Aplikasi Rekursi dan Pohon Keputusan dalam Variasi *Counterfeit Coin Puzzle*

Muhammad Alif Putra Yasa - 13520135[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520135@std.stei.itb.ac.id

***Abstract*—*Counterfeit Coin Puzzle* atau *Balancing Puzzle* merupakan permasalahan dimana tersedia beberapa objek seperti koin yang mana salah satunya berbeda dari yang lain, dan tantangannya adalah untuk menentukan koin mana yang berbeda. Teka-teki ini memiliki beberapa variasi dan artikel ini akan membahas cara untuk menyelesaikan teka-teki ini menggunakan rekursi dan relasi rekurens.**

***Keywords*—*Balancing Puzzle*, *Counterfeit Coin Puzzle*, Pohon Keputusan, Rekursi.**

I. PENDAHULUAN

Counterfeit Coin Puzzle merupakan teka-teki yang melipatkan sejumlah koin dan dengan menimbang berat relatif dari koin-koin tersebut, akan ditentukan koin mana yang memiliki berat yang berbeda.

Teka-teki ini pertama kali muncul di jurnal *American Mathematical Monthly* edisi tahun 1945 sebagai berikut:

"Given 9 coins, one of them fake and lighter, find the fake coin in two weighings on a balance scale."[1]

Berbagai variasi dari teka-teki ini pun kemudian bermunculan, salah satunya di jurnal *American Mathematical Monthly* edisi tahun 1946 sebagai berikut:

"There are 12 coins; one of them is fake. All real coins weigh the same. The fake coin is either lighter or heavier than the real coins. Find the fake coin and figure out whether it is heavier or lighter in 3 weighings on a balance scale."

Kedua teka-teki ini memiliki sejumlah kesamaan. Keduanya melibatkan penimbangan koin dan pencarian koin yang palsu, tetapi di teka-teki kedua, hanya dikatakan bahwa koin yang palsu memiliki berat yang berbeda dari koin yang asli.

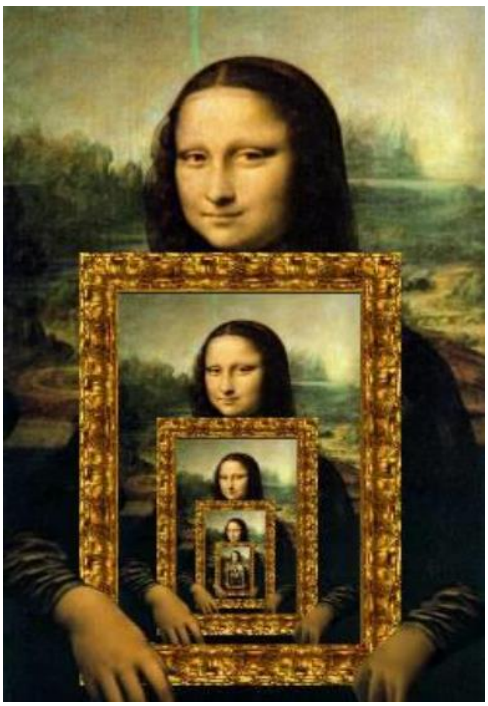
Di artikel ini, akan dibahas generalisasi dari tiga variasi dari teka-teki ini yaitu sebagai berikut:

1. Diberikan *N* buah koin, salah satunya merupakan koin palsu dan lebih berat daripada koin asli. Gunakan prinsip rekursi untuk mencari koin yang palsu penimbangan yang minimal.
2. Diberikan *N* buah koin, salah satunya merupakan koin palsu dan memiliki berat yang berbeda daripada koin asli. Gunakan prinsip rekursi untuk mencari koin yang palsu penimbangan yang minimal.
3. Diberikan *N* buah koin, paling banyak salah satunya merupakan koin palsu dan memiliki berat yang berbeda daripada koin asli. Gunakan prinsip rekursi untuk mencari koin yang palsu penimbangan yang minimal.

