Penggunaan Graf dalam Pemodelan Matematis Permainan Delapan Jari

Evan [1)]

1) Program Studi Teknik Informatika ITB, Bandung, email: evangozali@yahoo.com

Abstract – *Makalah ini membahas aplikasi graf dalam memodelkan permainan delapan jari. Dengan menggunakan graf dan bantuan komputasi komputer, kita dapat menentukan beberapa sifat dari permainan delapan jari. Sifat yang akan dibahas di sini adalah sifat kemenangan mutlak permainan, yaitu apakah permainan delapan jari dapat dimenangkan dengan pasti oleh salah satu pemain jika menggunakan algoritma yang tepat.*

Kata Kunci*: permainan delapan jari, kemenangan mutlak, keadaan, graf permainan, properti.*

1. PENDAHULUAN

Permainan delapan jari merupakan permainan sederhana yang cukup populer di kalangan pelajar SMP dan SMA. Permainan ini dimainkan oleh 2 orang dengan menggunakan 8 jari yang dimiliki masing-masing pemain (semua jari kecuali jempol). Jari pemain dapat berada dalam keadaan tertutup atau terbuka (menunjuk). Untuk mempersingkat penulisan, penulis menggunakan istilah *jari* untuk menyatakan jari yang terbuka, *jari kiri* menyatakan jari-jari terbuka di tangan kiri, dan *jari kanan* menyatakan jari-jari terbuka di tangan kanan.

Pada awal permainan, kedua pemain memiliki 1 jari kiri dan 1 jari kanan. Dengan seperangkat aturan yang ada, kedua pemain bergerak bergiliran sampai salah satu pemain kalah, yaitu tidak dapat melakukan gerakan yang valid.

Pada setiap giliran, ada 2 pilihan gerak yang mungkin dilakukan seorang pemain. Yang pertama adalah menyentuh salah satu tangan lawan dengan salah satu tangan pemain, dengan syarat tangan yang menyentuh maupun tangan yang disentuh harus memiliki sedikitnya 1 jari. Banyak jari tangan lawan yang disentuh harus ditambahkan dengan banyak jari tangan yang menyentuh lalu dimodulo 5. Misalnya tangan kiri pemain A dengan 3 jari menyentuh tangan kiri pemain B dengan 3 jari, maka jari kiri pemain B ditambahkan 3 dan dimodulo 5, hasilnya adalah (3 + 3) mod 5 = 1.

Gerakan kedua adalah mengubah banyaknya jari pada kedua tangan tanpa mengubah jumlah totalnya, dengan syarat pemain tidak boleh hanya menukar jumlah jari pada kedua tangan. Misalnya (1, 3) dapat diubah menjadi (0, 4), (2, 2), atau (4, 0), tapi tidak boleh diubah menjadi (3, 1).

Permainan berakhir saat salah satu pemain memiliki 0 jari kiri dan 0 jari kanan, sebab pada keadaan tersebut pemain sudah tidak dapat melakukan gerakan.

Permasalahan yang ingin dipecahkan adalah apakah permainan 8 jari ini dapat dimenangkan oleh salah satu pemain secara mutlak jika menggunakan algoritma yang tepat.

2. PEMODELAN MATEMATIS

Untuk memecahkan masalah di atas, kita perlu mempertimbangkan semua pilihan gerakan yang mungkin dilakukan oleh pemain. Tapi kemungkinan gerakan tersebut cukup banyak sehingga akan sulit jika kita mencoba menuliskan dan menelusurinya satu per satu. Karena itu penulis menggunakan bantuan komputasi komputer untuk menyelesaikan permasalahan di atas. Untuk melakukan komputasi, kita perlu memodelkan terlebih dahulu permainan delapan jari ke dalam sebuah model matematis yang dapat dikomputasi oleh komputer.

Dalam bab ini penulis akan memaparkan langkah-langkah penyederhanakan permainan delapan jari menjadi sebuah model matematis.

