Penerapan Algoritma *Depth First Search* dengan Teknik Heuristik pada Permainan Kartu 24

Aira thalca Avila Putra - 13520101
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520101@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas algoritma yang digunakan dalam penyelesaian permainan kartu 24. Algoritma penyelesaian terdiri dari algoritma Depth First Search dan dengan Teknik heuristik. Permainan Kartu 24 adalah salah satu jenis permainan yang digunakan menggunakan kartu remi. Permainan ini dimainkan oleh lebih dari satu orang dan bertujuan untuk berlomba mencari cara untuk membuat 4 kartu yang dikeluarkan pada setiap rondenya menjadi 24 dengan operasi kali, bagi, tambah, dan kurang. Terdapat beberapa algoritma yang dapat digunakan untuk menyelesaikan permaianan kartu 24. Penurunan, langkah-langkah, dan penjelasan algoritma akan dijelaskan secara rinci.**

Keywords—Heuristik, Depth First Search, Graf, Game 24, Algoritma

I. PENDAHULUAN

Saat ini, banyak sekali permainan kartu yang ada di dunia. Sebagai contoh, ada permaianan Poker, Solitaire, 41, Hearts, 24, dan lain-lain. Sejumlah ahli sejarah mengatakan bahwa permainan kartu ditemukan di Asia Barat oleh para pengembala sebagai evolusi dari permainan catur dengan menggunakan kerikil. Pendapat ini didasari dari permainan mahjong yang dianggap sebagai permainan kartu paling tua. Ahli lain berpendapat bahwa permainan kartu sudah ditemukan sejak abad ke 14 dengan adanya bukti sebuah manuskrip di Eropa yang menjelaskan bahwa ada sebuah permainan yang menggunakan kartu. Salah satu permainan kartu yang banyak dimainkan sekarang adalah permainan kartu 24

Gambar 2.1 Kartu Remi
Sumber : Grid.id

Permainan kartu 24 adalah jenis permainan kartu yang dapat mengasah otak manusia karena diperlukan pemikiran-pemikiran tertentu untuk menyelesaikan persoalannya. Permainan ini memiliki tujuan membentuk sebuah operasi aritmatika yang terbatas pada perkalian (x), penjumlahan (+), pembagian (/), dan pengurangan (-) serta penggunaan tanda kurung (()) pada 4 kartu yang merepresentasikan angka masing-masing yang mana hasil operasi aritmatika tersebut haruslah berjumlah 24.

Permainan ini cukup terkenal di kalangan remaja karena permainannya yang simpel namun tetap membutuhkan kemampuan berhitung dan kemampuan memikirkan strategi yang tepat untuk menyelesaikan persoalan ini. Permainan ini bisa dimainkan menggunakan kartu khusus yang memang digunakan untuk permainan 24, ataupun menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat jenis yaitu *spade*, *heart*, *clover*, dan *diamond.* Namun empat jenis kartu tersebut diabaikan pada permainan ini dan hanya nilai dari setiap kartulah yang digunakan. Sebagai contoh, kartu 2 *diamond* dan 2 *spade* keduanya bernilai 2. Kartu-kartu yang tidak memiliki nilai angka pada kartunya direpresentasikan sebagai berikut:

- kartu As bernilai 1
- kartu Jack bernilai 11
- kartu Queen bernilai 12
- kartu King bernilai 13.

II. DASAR TEORI

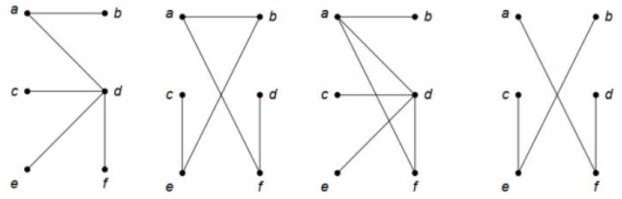
A. *Teknik Heuristik*

Heuristik berasal dari bahasa Yunani yaitu eureka yang berarti menemukan. Teknik heuristik adalah salah satu teknik yang digunakan untuk mempercepat pencarian solusi yang cukup banyak. Teknik ini digunakan untuk mengeliminasi beberapa kemungkinan solusi tanpa harus mengeksplorasi seluruh kemungkinan solusi secara penuh. Teknik ini biasanya dirancang untuk memecahkan persolan dengan mengabaikan apakah teknik tersebut terbukti benar secara matematis sebab teknik ini menggunakan pendekatan yang tidak formal, seperti berdasarkan penilaian intuitif, terkaan, atau akal sehat. Heuristik mengacu pada teknik memecahkan solusi berbasis

pengalaman, proses pembelajaran, dan penemuan solusi meskipun tidak dijamin optimal.