II. LANDASAN TEORI

A. Rekursi

Sebuah objek bersifat rekursif apabila objek tersebut didefinisikan menggunakan dirinya sendiri.



Gambar 1. Lukisan Mona Lisa memegang lukisan Mona Lisa (Sumber: *Slide* kuliah Rekursi dan Relasi Rekurens bagian 1, halaman 5)

B. Fungsi Rekursif

Fungsi Rekursif didefinisikan oleh dua bagian yaitu:

1. Basis

   Basis terdiri dari nilai fungsi yang terdefenisi secara eksplisit. Sebuah fungsi rekursif akan berhenti memanggil dirinya sendiri di Basis.
2. Rekurens

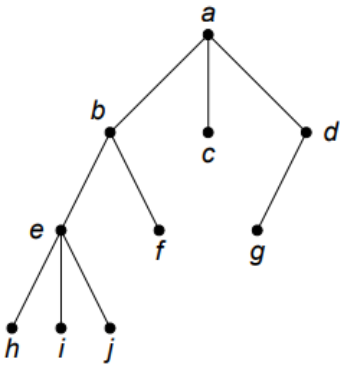
   Rekurens mendefinisikan fungsi dalam terminologi dirinya sendiri. Berisi kaidah untuk menemukan nilai fungsi pada suatu input dari nilai-nilai lainnya pada input yang lebih kecil.

C. Pohon

Pohon merupakan graf tak-berarah terhubung yang tidak mengandung sirkuit. Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n. Maka, pohon memenuhi syarat-syarat sebagai berikut:

1. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
2. G terhubung dan memiliki $m = n - 1$ buah sisi.
3. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
4. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
5. G terhubung dan semua sisinya adalah jembatan.

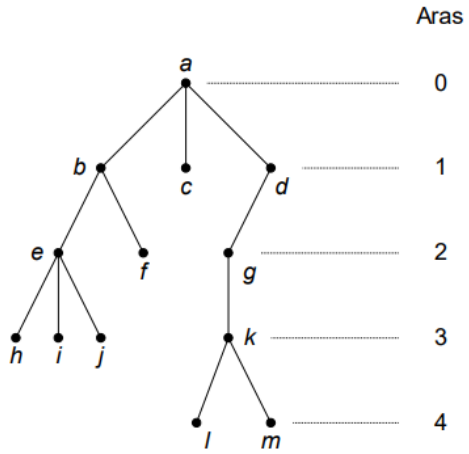
Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar (*rooted tree*).



Gambar 2. Contoh pohon dengan *a* sebagai akar
(Sumber: *Slide* kuliah Pohon bagian 2, halaman 2)

D. Terminologi Pohon Berakar

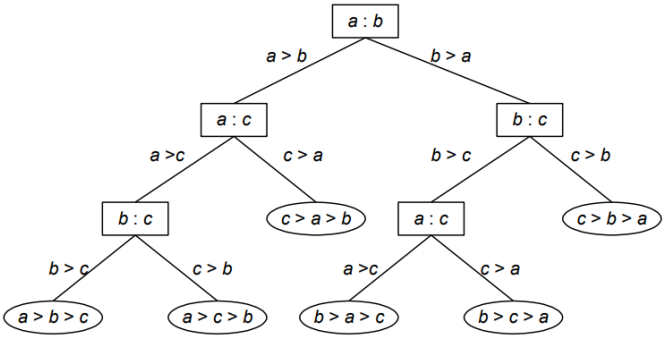
1. Lintasan (*Path*)
   Lintasan merupakan urutan sisi yang menghubungkan dua simpul.
2. Akar (*Root*)
   Akar merupakan simpul teratas dari sebuah pohon. Setiap pohon memiliki satu akar dan satu lintasan dari akar ke simpul manapun.
3. Orangtua (*Parent*)
   Simpul yang memiliki sisi kebawah menuju simpul anak (*child*).
4. Anak (*Child*)
   Simpul anak merupakan simpul dibawah simpul Orangtua (*parent*) dan dihubungkan dengan sisi.
5. Daun (*Leaf*)
   Daun adalah simpul yang tidak memiliki anak.
6. Aras (*level*)
   Aras merepresentasikan kedalaman simpul. Simpul akar berada di aras ke-0.