2.1. Keadaan pemain

Keadaan seorang pemain pada suatu saat dideskripsikan oleh banyak jari kiri dan jari kanan yang dimilikinya. Tapi penempatan jari kiri dan jari kanan tidak signifikan, yaitu keadaan (a, b) akan sama saja dengan (b, a) karena tidak mempengaruhi pilihan langkah yang dapat dilakukan pemain. Dengan demikian kita dapat menyederhanakan keadaan seorang pemain menjadi pasangan berurut (x, y) dengan $x \geq y$. Di sini x dan y sudah tidak berkorespondensi dengan jari kiri dan jari kanan. Jadi pemain dengan 2 jari kiri dan 4 jari kanan, ataupun pemain dengan 4 jari kiri dan 2 jari kanan, akan dimodelkan sebagai (4, 2).

Banyaknya keadaan yang mungkin dialami oleh seorang pemain adalah banyaknya bilangan x dan y yang memenuhi $4 \geq x \geq y \geq 0$. Langkah perhitungannya:

Misal $a = 4 - x$, $a \geq 0$

$b = x - y$, $b \geq 0$

Maka $a + b + y = (4 - x) + (x - y) + y = 4$

Persamaan di atas adalah kombinasi berulang dengan banyaknya kombinasi C (4+3-1, 4) = C (6, 4) = 15. Jadi ada 15 keadaan yang mungkin dialami seorang pemain.

2.2. Keadaan permainan

Keadaan permainan pada suatu saat dideskripsikan oleh keadaan masing-masing pemain dan siapa pemain yang memegang giliran bergerak. Ada sedikitnya dua cara untuk memodelkan keadaan permainan ini.

Cara pertama adalah memodelkan keadaan permainan ke dalam 3 elemen (k1, k2, g) di mana k1 menyatakan keadaan *pemain pertama*[1], k2 menyatakan keadaan *pemain kedua*, dan g menyatakan nomor pemain yang memegang giliran.

Cara kedua adalah memodelkan keadaan ke dalam 2 elemen saja yaitu (s1, s2) di mana s1 menyatakan keadaan pemain yang memegang giliran, dan s2 menyatakan keadaan pemain yang tidak memegang giliran.

Pemodelan pertama memberikan jumlah kasus yang lebih banyak dari pemodelan kedua karena mengandung elemen *giliran*. Oleh karena itu, penulis memilih pemodelan kedua untuk menyederhanakan komputasi.

Sebagai contoh, misal pemain pertama berada pada keadaan (3, 2), pemain kedua (4, 1), dan pemain kedua yang memegang giliran. Keadaan tersebut dimodelkan menjadi ((4, 1), (3, 2)). Perhatikan bahwa jika pemain pertama berada pada keadaan (4, 1), pemain kedua (3, 2), dan pemain pertama memegang giliran, keadaan tersebut akan dimodelkan menjadi ((4, 1), (3, 2)) juga. Dengan demikian model ini tidak memberitahukan kepada kita pemain mana yang sedang memegang giliran. Tapi model ini cukup untuk menyelesaikan permasalahan kita. Kita akan melihat buktinya pada bab **metode**.

Dengan demikian pada model ini permainan akan dimulai pada keadaan ((1, 1), (1, 1)) dan berakhir pada keadaan ((0, 0), (a, b)), di mana (a, b) ≠ (0, 0), karena pada keadaan ((0, 0), (a, b)) pemain yang memegang giliran sudah tidak dapat melakukan gerakan.

Banyaknya kemungkinan keadaan permainan pada model ini adalah banyaknya kemungkinan s1 dikali banyaknya kemungkinan s2. Tapi kita lihat bahwa keadaan ((a, b), (0, 0)) tidak mungkin terjadi. Oleh karena itu hanya ada 14 kemungkinan untuk s2. Jadi banyaknya kemungkinan keadaan permainan ada 15 x 14 = 210. Jumlah yang cukup banyak untuk ditelusuri satu per satu tanpa bantuan komputer.