B. Pohon

Pohon adalah graf tak-berarah yang terhubung yang tidak mengandung sirkuit.



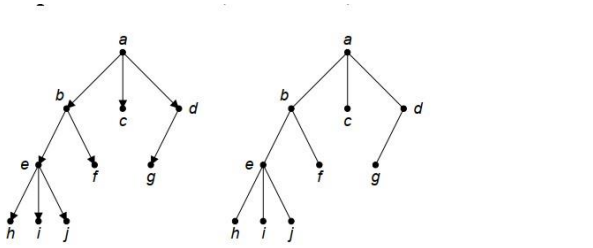
Gambar 2.1 Contoh Pohon dan bukan pohon
Sumber : Bahan Kuliah IF2120 Matematika Diskrit

Pada gambar diatas, graf ke 3 bukanlah pohon karena lintasan a,d,f membentuk sirkuit sementara graf ke 4 bukan pohon karena lintasan a,d,f dan c,e,b tidak terhubung.

Misalkan graf G = (V,E) adalah sebuah graf tak-berarah terhubung sederhana dan memiliki jumlah simpul n, maka G memiliki beberapa properti yaitu:

1. G adalah pohon.
2. G memiliki n-1 buah sisi.
3. G tidak mengandung sirkuit
4. Setiap pasang simpul di G hanya memiliki 1 lintasan yang menghubungkannya
5. G terhubung dan semua sisinya dalah jembatan (jika sisi tersebut diputus akan menyebabkan G menjadi dua pohon).

Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar (*rooted tree*).



Gambar 2.2 Contoh *Rooted Tree*
Sumber : Bahan Kuliah IF2120 Matematika Diskrit

Pohon berakar memiliki beberapa terminologi, yaitu:

1. Anak (*child/children)* dan orangtua (*parent)*

   Perhatikan gambar 2.1. simpul b,c, dan d adalah anak dari simpul a. Sebaliknya, simpul a adalah orangtua dari simpul b,c, dan d.
2. Lintasan (*path)*

   Lintasan dari simpul v_0 ke simpul v_n di dalam pohon G adalah kumpulan simpul yang menghubungkan v_0 dan v_n termasuk kedua simpul tersebut. Perhatikan kembali gambar 2.1. Lintasan dari simpul a ke simpul h adalah a,b,e,h dan memiliki panjang lintasan 3.
3. Saudara Kandung (*siblings)*

   Saudara kandung pada pohon G adalah kumpulan simpul yang memiliki orang tua yang sama. Perhatikan gambar 2.1. Simpul h memiliki saudara kandung simpul i dan simpul j.
4. Derajat (*degree)*

   Derajat sebuah simpul adalah jumlah anak pada simpul tersebut. Derajat dari sebuah pohon adalah derajat maksimum dari simpul yang ada di pohon tersebut.
5. Daun (*leaf)*

   Simpul yang berderajat nol atau tidak mempunyai anak disebut daun. Perhatikan kembali gambar 2.1. Simpul h, i, j, f, c, dan g adalah daun karena tidak memiliki anak.
6. Simpul dalam (*internal nodes)*

   Simpul yang mempunyai anak disebut simpul dalam. Perhatikan kembali gambar 2.1. Simpul a, b, d, dan e disebut simpul dalam karena memiliki anak.
7. Tinggi (*height)*

   Tingkat maksimum dari sebuah pohon adalah tinggi pohon tersebut. Pada gambar 2.1, pohon tersebut memiliki tinggi 3.