Gambar 3. Contoh pohon dengan tujuh aras
(Sumber: *Slide* kuliah Pohon bagian 2, halaman 9)

E. Pohon Keputusan

Pohon Keputusan merupakan jenis pohon dimana setiap simpul dalam menandakan sebuah tes, setiap cabang menandakan hasil dari tes, dan setiap daun menandakan hasil akhir atau konklusi.



Gambar 4. Contoh pohon keputusan
(Sumber: *Slide* kuliah Pohon bagian 2, halaman 22)

III. PEMBAHASAN

A. Variasi Pertama

"Diberikan N buah koin, salah satunya merupakan koin palsu dan lebih berat daripada koin asli. Gunakan prinsip rekursi untuk mencari koin yang palsu penimbangan yang minimal."

Dalam penyelesaian menggunakan rekursi, diperlukan basis. Karena terdapat tepat satu koin palsu, pada variasi pertama ini, basis yang digunakan adalah pasa saat ada 1 koin, maka itu koin yang palsu.

Rekurens dari fungsi ini adalah untuk membagi kumpulan koin menjadi kumpulan yang lebih kecil. Di bagian ini, kumpulan koin akan dibagi menjadi tiga, A, B, dan C. A dan B memiliki jumlah koin yang sama, lalu C berisi sisa kumpulan koin yang tidak termasuk di A atau B. Dengan cara ini, Saat dilakukan penimbangan A dengan B, akan ada tiga hasil sebagai berikut:

1. Berat A lebih besar dari berat B
   Besarnya berat A menandakan bahwa koin yang palsu berada dalam tumpukan A, sehingga dapat dilakukan pencarian kembali menggunakan fungsi yang sama untuk kumpulan A.

2. Berat B lebih besar dari berat A

   Besarnya berat B menandakan bahwa koin yang palsu berada dalam tumpukan B, sehingga dapat dilakukan pencarian kembali menggunakan fungsi yang sama untuk kumpulan B.

3. Berat B sama dengan berat A

   Sama beratnya A dan B menandakan bahwa koin palsu berada dalam tumpukan C, sehingga dapat dilakukan pencarian kembali menggunakan fungsi yang sama untuk kumpulan B.

Secara garis besar, fungsi dapat dinyatakan sebagai pseudocode berikut:

```
function var1(TKoin):
{
  TKoin adalah kumpulan koin
  fungsi length menghitung banyaknya koin
  fungsi weight menghitung berat tumpukan
}
  if length(TKoin) == 1:
    output(TKoin)
  else:
  { Pisahkan koin menjadi A, B, dan C }
    if weight(A) > weight(B):
      var1(A)
    else if weight(A) < weight(B):
      var1(B)
    else:
      var1(C)
```

Setiap rekursi, fungsi akan mengurangi tumpukan koin sebanyak sepertiganya.

Pembuatan pohon keputusan dapat dilakukan menggunakan fungsi yang sama, tetapi dihilangkan statement *if-else*-nya. Berikut kode python untuk meng-output pohon keputusan:

```
def printNode(depth, *args):
    if depth > 0:
        print("|   " * (depth - 1) + "|  ", *args)
    else:
        print(*args)

def var1rec(depth, l):
    length = len(l)
    if length > 1:
        binSize = math.ceil(length / 3)
        l1 = l[:binSize]
        l2 = l[binSize: 2 * binSize]
        l3 = l[2 * binSize:]

        printNode(depth, l1, ':', l2)

        printNode(depth, "-->", l1, '>', l2)
        var1rec(depth + 1, l1)

        printNode(depth, "-->", l1, '<', l2)
        var1rec(depth + 1, l2)

        printNode(depth, "-->", l1, '=', l2)
        var1rec(depth + 1, l3)
    else:
        printNode(depth, l)

def var1(l):
    printNode(0, l)
    var1rec(1, l)
```