2.3. Alur permainan

Alur permainan dapat dimodelkan menjadi sebuah graf sederhana berarah. Simpul menyatakan keadaan permainan, dan sisi berarah menyatakan hubungan antara 2 simpul. Jika sebuah sisi menghubungkan simpul A **ke** simpul B, C, dan D, artinya langkah yang dibuat pemain pada keadaan A mungkin menghasilkan keadaan B, C, atau D. Mulai sekarang penulis menggunakan istilah **simpul tujuan** dari A untuk menyatakan simpul-simpul yang dihubungkan oleh sebuah sisi dari arah A, dan **simpul asal** dari A adalah simpul-simpul yang dihubungkan oleh sebuah sisi ke A.

Permainan dimulai di simpul ((1, 1), (1, 1)) dan berakhir pada simpul ((0, 0), (a, b)). Karena itu simpul ((0, 0), (a, b)) tidak memiliki simpul tujuan.

Untuk membangun sisi-sisi pada graf, kita perlu memperhitungkan semua kemungkinan langkah yang dapat diambil oleh pemain pada suatu keadaan. Seperti telah dibahas sebelumnya, ada 2 macam langkah yang dapat diambil oleh seorang pemain.

1. Menyentuh tangan lawan, dengan syarat jari tangan yang menyentuh dan jari tangan yang disentuh tidak nol. Ada 4 kemungkinan langkah yang bisa diambil, yaitu tangan kiri menyentuh tangan kiri, tangan kiri menyentuh tangan kanan, tangan kanan menyentuh tangan kiri, dan tangan kanan menyentuh tangan kanan. Pada model keadaan yang kita miliki, suatu keadaan ((a, b), (c, d)) dapat berubah menjadi maksimal 4 keadaan, yaitu:
   (((c + a) mod 5, d)[2], (a, b))
   (((c + b) mod 5, d), (a, b))
   ((c, (d + a) mod 5), (a, b))
   ((c, (d + b) mod 5), (a, b))
   Karena 1 ≤ a, b ≤ 4, maka c atau d **pasti** berubah. Dengan demikian langkah ini mengubah keadaan (s1, s2) menjadi (s3, s1) di mana s2 ≠ s3.
2. Mengubah banyaknya jari dengan tetap mempertahankan jumlah jari. Keadaan ((a, b), (c, d)) dapat berubah menjadi ((c, d), (e, f)) di mana a+b = e+f dan (a, b) ≠ (e, f). Banyaknya kemungkinan yang ada tergantung dari jumlah a+b. Karena (a, b) ≠ (e, f), langkah ini mengubah keadaan (s1, s2) menjadi (s2, s3) di mana s1 ≠ s3.

Mengapa penulis dapat mengatakan bahwa graf yang terbentuk adalah graf sederhana? Graf sederhana memiliki sifat tidak memiliki sisi ganda dan tidak memiliki gelang. Sifat yang pertama jelas terpenuhi karena kita tidak perlu membentuk 2 buah sisi yang sama. Bagaimana dengan sifat yang kedua? Sifat kedua akan terpenuhi jika tidak ada simpul yang memiliki dirinya sendiri sebagai simpul anak. Jadi kita harus membuktikan bahwa langkah apapun yang diambil oleh pemain dalam keadaan apapun tidak mungkin menghasilkan keadaan yang sama.

Hal ini dapat kita buktikan dengan membaginya ke dalam 2 kasus berikut:

1. Keadaan kedua pemain sama, kita simbolkan dengan (s1, s1). Jika pemain yang mendapat giliran mengambil langkah pertama, keadaan akan berubah menjadi (s2, s1) di mana s1 ≠ s2. Jika langkah kedua yang diambil, keadaan akan berubah menjadi (s1, s2) di mana s1 ≠ s2. Dengan demikian keadaan (s1, s1) tidak mungkin

[1] Dalam makalah ini yang dimaksud dengan *pemain pertama* adalah pemain yang mendapat giliran bergerak pertama di awal permainan, dan *pemain kedua* adalah pemain yang mendapat giliran kedua.