Pohon Dinamis adalah pohon yang terbentuk ketika pencarian dimulai, pada pohon dinamis, terdapat beberapa representasi, yaitu:

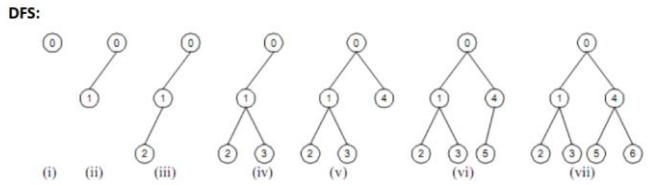
- Pohon ruang status: Seluruh simpul di dalam pohon dinamis
- Akar: *initial state*
- Daun: *goal state*
- Cabang: Operator/langkah dalam persoalan
- Ruang status: himpunan semua simpul
- Ruang solusi: himpunan status solusi
- Solusi: path ke status solusi

Pembangkitan status pada pohon dinamis dilakukan dengan cara mengaplikasikan langkah legal kepada simpul suatu

jalur. Jalur dari simpul akar ke simpul goal berisi rangkaian operator yang mengarah pada solusi persoalan.

C. Depth First Search

Algoritma Depth First Search merupakan salah satu penyelesaian masalah dengan cara melakukan traversal graf dengan cara yang sistematik. Pada DFS, algoritma mengutamakan pencarian terhadap anak dari node terlebih dahulu dibanding tetangga dari node itu. Pada algoritma DFS Dengan graf dinamis, pencarian solusi bisa dikatakan merupakan pembentukan sebuah pohon dinamis. Setiap simpul diperiksa apakah solusi telah tercapai atau tidak. Jika simpul merupakan solusi, pencarian dapat selesai atau dilanjutkan mencari solusi lain.



Gambar 2.3 Contoh Pembentukan Pohon Ruang Status pada DFS
Sumber : Bahan Kuliah IF2120 Strategi Algoritma

D. Permainan Kartu 24

Permainan kartu 24 adalah jenis permainan kartu yang dimainkan pada kartu remi dan ditemukan di Shanghai pada 1960-an. Permainan ini dimanikan dengan tujuan untuk mengubah 4 angka pada kartu yang tertera menjadi sebuah operasi matematika dan menghasilkan hasil 24. Operasi matematika yang diperbolehkan dalam permainan ini adalah kali, bagi, tambah, dan kurang. Kartu 2, 3, 4, 5, 6, 7, 8, dan 9 pada kartu remi bernilai sesuai dengan angkanya sementara kartu As bernilai 1, kartu Jack bernilai 11, kartu Queen bernilai 12, dan kartu King bernilai 13. Tata cara bermain permainan kartu 24 adalah sebagai berikut:

1. Pada awal permainan, pemain mengambil 4 kartu pada deck remi yang ada dengan nilai-nilai yang sudah disebutkan, yaitu kartu 2, 3, 4, 5, 6, 7, 8, dan 9 pada kartu remi bernilai sesuai dengan angkanya sementara kartu As bernilai 1, kartu Jack bernilai 11, kartu Queen bernilai 12, dan kartu King bernilai 13.
2. Pemain yang bermain mencoba mencari cara untuk membuat 4 angka yang terdapat pada kartu tersebut mencapai hasil 24 dengan operasi matematika perkalian, pembagian, penambahan, pengurangan, dan juga tanda kurung. Setiap kartu hanya boleh digunakan sekali dan setiap kartu harus digunakan.
3. Pemain yang paling cepat menyelesaikan persoalan tersebut akan mendapat poin
4. Permainan berakhir ketika kartu pada deck remi sudah habis.
5. Pemain yang mendapat poin paling banyak adalah pemenang dari permainan tersebut.

