bisa dipilih yaitu tambah, kurang, kali dan bagi.

Total kemungkinan solusi adalah kombinasi dari semua langkah yang mungkin. Jika terdapat n angka yang tersedia, maka solusi akan terdiri dari n-1 langkah. Setiap langkah akan diambil dari nilai n yang berbeda yaitu mulai dari n hingga 2. Dengan demikian total kemungkinan solusi dari permainan 24 ini ada sebanyak

$$\prod_{k=2}^{n} 4P_2^n = 4^{n-1} \times n! \times (n-1)!$$

### B. *Penggunaan Algoritma Runut-balik*

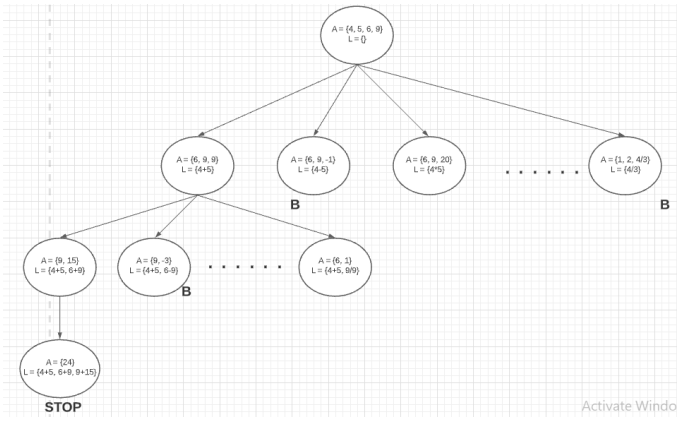
Pada permainan 24 terdapat beberapa aturan yang umumnya digunakan. Aturan tersebut adalah setiap operasi tidak boleh menghasilkan bilangan negatif ataupun pecahan. Dalam implementasi kali ini penulis akan mengasumsikan permainan 24 yang dilakukan adalah permainan 24 standar sehingga kedua peraturan ini akan diterapkan.

Dalam penerapan algoritma runut-balik langkah pertama yang dapat dilakukan adalah menentukan properti umumnya. Algoritma runut-balik memiliki 3 properti umum yaitu, himpunan solusi, fungsi pembangkit, dan fungsi pembatas.

Hal pertama yang perlu diperhatikan adalah, dalam permainan 24 ini terdapat 2 unsur yang sangat penting, yaitu nilai yang tersedia dan langkah yang diambil. Kedua nilai tersebut sangatlah penting dan harus disimpan pada setiap simpul. Misal langkah yang telah diambil disimpan dalam bentuk L = {$x_1$, $x_2$, …, $x_n$} dan nilai angka yang tersedia disimpan dalam bentuk A = {$y_1$, $y_2$, …, $y_n$} dimana $y_i \in R$ dan $x_i \in S_i$.

Karena solusi dari permainan ini adalah langkah-langkah yang diterapkan, maka untuk permainan dengan jumlah angka sebanyak n, solusi akan berbentuk X = {$x_1$, $x_2$, …, $x_{n-1}$}, dimana $S_i$ = {$y_a$op$y_b$} dengan a != b, $y_a$, $y_b \in$ A, dan op ∈ {+, -, *, /}. Untuk fungsi pembangkit, T(A, L) akan membangkitkan semua x dimana $x \in S_i$. Untuk fungsi pembatas, B(L, A) akan bernilai *true* jika dan hanya jika semua elemen A merupakan bilangan cacah.

Pada kasus yang akan diteliti, pemilihan $y_a$ dan $y_b$ akan diprioritaskan berdasarkan urutan pada A, sedangkan operasi yang dipilih akan mengikuti prioritas tambah, kurang, kali, dan bagi.




Gambar 3.1 Contoh Pohon Ruang Status Permainan 24

Pada algoritma yang penulis buat menggunakan Bahasa Java, implementasi algoritma Runut-balik akan digunakan dengan bantuan fungsi rekursif, dan paradigma berbasis objek. Pada program utama akan dibuat suatu objek node yang akan berisi angka-angka awal yang tersedia. Kemudian Ketika node tersebut terbuat, algoritma runut balik akan dijalankan dengan menggunakan fungsi rekursif untuk membangkitkan anak dari

setiap simpul. Pembangkitan simpul baru akan mengikuti prioritas yang telah dijelaskan sebelumnya. Jika angka awal yang disediakan adalah 4, 5, 6, dan 9 maka objek yang terbentuk akan memiliki struktur seperti yang tergambar pada gambar 3.1. simpul akan di-*expand* mulai dari paling kanan. Ketika simpul yang terbentuk sudah membentuk sebuah solusi(A = {24}), maka pengisian graf akan langsung terhenti.