[2] Seharusnya keadaan yang mungkin adalah ((c + a) mod 5, d) atau (d, (c + a) mod 5), tergantung mana yang lebih besar, d atau (c + a) mod 5. Tapi untuk menyederhanakan penulisan, penulis hanya menuliskan 1 kemungkinan saja.

menghasilkan keadaan (s1, s1) lagi.

2. Keadaan kedua pemain berbeda, kita simbolkan dengan (s1, s2) di mana s1 ≠ s2. Jika langkah pertama yang diambil, keadaan akan berubah menjadi (s3, s1) di mana s2 ≠ s3. Karena s1 ≠ s2, keadaan yang dihasilkan pasti berbeda dengan keadaan awal. Jika langkah kedua yang diambil, keadaan akan berubah menjadi (s2, s3) di mana s1≠ s3. Karena s1 ≠ s2, keadaan yang dihasilkan pasti berbeda dengan keadaan awal.

Jadi pada kasus apapun, suatu keadaan tidak mungkin menghasilkan keadaan yang sama. Karena itu graf permainan adalah sebuah graf sederhana.

Di bawah ini adalah algoritma untuk membangun sisi-sisi graf jika diberikan simpul-simpul sebagai record <a, b, c, d : integer> yang menyatakan keadaan ((a, b), (c, d)), dan didefinisikan procedure buatsisi(x, y : simpul) adalah procedure untuk membuat sebuah sisi yang menghubungkan simpul x ke simpul y.

```
Procedure generatesisi(input N : set of
simpul)
  {Kamus}
  n : simpul
  m : simpul
  i : integer
  jum : integer
  {Algoritma}
  For each n in N do {n adalah simpul-simpul
dalam set N}
    {langkah pertama, ada 4 kemungkinan}
    m.c ← n.a
    m.d ← n.b
    if (n.c ≠ 0) and (n.a ≠ 0) then
      m.a ← max((n.c + n.a)mod 5, n.d)
      m.b ← min((n.c + n.a)mod 5, n.d)
      buatsisi(n, m)
    end if
    if (n.c ≠ 0) and (n.b ≠ 0) then
      m.a ← max((n.c + n.b)mod 5, n.d)
      m.b ← min((n.c + n.b)mod 5, n.d)
      buatsisi(n, m)
    end if
    if (n.d ≠ 0) and (n.a ≠ 0) then
      m.a ← max((n.d + n.a)mod 5, n.c)
      m.b ← min((n.d + n.a)mod 5, n.c)
      buatsisi(n, m)
    end if
    if (n.d ≠ 0) and (n.b ≠ 0) then
      m.a ← max((n.d + n.b)mod 5, n.c)
      m.b ← min((n.d + n.b)mod 5, n.c)
      buatsisi(n, m)
    end if
    {langkah kedua}
    m.a ← n.c
    m.b ← n.d
    jum ← n.a + n.b
    i ← min(jum, 4)
    while (i >= (jum – i)) and (i >= 0) do
      if (i ≠n.a) then
        m.c ← i
        m.d ← jum – i
        buatsisi(n, m)
      end if
      i ← i – 1
    end while
  End for
End procedure
```

3. METODE

Pada bab ini akan dibahas metode yang digunakan penulis untuk menyelesaikan permasalahan.

3.1. Properti simpul

Setiap simpul memiliki properti unik yang menentukan hasil akhir dari permainan. Ada 3 macam properti unik tersebut, penulis sebut M (*Menang*), K (*Kalah*), dan T (*Tidak tentu*).

Sebuah simpul memiliki properti M jika pada keadaan tersebut pemain yang memegang giliran memiliki kepastian untuk menang, apabila mengikuti langkah-langkah tertentu. Misalnya simpul ((1, 1), (4, 0)) memiliki properti M karena pemain yang memegang giliran tinggal menyentuh tangan pemain lawan sehingga keadaan pemain lawan berubah menjadi (0,0) yang berarti kemenangan bagi pemain yang memegang giliran.