Permainan pun dapat digunakan tanpa menggunakan kartu remi, tapi dengan kartu khusus yang sudah tersedia

Gambar 2.4 Contoh kartu untuk permaianan kartu 24 tanpa kartu remi
Sumber : 24game.com

III. ANALISIS DAN PEMBAHASAN

A. Algoritma Depth First Search pada Permainan Kartu 24

Berikut ini adalah penjelasan algoritma yang saya gunakan untuk menyelesaikan permainan kartu 24. Ide dari algoritma ini adalah menggabungkan dua angka yang mungkin menjadi satu angka dan memanggil fungsi solve kembali dengan angka angka yang tersisa dan angka yang sudah digabung sehingga penyelesaian menjadi rekursif. Lalu saat kartu tersisa dua akan dilakukan pengecekan apakah dengan penambahan, pengurangan, pembagian, atau perkalian angka 24 dapat dicapai, jika ya maka itu solusinya jika tidak maka kita kembali (backtrack) ke node sebelumnya (kondisi sebelum digabung) dan mencoba penggabungan yang lain.

1. Diberikan empat angka yang menjadi persoalan. Empat angka ini menjadi akar simpul (*initial state)*
2. Akan dilakukan pengecekan secara heuristik apakah empat angka ini dapat mencapai angka 24 hanya dengan penambahan dan pengurangan, jika iya maka return ekspresi yang menghasilkan angka 24
3. Jika empat angka tadi tidak bisa menghasilkan angka 24 hanya dengan penambahan dan pengurangan, maka akan dilakukan algoritma DFS
4. Akan dilakukan pengecekan apakah kartu yang akan dicek sudah mencapai simpul daun. Dalam hal ini simpul daun berarti penggabungan 4 angka tersebut sudah menjadi dua angka.
5. Jika sudah mencapai simpul daun, akan dilakukan empat operasi dan akan dilakukan pengecekan apakah salah satu operasi menghasilkan angka 24, jika iya maka return ekspresi yang menghasilkan angka tersebut, jika tidak maka kembali (Backtrack) ke simpul parent dan dilakukan penggabungan dua angka lain.

6. Jika belum mencapai simpul daun, akan di generate semua pair yang mungkin dari list kartu yang tersedia.
7. Untuk setiap pair, akan di lakukan generate child dari parent, yaitu melakukan operasi tambah (a-b), kali(a x b), kurang (a-b), dan bagi(a/b jika b tidak 0) pada dua angka tersebut sebagai penggabungan. Copy list kartu ke sebuah list dan lakukan operasi hapus dua angka pada pair dari list baru dan masukkan hasil ekspresi tersebut ke dalam list baru. list ini yang menjadi child dari parent.
8. Untuk setiap list baru, akan dilakukan step dari 4-7 kembali.
9. Jika seluruh simpul sudah di generate dan tidak ada simpul daun yang merupakan solusi, maka tidak ada solusi untuk persoalan tersebut.

B. Implementasi

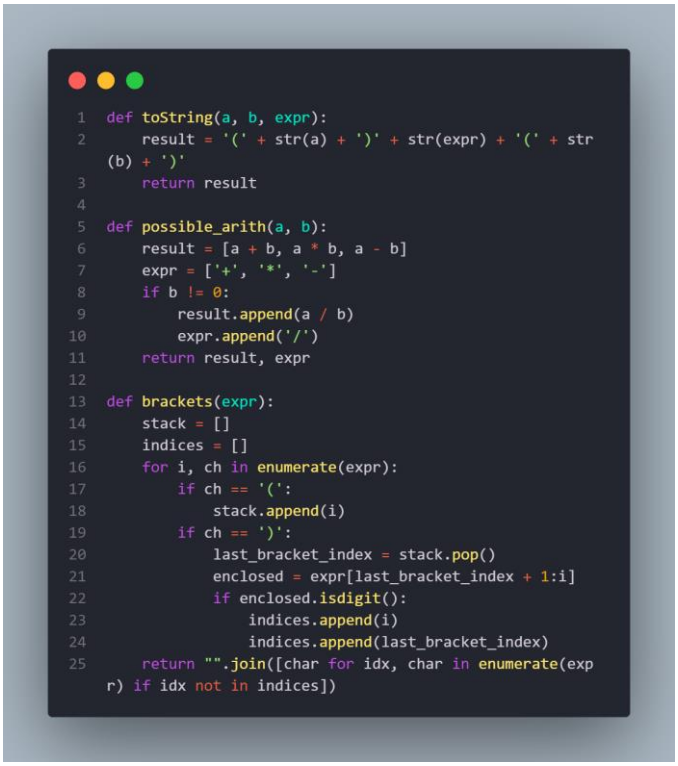
```python
import itertools

def solve(card, expresi=[]):
    if expresi == []:
        expresi = [str(n) for n in card]

    if sum(card) >= 24:
        if sum(card) == 24:
            a = expresi[0]
            for i in range(1, len(expresi)):
                a += "+"
                a += expresi[i]
            return a
        else:
            sisa = sum(card) - 24
            if sisa%2==0:
                sisa = sisa/2
                if (sisa in card):
                    index_sisa = card.index(sisa)
                    a = ""
                    for i in range(len(expresi)):
                        if (i == index_sisa):
                            a += "-"
                            a += expresi[i]
                        elif (i == 0):
                            a += expresi[i]
                        else:
                            a += "+"
                            a += expresi[i]
                    return a

    if len(card) == 1:
        if card[0] == 24:
            return card[0]
        else:
            return False

    elif len(card) == 2:
        for i in range(2):
            possible, exp_possible = possible_arith(card[i%2], card[(i+1)%2])
            for j in range (len(possible)):
                if abs(possible[j] - 24) < 1e-8:
                    return toString(expresi[i%2], expresi[(i+1)%2], exp_possible[j])
        return False
```