## IV. Uji Coba

Pada pengetesan program yang penulis buat, akan digunakan 5 kasus yang terdiri dari 1 kasus 3 angka, 3 kasus 4 angka, dan 1 kasus 5 angka. 3 kasus 4 angka dibagi menjadi 2 kasus yang memiliki solusi dan 1 kasus yang tidak memiliki solusi.

### A. *Kasus 1*

Pada kasus ini angka awal dari permainan berjumlah 3 yaitu {2, 3, 4}. Seperti yang terlihat pada gambar 4.1 dan 4.2, solusi dari permainan sudah benar, dan dari 3! * 2! * $4^2$ = 192 kemungkinan solusi, program hanya perlu mengecek 10 solusi untuk menemukan solusi yang tepat.

```
angka: 2, 3, 4
solusi | 3*2=6 | 6*4=24
Total simpul daun: 10
```

Gambar 4.1 solusi kasus 1

```
simpul 1 angka: 4, 5 | 3+2=5
  simpul 1-1 angka: 9 | 3+2=5 | 5+4=9
  simpul 1-2 angka: 1 | 3+2=5 | 5-4=1
  simpul 1-3 angka: 20 | 3+2=5 | 5*4=20
simpul 2 angka: 4, 1 | 3-2=1
  simpul 2-1 angka: 5 | 3-2=1 | 4+1=5
  simpul 2-2 angka: 3 | 3-2=1 | 4-1=3
  simpul 2-3 angka: 4 | 3-2=1 | 4*1=4
  simpul 2-4 angka: 4 | 3-2=1 | 4/1=4
simpul 3 angka: 4, 6 | 3*2=6
  simpul 3-1 angka: 10 | 3*2=6 | 6+4=10
  simpul 3-2 angka: 2 | 3*2=6 | 6-4=2
  simpul 3-3 angka: 24 | 3*2=6 | 6*4=24
simpul 4 angka: 3, 6 | 4+2=6
simpul 5 angka: 3, 2 | 4-2=2
simpul 6 angka: 3, 8 | 4*2=8
simpul 7 angka: 3, 2 | 4/2=2
simpul 8 angka: 2, 7 | 4+3=7
simpul 9 angka: 2, 1 | 4-3=1
simpul 10 angka: 2, 12 | 4*3=12
```

Gambar 4.2 graf kasus 1

### B. *Kasus 2*

Pada kasus ini angka awal dari permainan berjumlah 3 yaitu {9, 6, 5, 8}. Seperti yang terlihat pada gambar 4.3 dan 4.3, solusi dari permainan sudah benar, dan dari 4! * 3! * $4^3$ = 9216 kemungkinan solusi, program hanya perlu mengecek 22 solusi untuk menemukan solusi yang tepat.

```
angka: 9, 6, 5, 8
solusi | 9+6=15 | 15/5=3 | 8*3=24
Total simpul daun: 22
```

Gambar 4.3 solusi kasus 2

```
simpul 1 angka: 5, 8, 15 | 9+6=15
  simpul 1-1 angka: 15, 13 | 9+6=15 | 8+5=13
    simpul 1-1-1 angka: 28 | 9+6=15 | 8+5=13 | 15+13=28
    simpul 1-1-2 angka: 2 | 9+6=15 | 8+5=13 | 15-13=2
    simpul 1-1-3 angka: 195 | 9+6=15 | 8+5=13 | 15*13=195
  simpul 1-2 angka: 15, 3 | 9+6=15 | 8-5=3
    simpul 1-2-1 angka: 18 | 9+6=15 | 8-5=3 | 15+3=18
    simpul 1-2-2 angka: 12 | 9+6=15 | 8-5=3 | 15-3=12
    simpul 1-2-3 angka: 45 | 9+6=15 | 8-5=3 | 15*3=45
    simpul 1-2-4 angka: 5 | 9+6=15 | 8-5=3 | 15/3=5
  simpul 1-3 angka: 15, 40 | 9+6=15 | 8*5=40
    simpul 1-3-1 angka: 55 | 9+6=15 | 8*5=40 | 40+15=55
    simpul 1-3-2 angka: 25 | 9+6=15 | 8*5=40 | 40-15=25
    simpul 1-3-3 angka: 600 | 9+6=15 | 8*5=40 | 40*15=600
```