Sebuah simpul memiliki properti K jika pada keadaan tersebut pemain yang memegang giliran memiliki kepastian untuk kalah, tidak peduli apapun langkah yang diambil, dengan asumsi bahwa pemain lawan selalu mengambil langkah terbaik yang menuju kepada kemenangannya. Misalnya simpul ((1,0),(3,0)) memiliki properti K karena permain yang memegang giliran harus menyentuh tangan lawan menjadi (4, 0), kemudian lawan dapat menyentuh tangan pemain menjadi (0, 0). Selain itu simpul ((0, 0), (a, b)) kita definisikan memiliki properti K juga karena pada keadaan ini pemain yang memegang giliran memang sudah kalah.

Yang terakhir, sebuah simpul memiliki properti T jika keadaan tersebut tidak memberikan kepastian kemenangan ataupun kekalahan bagi pemain yang memegang giliran.

Dengan demikian, tugas penulis adalah menentukan properti dari simpul awal, yaitu simpul ((1, 1), (1, 1)). Jika properti simpul awal adalah M, pemain pertama akan selalu menang. Jika properti simpul awal adalah K, pemain pertama akan selalu kalah. Jika properti simpul awal adalah T, maka permainan tidak dapat dimenangkan secara mutlak oleh salah satu pemain. Bagaimana cara penulis mencari properti simpul awal akan dijelaskan pada kedua subbab berikutnya.

3.2. Lemma properti

Di bawah ini akan penulis jabarkan lemma-lemma yang berkaitan dengan properti simpul.

Lemma 1: sebuah simpul memiliki properti M jika dan hanya jika memiliki setidaknya 1 simpul tujuan dengan properti K.

Bukti: cukup jelas. Pada keadaan tersebut, jika pemain yang memegang giliran memilih langkah yang menuju ke simpul berproperti K, maka pemain lawan memiliki kepastian untuk kalah, yang berarti pemain yang memegang giliran memiliki kepastian untuk menang.

Lemma 2: sebuah simpul memiliki properti K jika dan hanya jika tidak memiliki simpul tujuan atau semua simpul tujuannya berproperti M.

Bukti: simpul yang tidak memiliki simpul tujuan adalah simpul-simpul dengan keadaan ((0, 0), (a, b)), dan telah kita definisikan di atas bahwa simpul tersebut memiliki properti K. Itu bukti untuk kasus pertama. Untuk kasus kedua, jika semua simpul tujuan berproperti M, maka langkah apapun yang diambil oleh pemain yang memegang giliran akan menghasilkan kekalahan karena pada keadaan berikutnya pemain lawan memiliki kepastian menang.

Lemma 3: sebuah simpul memiliki properti T jika dan hanya jika memiliki simpul tujuan yang berproperti T.

Bukti: simpul yang memiliki properti T adalah simpul-simpul yang tidak memiliki properti M ataupun K. Karena simpul tidak memiliki properti K, maka simpul memenuhi ingkaran dari sifat properti K, yaitu memiliki minimal satu simpul tujuan dan memiliki minimal 1 simpul tujuan yang tidak berproperti M. Tapi karena simpul juga tidak memenuhi properti M, berarti simpul tidak boleh memiliki simpul tujuan berproperti K. Dengan demikian minimal 1 simpul tujuannya tidak boleh berproperti M ataupun K, yang berarti berproperti T.

3.3. Algoritma penelusuran

Dengan memanfaatkan lemma-lemma di atas, kita dapat menentukan properti setiap simpul. Caranya, pertama set properti simpul-simpul ((0, 0), (a, b)) menjadi K, dan simpul-simpul lainnya menjadi T. Kemudian untuk setiap simpul berproperti T, periksa apakah simpul tersebut memenuhi lemma 1 atau lemma 2. Jika ya, maka properti simpul diubah menjadi M untuk lemma 1 atau K untuk lemma 2. Akibatnya jumlah simpul berproperti T akan berkurang. Kemudian periksa lagi setiap simpul berproperti T dan ubah propertinya bila perlu. Langkah ini terus diulangi sampai tidak ada lagi simpul berproperti T yang berubah properti. Pada keadaan ini semua simpul telah memiliki properti yang seharusnya.