Mengeksekusi `var1` untuk $N = 9$ (atau `l = [A..I]`) menghasilkan output berikut:

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
|   ['A', 'B', 'C'] : ['D', 'E', 'F']
|   --> ['A', 'B', 'C'] > ['D', 'E', 'F']
|   |   ['A'] : ['B']
|   |   --> ['A'] > ['B']
|   |   |   ['A']
|   |   --> ['A'] < ['B']
|   |   |   ['B']
|   |   --> ['A'] = ['B']
|   |   |   ['C']
|   --> ['A', 'B', 'C'] < ['D', 'E', 'F']
|   |   ['D'] : ['E']
|   |   --> ['D'] > ['E']
|   |   |   ['D']
|   |   --> ['D'] < ['E']
|   |   |   ['E']
|   |   --> ['D'] = ['E']
|   |   |   ['F']
|   --> ['A', 'B', 'C'] = ['D', 'E', 'F']
|   |   ['G'] : ['H']
|   |   --> ['G'] > ['H']
|   |   |   ['G']
|   |   --> ['G'] < ['H']
|   |   |   ['H']
|   |   --> ['G'] = ['H']
|   |   |   ['I']
```

Line yang dimulai dengan indentasi, yang merepresentasikan aras pohon. Setelah indentasi, ada tiga jenis output. Jika output tersebut memiliki '`:`', maka itu adalah sebuah simpul dalam, menandakan perbandingan. Output yang diawali dengan `-->` merupakan cabang, menandakan sebuah hasil dari perbadingan. Hasil selain itu merupakan daun, yang menandakan hasil akhir.

Dapat dilihat bahwa fungsi membentuk pohon *3-ary*, dan di setiap daun terdapat hasil yang didapatkan. Dari ini, kita juga tahu bahwa untuk menentukan koin yang palsu menggunakan rekursi, dibutuhkan paling banyak $\lceil \log_3(N) \rceil$ pengukuran.

B. Variasi Kedua

"Diberikan N buah koin, salah satunya merupakan koin palsu dan memiliki berat yang berbeda daripada koin asli. Gunakan prinsip rekursi untuk mencari koin yang palsu penimbangan yang minimal."

Variasi kali ini berbeda dengan variasi pertama, dimana di variasi ini koin yang palsu memiliki berat yang lebih atau kurang dari berat koin asli sehingga perbandingan langsung tidak dapat dilakukan. Untuk menyelesaikan soal ini, percabangan yang bisa dilakukan adalah ketika dua kumpulan koin memiliki berat yang sama atau berbeda. Hal lain yang perlu diingat adalah perlunya untuk mencatat koin yang asli. Hal ini akan membantu dalam perbandingan antara dua koin, karena tidak pastinya berat relatif koin palsu terhadap koin asli.

Basis pada fungsi variasi dua ini ada dua, ketika di tumpukan koin hanya ada satu koin, koin itu merupakan koin yang palsu dan ketika tumpukan koin kosong atau tumpukan koin asli kosong, meng-*output error*. Basis kedua tersebut diperlukan untuk meng-*cover* kasus N = 2, karena sifat koin palsu yang memiliki berat berbeda dengan koin asli, dan kita tidak memiliki koin asli sebagai referensi, akan ter-*output* bahwa hasil tidak bisa didapatkan.

Rekurens fungsi ini ada dua, ketika banyak koin di tumpukan tepat dua, dan ketika banyak koin di tumpukan lebih dari dua.

Rekurens untuk jumlah koin tepat dua diperlukan karena diperlukan penimbangan yang berbeda dengan kasus koin lebih dari dua.

Secara garis besar, fungsi dapat dinyatakan dengan pseudocode berikut:

```
function var2(TKoin, TAsli):
  {
    TKoin adalah kumpulan koin
    TAsli adalah tumpukan koin yang asli, awalnya kosong
    fungsi length menghitung banyaknya koin
    fungsi weight menghitung berat tumpukan
  }
    len = length(TKoin)
    if len > 2:
    {
      Bagi TKoin menjadi A, B, dan C. A dan B memiliki jumlah koin yang sama.
    }
      if weight(A) != weight(B):
        var2(A + B, TAsli + C)
      else:
        var2(C, TAsli + A + B)
    else if len == 1:
      { Output TKoin }
    else if len == 0 or isEmpty(TAsli):
      { Output Error / Kasus tidak mungkin }
    else:
    {
      Pisahkan TKoin menjadi A dan B, ambil C dari TAsli, lalu bandingkan satu persatu
    }
      if weight(A) != weight(C):
        var2(A, TAsli + B)
      else:
        var2(B, TAsli + A)
```

Dari fungsi ini, setiap pemanggilan fungsi, kumpulan koin akan menjadi sekitar seperdua dari sebelumnya.

Pohon keputusan untuk variasi ini dapat didapatkan dengan kode *python* berikut:

```
def var2rec(depth, l, real):
    # Fungsi printNode sama dengan sebelumnya
    length = len(l)
    if length > 2:

        binSize = math.ceil((length // 2) / 2)
        l1 = l[:binSize]
        l2 = l[binSize: 2 * binSize]
        l3 = l[2 * binSize:]

        printNode(depth, l1, ':', l2)

        printNode(depth, '-->', l1, '!=', l2)
        var2rec(depth + 1, l1 + l2, real + l3)

        printNode(depth, '-->', l1, '==', l2)
        var2rec(depth + 1, l3, real + l1 + l2)

    elif length == 1:
        printNode(depth, l)
    elif not real or length == 0:
        printNode(depth, [])
    else:
        l1 = [l[0]]
        l2 = [l[1]]
        l3 = [real[0]]

        printNode(depth, l1, ':', l3)

        printNode(depth, '-->', l1, '!=', l3)
        var2rec(depth + 1, l1, real + l2)

        printNode(depth, '-->', l1, '==', l3)
        var2rec(depth + 1, l2, real + l1)
def var2(l):
    printNode(0, l)
    var2rec(1, l, [])
```

Mengeksekusi `var2` untuk $N = 5$ atau `l = [A..E]` menghasilkan pohon berikut:

```
['A', 'B', 'C', 'D', 'E']
|   ['A'] : ['B']
|   --> ['A'] != ['B']
|   |   ['A'] : ['C']
|   |   --> ['A'] != ['C']
|   |   |   ['A']
|   |   --> ['A'] == ['C']
|   |   |   ['B']
|   --> ['A'] == ['B']
|   |   ['C'] : ['D']
|   |   --> ['C'] != ['D']
|   |   |   ['C'] : ['A']
|   |   |   --> ['C'] != ['A']
|   |   |   |   ['C']
|   |   |   --> ['C'] == ['A']
|   |   |   |   ['D']
|   |   --> ['C'] == ['D']
|   |   |   ['E']
```

Dapat dilihat dari hasil tersebut, ketika koin yang tersisa hanya *C* dan *D*, perbandingan koin biasa tidak memiliki makna, sehingga diperlukan perbandingan dengan koin referensi yang diambil dari *TAsli*.

Fungsi ini membentuk pohon *binary*, dan di setiap daun terdapat hasil yang didapatkan. Dari ini, kita juga tahu bahwa untuk menentukan koin yang palsu menggunakan rekursi, dibutuhkan paling banyak $\lceil \log_2(N) \rceil$ pengukuran.

C. Variasi Ketiga

"Diberikan N buah koin, paling banyak salah satunya merupakan koin palsu dan memiliki berat yang berbeda daripada koin asli. Gunakan prinsip rekursi untuk mencari koin yang palsu penimbangan yang minimal."

Variasi ini berbeda dengan variasi kedua, dimana di variasi kedua, terjamin bahwa ada satu koin palsu. Di variasi ini, bisa ada satu koin palsu atau tidak ada sama sekali.

Salah satu cara untuk mengerjakan variasi ini adalah dengan menggunakan solusi variasi kedua, ditambah perbandingan dengan koin referensi di akhir. Namun, kita dapat memotong perbandingan akhir tersebut menggunakan variabel boolean `knowHere`, dimana nilai `True` menandakan bahwa diketahui ada koin yang palsu di tumpukan yang diterima dan `False` menandakan bahwa tidak diketahui keberadaan koin palsu ditumpukan koin. Pada pemanggilan pertama, `knowHere` bernilai `False`.

Basis dari fungsi ini ada dua, menggunakan dua basis, yang pertama adalah saat banyak koin nol dan yang kedua adalah saat koin satu dan `knowHere` bernilai `True`.

Rekurens dari fungsi ini ada tiga, saat banyak koin lebih dari dua, saat banyak koin adalah satu dan `knowHere` bernilai `False`, dan saat banyak koin adalah dua.

Secara garis besar, fungsi dapat dinyatakan sebagai berikut:

```
function var3(TKoin, TAsli, knowHere):
  {
    TKoin adalah kumpulan koin
    TAsli adalah tumpukan koin yang asli, awalnya kosong
    knowHere bernilai True apabila diketahui adanya koin palsu di tumpukan koin
    fungsi length menghitung banyaknya koin
    fungsi weight menghitung berat tumpukan
  }
    len = length(TKoin)
    if len > 2:
    {
      Bagi TKoin menjadi A, B, dan C. A dan B memiliki jumlah koin yang sama.
    }
      if weight(A) != weight(B):
        var3(A + B, TAsli + C, True)
      else:
        var3(C, TAsli + A + B, knowHere)
    else if len == 1 and knowHere:
      { Output TKoin }
    else if len == 0 or isEmpty(TAsli):
      { Output Error / Kasus tidak mungkin }
    else if len == 1 and not knowHere:
      { R berisi koin referensi }
      if weight(TKoin) != weight(R):
        var3(TKoin, TAsli, True)
      else:
        { Output tidak ada koin palsu }
    else:
    {
      Pisahkan TKoin menjadi A dan B, ambil C dari TAsli, lalu bandingkan satu persatu
    }
      if weight(A) != weight(C):
        var3(A, TAsli + B, True)
      else:
        var3(B, TAsli + A, knowHere)
```

Berikut merupakan kode python untuk meng-output pohon keputusan variasi ketiga:

```
def var3rec(depth, l, real, knowHere):
    length = len(l)
    if length > 2:
        binSize = math.ceil((length // 2) / 2)

        l1 = l[:binSize]
        l2 = l[binSize: 2 * binSize]
        l3 = l[2 * binSize:]

        printNode(depth, l1, ':', l2)

        printNode(depth, '-->', l1, '!=', l2)
        var3rec(depth + 1, l1 + l2, real + l3, True)

        printNode(depth, '-->', l1, '==', l2)
        var3rec(depth + 1, l3, real + l1 + l2, knowHere)
    elif length == 1 and knowHere:
        printNode(depth, l)
    elif not real or length == 0:
        printNode(depth, "TIDAK BISA DITENTUKAN")
    elif length == 1 and not knowHere:
        l3 = [real[0]]
        printNode(depth, l, ':', l3)

        printNode(depth, '-->', l, '!=', l3)
        var3rec(depth + 1, l, real, not knowHere)

        printNode(depth, '-->', l, '==', l3)
        printNode(depth + 1, "TIDAK ADA")
    else:
        l1 = [l[0]]
        l2 = [l[1]]
        l3 = [real[0]]
        printNode(depth, l1, ':', l3)

        printNode(depth, '-->', l1, '!=', l3)
        var3rec(depth + 1, l1, real + l2, True)

        printNode(depth, '-->', l1, '==', l3)
        var3rec(depth + 1, l2, real + l1, knowHere)
def var3(l):
    printNode(0, l)
    var3rec(1, l, [], False)
```

Mengeksekusi `var3` untuk $N = 3$ atau `l = [A..C]` menghasilkan pohon berikut:

```
['A', 'B', 'C']
|   ['A'] : ['B']
|   --> ['A'] != ['B']
|   |   ['A'] : ['C']
|   |   --> ['A'] != ['C']
|   |   |   ['A']
|   |   --> ['A'] == ['C']
|   |   |   ['B']
|   --> ['A'] == ['B']
|   |   ['C'] : ['A']
|   |   --> ['C'] != ['A']
|   |   |   ['C']
|   |   --> ['C'] == ['A']
|   |   |   TIDAK ADA
```

Fungsi ini membentuk pohon *binary*, dan di setiap daun terdapat hasil yang didapatkan. Dari hasil ini, kita juga tahu bahwa untuk menentukan adanya atau lokasi koin yang palsu menggunakan rekursi, dibutuhkan paling banyak $\lceil \log_2(N+1) \rceil$ pengukuran.

IV. KESIMPULAN

Counterfeit Coin Puzzle adalah teka-teki yang terlihat simpel, tetapi saat diteliti lebih dalam, memiliki konsep-konsep yang menarik dan relevan ke matematika, terutama matematika diskrit.

Variasi dari teka-teki ini pun dapat diselesaikan dengan cara rekursi, yaitu dengan membagi permasalahan ke yang lebih kecil sampai ke kasus dasar atau basis. Dengan rekursi, fungsi memerlukan kasus dasar atau basis dimana hasilnya dinyatakan secara eksplisit dan rekurens yang mengubah input yang pada akhirnya akan sampai ke basis.

Pohon keputusan dapat dihasilkan dengan fungsi yang sama untuk menentukan hasil, tetapi tanpa percabangan *if*. Tanpa percabangan if, fungsi rekursi akan meng-output semua kemungkinan yang ada.

V. UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan ke hadirat Allah SWT atas berkat rahmat dan karunia-Nya artikel berjudul “Aplikasi Rekursi dan Pohon Keputusan dalam Variasi *Counterfeit Coin Puzzle*” dapat diselesaikan dengan tepat waktu.

Ucapan terima kasih penulis sampaikan kepada Ibu Dr. Nur Ulfa Maulidevi, S.T., M. Sc. selaku dosen pengajar Mata Kuliah IF2120 Matematika Diskrit Kelas 03 yang telah membimbing

mahasiswanya memahami materi perkuliahan. Beliau juga tidak henti-hentinya mengingatkan mahasiswa untuk menulis artikel agar dapat selesai tepat waktu.

Ucapan terima kasih penulis sampaikan kepada kepada Bapak Dr. Ir. Rinaldi Munir, M.T., Situs yang Beliau kelola menjadi salah satu sumber utama penulisan artikel ini karena terdapat banyak informasi dan dokumentasi yang terarsip.

Terakhir, penulis mengucapkan terima kasih kepada keluarga dan kolega yang telah membantu dalam mendukung penyelesaian makalah ini.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Rekursi-dan-relasi-rekurens-(Bagian-1).pdf (Diakses pada 12 Desember 2021, 10.00 WIB)

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf (Diakses pada 12 Desember 2021, 10.00 WIB)

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag2.pdf (Diakses pada 12 Desember 2021, 10.00 WIB)

[4] E. D. Schell, Problem E651—Weighed and found wanting, America Mathematical Monthly ed. 52, 1945, pp. 42.

[5] D. Eves, Problem E712—The extended coin problem, America Mathematical Monthly ed 53, 1946 pp. 156.

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.

Batam, 12 Desember 2021

Muhammad Alif Putra Yasa - 13520135