Bagian ini adalah bagian heuristik pada program, dapat dilihat bahwa sebelum melakukan generate child dari parent dan melakukan pencarian secara DFS, kita akan mencoba mengecek apakah list kartu sudah dapat menghasilkan nilai 24 hanya dengan penambahan dan pengurangan. Jika ya maka akan dibuat string sebagai output dari program yang berupa ekspresi dari operasi aritmatika yang menghasilkan nilai 24 dan program dapat langsung dihentikan. Selain itu akan dilakukan pula pengecekan apabila kartu yang diinput ternyata hanya satu angka. Yang terakhir, akan dilakukan pengecekan apakah simpul yang sedang diperiksa adalah simpul daun (dalam hal ini simpul daun berarti banyaknya angka pada list kartu tersisa 2). Jika ya maka akan dilakukan pengecekan kedua angka tersebut terhadap operasi perkalian, pembagian, penjumlahan, dan pengurangan. Apabila salah satu operasi tersebut menghasilkan angka 24, maka akan di panggil fungsi toString yang mengembalikan string berupa operasi yang dijalankan untuk mendapatkan nilai 24. Jika tidak maka simpul daun ini bukanlah solusi dari permasalahan ini.

```python
pairs = set(itertools.permutations(card, 2))

    for pair in pairs:
        possible, exp_possible = possible_arith(pair[0], pair[1])
        for i in range(len(possible)):
            exp_now = exp_possible[i]
            j = card.index(pair[0])
            if (pair[0] == pair[1]):
                k = card.index(pair[1], j + 1)
            else:
                k = card.index(pair[1])
            exp_string = toString(expresi[j], expresi[k], exp_now)

            card_now = card[:]
            expresi_now = expresi[:]

            card_now.pop(j)
            k = card_now.index(pair[1])
            card_now.pop(k)
            card_now.append(possible[i])

            expresi_now.pop(j)
            expresi_now.pop(k)
            expresi_now.append(exp_string)

            result = solve(card_now, expresi_now)
            if result:
                return brackets(result)
            else:
                continue
```

Kemudian jika simpul yang diperiksa bukan simpul daun, maka akan dilakukan permutasi 2 angka dari seluruh angka yang ada di list. Lalu untuk setiap pair, akan dilakukan operasi perkalian, penjumlahan, pembagian, dan pengurangan. Setelah itu untuk setiap operasi akan dibuat string yang merupakan operasi yang dilakukan terhadap pair. Kemudian untuk setiap operasi akan dibuat list kartu baru dan list