Algoritma di atas dapat dioptimisasi dengan cara cukup memeriksa simpul-simpul berproperti T yang memiliki setidaknya satu simpul tujuan berproperti M atau K, karena simpul berproperti T yang semua simpul tujuannya berproperti T jelas tidak perlu diperiksa. Di bawah ini diberikan algoritma yang sudah dioptimisasi, dengan mendefinisikan fungsi-fungsi berikut:

function prop(n: simpul) → (M, K, T)
{memberikan properti dari simpul n}
procedure setprop(n: simpul, x: (M, K, T))
{mengubah properti simpul n menjadi x}
function asal(n: simpul) → set of simpul
{menghasilkan himpunan simpul asal n}
function tujuan(n: simpul) → set of simpul
{menghasilkan himpunan simpul tujuan n}
procedure insert(s: set of simpul, n: simpul)
{memasukkan elemen n ke dalam himpunan s, jika belum ada}
procedure delete(s: set of simpul, n: simpul)
{menghapus elemen n dari himpunan s, jika ada}

```
procedure giveproperti(input N: set of
simpul)
  {Kamus}
  H, G : set of simpul
  n, m : simpul
  finish, kalah, menang : boolean
  i, j : integer
  {Algoritma}
  {pertama-tama berikan properti T kepada
semua simpul}
  for each n in N do
    setprop(n, T)
  end for
  {beri properti K ke simpul ((0,0),(a,b))
  n.a ← 0
  n.b ← 0
  for n.c ← 4 downto 0 do
    for n.d ← n.c downto 0 do
      setprop(n, K)
      for each m in asal(n) do
        insert(H, m)
      end for
    end for
  end for
  {telusuri simpul}
  repeat
    finish ← true
    G ← H
    for each n in G do
      kalah ← true
      menang ← false
      for each m in tujuan(n) do
        if prop(m) = K then
          menang ← true
          kalah ← false
        else if prop(m) = T then
          kalah ← false
        end if
      end for
      if menang then
        setprop(n, M)
      if kalah then
        setprop(n, K)
      end if
      if menang or kalah then
        finish ← false
        for each m in asal(n) do
          if prop(m) = T then
            insert(G, m)
          end if
        end for
        delete(G, n)
      end if
    end for
    H ← G
  until finish
end procedure
```

4. IMPLEMENTASI

Penulis menggunakan bahasa Pascal untuk mengimplementasikan algoritma di atas. Representasi graf yang digunakan adalah representasi dalam bentuk matriks. Untuk itu setiap keadaan permainan harus dipetakan menjadi sebuah bilangan integer yang unik. Cara pemetaannya akan dijelaskan di bawah ini.

4.1. Pemetaan keadaan pemain

Seperti dijelaskan pada bab 2, ada 15 keadaan pemain yang mungkin terjadi. Jadi keadaan pemain dapat dipetakan ke dalam 15 bilangan berbeda, yaitu angka 0 sampai 14. Berikut adalah tabel pemetaan keadaan pemain.

Tabel 1. Pemetaan keadaan pemain

State	Num	State	Num	State	Num
(0, 0)	0	(2, 2)	5	(4, 0)	10
(1, 0)	1	(3, 0)	6	(4, 1)	11
(1, 1)	2	(3, 1)	7	(4, 2)	12
(2, 0)	3	(3, 2)	8	(4, 3)	13
(2, 1)	4	(3, 3)	9	(4, 4)	14

Di bawah ini adalah algoritma untuk mengubah keadaan pemain menjadi sebuah bilangan, dan sebaliknya. Keadaan pemain disimpan sebagai record <a, b : integer>.

```
constant tabelsigma : array[0..5] of integer
= {0, 1, 3, 6, 10, 15}

function keadtoint(s : keadaan):integer
  → tabelsigma[s.a] + s.b
end function

function inttokead(x : integer):keadaan
  {Kamus}
  s : keadaan
  i : integer
  {Algoritma}
  i ← 0
  while (tabelsigma[i] ≤ x) do
    i ← i + 1
  end while
  s.a ← i – 1
  s.b ← x – tabelsigma[i-1]
  → s
end function
```