Gambar 4.4 sebagian graf kasus 2

### C. *Kasus 3*

Pada kasus ini angka awal dari permainan berjumlah 3 yaitu {8, 6, 3, 3}. Seperti yang terlihat pada gambar 4.5 dan 4.6, solusi dari permainan sudah benar, dan dari 4! * 3! * $4^3$ = 9216 kemungkinan solusi, program hanya perlu mengecek 345 solusi untuk menemukan solusi yang tepat.

```
angka: 8, 6, 3, 3
solusi | 6+3=9 | 9*8=72 | 72/3=24
Total simpul daun: 345
```

Gambar 4.5 solusi kasus 3

```
simpul 1 angka: 3, 3, 14 | 8+6=14
  simpul 1-1 angka: 14, 6 | 8+6=14 | 3+3=6
    simpul 1-1-1 angka: 20 | 8+6=14 | 3+3=6 | 14+6=20
    simpul 1-1-2 angka: 8 | 8+6=14 | 3+3=6 | 14-6=8
    simpul 1-1-3 angka: 84 | 8+6=14 | 3+3=6 | 14*6=84
  simpul 1-2 angka: 14, 0 | 8+6=14 | 3-3=0
    simpul 1-2-1 angka: 14 | 8+6=14 | 3-3=0 | 14+0=14
    simpul 1-2-2 angka: 14 | 8+6=14 | 3-3=0 | 14-0=14
    simpul 1-2-3 angka: 0 | 8+6=14 | 3-3=0 | 14*0=0
  simpul 1-3 angka: 14, 9 | 8+6=14 | 3*3=9
    simpul 1-3-1 angka: 23 | 8+6=14 | 3*3=9 | 14+9=23
    simpul 1-3-2 angka: 5 | 8+6=14 | 3*3=9 | 14-9=5
    simpul 1-3-3 angka: 126 | 8+6=14 | 3*3=9 | 14*9=126
```

Gambar 4.6 sebagian graf kasus 3

### D. *Kasus 4*

Pada kasus ini angka awal dari permainan berjumlah 3 yaitu {10, 11, 11, 7}. Seperti yang terlihat pada gambar 4.7 dan 4.8,

solusi dari permainan sudah benar, dan dari 4! * 3! * $4^3$ = 9216 kemungkinan solusi, program hanya perlu mengecek 588 solusi untuk menentukan bahwa tidak ada solusi yang mungkin.

```
angka: 10, 11, 11, 7
tidak ada solusi
Total simpul daun: 588
```

Gambar 4.7 solusi kasus 4

```
simpul 5 angka: 11, 7, 1 | 11-10=1
  simpul 5-1 angka: 1, 18 | 11-10=1 | 11+7=18
    simpul 5-1-1 angka: 19 | 11-10=1 | 11+7=18 | 18+1=19
    simpul 5-1-2 angka: 17 | 11-10=1 | 11+7=18 | 18-1=17
    simpul 5-1-3 angka: 18 | 11-10=1 | 11+7=18 | 18*1=18
    simpul 5-1-4 angka: 18 | 11-10=1 | 11+7=18 | 18/1=18
  simpul 5-2 angka: 1, 4 | 11-10=1 | 11-7=4
    simpul 5-2-1 angka: 5 | 11-10=1 | 11-7=4 | 4+1=5
    simpul 5-2-2 angka: 3 | 11-10=1 | 11-7=4 | 4-1=3
    simpul 5-2-3 angka: 4 | 11-10=1 | 11-7=4 | 4*1=4
    simpul 5-2-4 angka: 4 | 11-10=1 | 11-7=4 | 4/1=4
  simpul 5-3 angka: 1, 77 | 11-10=1 | 11*7=77
    simpul 5-3-1 angka: 78 | 11-10=1 | 11*7=77 | 77+1=78
    simpul 5-3-2 angka: 76 | 11-10=1 | 11*7=77 | 77-1=76
    simpul 5-3-3 angka: 77 | 11-10=1 | 11*7=77 | 77*1=77
    simpul 5-3-4 angka: 77 | 11-10=1 | 11*7=77 | 77/1=77
```