ekspresi baru yang merupakan copy dari list kartu dan ekspresi parent. Lalu kedua list tersebut akan menghapus dua elemen yang terdapat pada pair dan menambahkan hasil operasi pada pair tersebut kedalam list sehingga dibentuklah simpul child. Lalu child tersebut akan menjalankan proses solve lagi dan mencoba membuat child baru hingga sampai simpul daun sehingga algoritma yang didapat adalah algoritma Depth First Search karena mengutamakan penelusuran child terlebih dahulu dibanding neighbour.]



```python
def toString(a, b, expr):
    result = '(' + str(a) + ')' + str(expr) + '(' + str(b) + ')'
    return result

def possible_arith(a, b):
    result = [a + b, a * b, a - b]
    expr = ['+', '*', '-']
    if b != 0:
        result.append(a / b)
        expr.append('/')
    return result, expr

def brackets(expr):
    stack = []
    indices = []
    for i, ch in enumerate(expr):
        if ch == '(':
            stack.append(i)
        if ch == ')':
            last_bracket_index = stack.pop()
            enclosed = expr[last_bracket_index + 1:i]
            if enclosed.isdigit():
                indices.append(i)
                indices.append(last_bracket_index)
    return "".join([char for idx, char in enumerate(expr) if idx not in indices])
```

Ketiga fungsi ini adalah fungsi untuk membuat string yang nantinya akan menjadi output program dan generate seluruh kemungkinan operasi.

Fungsi toString akan membuat string operasi dari dua angka/ekspresi dengan character +, -, x, atau / bergantung operasi yang dilakukan.

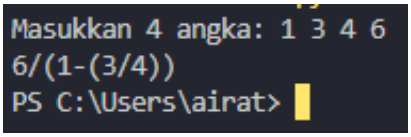
Fungsi possible_arith akan mengembalikan seluruh nilai dan juga character operasi dari setiap operasi yang mungkin yaitu penjumlahan, pengurangan, perkalian, dan pembagian jika penyebut bukan 0.

Fungsi brackets akan mengembalikan string ekspresi akhir yang sudah tidak memiliki bracket yang redundan sehingga output program lebih enak untuk dilihat.

C. Pengujian

Berikut ini adalah contoh kasus penyelesaian Game 24 pada beberapa kasus yang rumit untuk diselesaikan. Penulis menggunakan kasus yang terdapat pada website samidavies.

1. Card = 1, 3, 4, 6

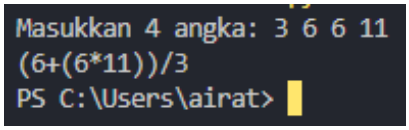


```
Masukkan 4 angka: 1 3 4 6
6/(1-(3/4))
PS C:\Users\airat>
```

Gambar 3.1 Hasil Test Case 1
Sumber : Dokumen Penulis

Solusi = 6 / (1 – ¾) = 24

2. Card = 3, 6, 6, 11

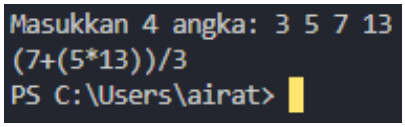


```
Masukkan 4 angka: 3 6 6 11
(6+(6*11))/3
PS C:\Users\airat>
```

Gambar 3.2 Hasil Test Case 2
Sumber : Dokumen Penulis

Solusi = (6 + 6*11) / 3 = 24

3. Card = 3, 5, 7, 13

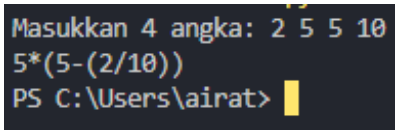


```
Masukkan 4 angka: 3 5 7 13
(7+(5*13))/3
PS C:\Users\airat>
```

Gambar 3.3 Hasil Test Case 3
Sumber : Dokumen Penulis

Solusi = (7 + 5*13) / 3 = 24

4. Card = 2, 5, 5, 10

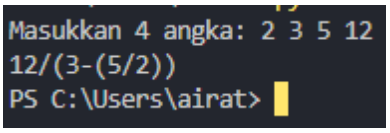


```
Masukkan 4 angka: 2 5 5 10
5*(5-(2/10))
PS C:\Users\airat>
```

Gambar 3.4 Hasil Test Case 4
Sumber : Dokumen Penulis

Solusi = 5 * (5 – 2/10) = 24

5. Card = 2, 3, 5, 12



```
Masukkan 4 angka: 2 3 5 12
12/(3-(5/2))
PS C:\Users\airat>
```

Gambar 3.5 Hasil Test Case 5
Sumber : Dokumen Penulis

Solusi = 12 / (3 – 5/2) = 24

IV. KESIMPULAN

Permainan kartu 24 merupakan permainan yang memerlukan keterampilan otak dan kemampuan berlogika yang cukup tinggi untuk dapat menyelesaikannya. Permainan ini dapat diselesaikan dengan algoritma brute force, greedy, backtracking, ataupun algoritma Depth First Search. Pada makalah ini, penulis menggunakan algoritma Depth First Search dengan optimasi pada bagian awal algoritma yaitu jika persoalan dapat diselesaikan cukup dengan penjumlahan dan pengurangan.

Penggunaan algoritma Depth First Search dengan menggunakan pohon dinamis sangat berguna dan mampu menyelesaikan persoalan permainan kartu 24 dengan sangat cepat dibandingkan menggunakan algoritma Brute Force. Pembangunan pohon ruang status dilakukan dengan menggabungkan dua angka menjadi satu angka dengan ekspresi tertentu hingga akhirnya tersisa dua angka saja (simpul daun) dan pengecekan apakah solusi sudah sesuai atau belum dan melakukan backtracking kembali ke simpul sebelumnya ada pada simpul daun.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu penciptaan makalah ini sehingga selesai tepat pada waktunya, terutama kepada Tuhan Yang Maha Esa karena tanpa karunianya penciptaan makalah ini tidak dapat terlaksana. Penulis juga mengucapkan terima kasih kepada Dr. Nur Ulfa Maulidevi, S.T, M.Sc selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas 02. Ucapan terima kasih juga kepada seluruh tim dosen IF2211 Strategi Algoritma yang tanpa bimbingannya makalah ini tidak akan pernah terwujud.

Penulis juga mengucapkan terima kasih yang sebesar besarnya kepada tim asisten IF2211 Strategi Algoritma yang telah dengan sabar memberi ilmu dan melakukan peniliaian serta mengoreksi kesalahan pada ujian ataupun tugas.

REFERENSI

[1] Munir,Rinaldi. 2021. *Pohon (Bag. 1): Bahan Kuliah IF2120 Matematika Diskrit*. Merupakan slide bahan ajar perkuliahan yang diunduh dari Website Kuliah pada tanggal 21 Mei 2022.

[2] Munir,Rinaldi. 2021. *Pohon (Bag. 2): Bahan Kuliah IF2120 Matematika Diskrit*. Merupakan slide bahan ajar perkuliahan yang diunduh dari Website Kuliah pada tanggal 21 Mei 2022.

[3] Munir,Rinaldi. 2022. *Algoritma Backtracking (Bag. 1): Bahan Kuliah IF2211 Matematika Diskrit.* Merupakan slide bahan ajar perkuliahan yang diunduh dari Website Kuliah pada tanggal 21 Mei 2022.

[4] Munir,Rinaldi. 2022. *Algoritma Greedy (Bag. 1): Bahan Kuliah IF2211 Matematika Diskrit*. Merupakan slide bahan ajar perkuliahan yang diunduh dari Website Kuliah pada tanggal 21 Mei 2022.

[5] Munir,Rinaldi. 2022. *Breadth First Search dan Depth First Search (Bag. 1): Bahan Kuliah IF2211 Matematika Diskrit.* Merupakan slide bahan ajar perkuliahan yang diunduh dari Website Kuliah pada tanggal 21 Mei 2022.

[6] Munir,Rinaldi. 2022. *Breadth First Search dan Depth First Search (Bag. 2): Bahan Kuliah IF2211 Matematika Diskrit.* Merupakan slide bahan ajar perkuliahan yang diunduh dari Website Kuliah pada tanggal 21 Mei 2022.

[7] Davies,Sami. 2017/ The Hardest Game of 24 . Diakses pada tanggal 21 Mei 2022.

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, 20 Mei 2022

Aira Thalca Avila Putra 13520101