Gambar 4.8 sebagian graf kasus 4

### *E. Kasus 5*

Pada kasus ini angka awal dari permainan berjumlah 3 yaitu {9, 6, 5, 8}. Seperti yang terlihat pada gambar 4.9 dan 4.10, solusi dari permainan sudah benar, dan dari 5! * 4! * $4^4$ = 737280 kemungkinan solusi, program hanya perlu mengecek 768 solusi untuk menemukan solusi yang tepat.

```
angka: 4, 11, 1, 8, 9
solusi | 11-4=7 | 8*1=8 | 9+7=16 | 16+8=24
Total simpul daun: 768
```

Gambar 4.9 solusi kasus 5

```
simpul 1-4-6 angka: 15, 72 | 11+4=15 | 8/1=8 | 9*8=72
  simpul 1-4-6-1 angka: 87 | 11+4=15 | 8/1=8 | 9*8=72 | 72+15=87
  simpul 1-4-6-2 angka: 57 | 11+4=15 | 8/1=8 | 9*8=72 | 72-15=57
  simpul 1-4-6-3 angka: 1080 | 11+4=15 | 8/1=8 | 9*8=72 | 72*15=1080
simpul 1-4-7 angka: 9, 23 | 11+4=15 | 8/1=8 | 15+8=23
  simpul 1-4-7-1 angka: 32 | 11+4=15 | 8/1=8 | 15+8=23 | 23+9=32
  simpul 1-4-7-2 angka: 14 | 11+4=15 | 8/1=8 | 15+8=23 | 23-9=14
  simpul 1-4-7-3 angka: 207 | 11+4=15 | 8/1=8 | 15+8=23 | 23*9=207
simpul 1-4-8 angka: 9, 7 | 11+4=15 | 8/1=8 | 15-8=7
  simpul 1-4-8-1 angka: 16 | 11+4=15 | 8/1=8 | 15-8=7 | 9+7=16
  simpul 1-4-8-2 angka: 2 | 11+4=15 | 8/1=8 | 15-8=7 | 9-7=2
  simpul 1-4-8-3 angka: 63 | 11+4=15 | 8/1=8 | 15-8=7 | 9*7=63
```

Gambar 4.10 sebagian graf kasus 5

## V. KESIMPULAN

Dari berbagai kasus uji coba yang telah dilakukan pada bab 4, terlihat bahwa algoritma runut-balik jauh lebih efisien dan efektif dibandingkan dengan algoritma *bruteforce* dalam mencari solusi dari permainan 24. Hal ini terjadi karena pada permainan 24, ada banyak aturan sehingga ada banyak kemungkinan solusi yang bisa dipangkas oleh algoritma runut-balik.

## VIDEO LINK AT YOUTUBE

https://youtu.be/KgNWW4A_qKw


## UCAPAN TERIMA KASIH

Segala puji dan syukur penulis panjatkan kepada Allah SWT. karena berkat rahmat dan hidayat-Nya penulis dapat menyelesaikan makalah ini dengan baik. Terima kasih juga saya ucapkan kepada bapak STIMA siapapun itu, karena berkat bimbingannyalah saya dapat mengerti berbagai macam ilmu dan mengaplikasikan ilmu tersebut dalam bentuk makalah ini.


## REFERENCES


[1] M. Yan Arie, "Perbandingan Algoritma Brute Force dan Algoritma Backtracking dalam Menyelesaikan Permainan Sudoku," unpublished.

[2] https://www.w3.org/2011/Talks/01-14-steven-phenotype/ diakses pada 9 Mei 2021.

[3] https://www.kompasiana.com/nprih/5a68a301ab12ae03d73ae622/bermain-dua-empat-berlatih-kreatif-merakit-alternatif?page=all diakses pada 10 Mei 2021.

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf diakses pada 9 Mei 2021.

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf diakses pada 9 Mei 2021.

[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf diakses pada 9 Mei 2021.


## 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

Syihabuddin Yahya Muhammad 13519149