4.2. Pemetaan simpul

Pada bab 2 telah dijelaskan bahwa ada 210 keadaan permainan yang mungkin terjadi. Apabila simpul dinyatakan sebagai (s1, s2) di mana s1 dan s2 menyatakan keadaan pemain yang telah dipetakan menjadi integer, maka jangkauan s1 adalah 0-14 dan jangkauan s2 adalah 1-14, karena keadaan ((a, b), (0, 0)) tidak mungkin terjadi. Pemetaan simpul yang penulis ambil adalah s = 14 x s1 + s2. Dengan demikian jangkauan dari s adalah 1-210. Di bawah ini adalah algoritma untuk mengubah simpul menjadi integer dan sebaliknya, di mana simpul dinyatakan sebagai record <s1, s2 : integer>.

```
function simpultoint(s : simpul):integer
  → 14 * s1 + s2
end function

function inttosimpul(x : integer):simpul
  {Kamus}
  s : simpul
  {Algoritma}
  s.a ← (x – 1) div 14
  s.b ← (x – 1) mod 14 + 1
  → s
end function
```

5. HASIL DAN PEMBAHASAN

Setelah program dibuat dan setiap simpul telah diberikan properti, ternyata simpul ((1, 1), (1, 1)) memiliki properti T. Artinya permainan delapan jari tidak dapat dimenangkan secara mutlak oleh salah satu pemain.

Hasil yang mengejutkan penulis adalah, ternyata simpul ((1, 1), (2, 0)) berproperti M. Ini artinya jika di awal permainan pemain pertama memilih untuk mengubah jarinya menjadi (2, 0), pemain kedua memiliki kepastian untuk menang asalkan mengikuti langkah yang tepat.

6. KESIMPULAN

Kesimpulan dari makalah ini adalah permainan delapan jari tidak dapat dimenangkan secara mutlak oleh salah satu pemain. Tapi apabila pemain pertama salah mengambil langkah pertama, pemain kedua dapat dengan pasti memenangkan permainan.

7. PENUTUP

Saran dan kritik untuk penulis dapat ditujukan ke alamat email penulis. Apabila ada pembaca yang berminat melihat *source code* program juga dapat menyampaikan permintaannya melalui email. Apabila ada pembaca yang menemukan sesuatu yang salah dalam makalah ini, mohon menyampaikannya juga kepada penulis.

Untuk pengembangan selanjutnya, makalah ini dapat dijadikan dasar untuk membuat A.I. dari game permainan delapan jari dengan memanfaatkan algoritma-algoritma yang telah dibahas dalam makalah ini.

DAFTAR REFERENSI

[1] Ir. Rinaldi Munir, MT, *Diktat kuliah IF2153 Matematika Diskrit (Edisi Keempat*), Teknik Informatika ITB, 2003

LAMPIRAN: DAFTAR KEADAAN PERMAINAN DAN PROPERTINYA

0 0 1 0 K	2 0 1 0 M	3 0 1 0 M	3 3 1 0 M	4 2 1 0 M
0 0 1 1 K	2 0 1 1 T	3 0 1 1 K	3 3 1 1 T	4 2 1 1 M
0 0 2 0 K	2 0 2 0 T	3 0 2 0 M	3 3 2 0 M	4 2 2 0 M
0 0 2 1 K	2 0 2 1 T	3 0 2 1 M	3 3 2 1 M	4 2 2 1 M
0 0 2 2 K	2 0 2 2 T	3 0 2 2 T	3 3 2 2 T	4 2 2 2 T
0 0 3 0 K	2 0 3 0 M	3 0 3 0 M	3 3 3 0 M	4 2 3 0 M
0 0 3 1 K	2 0 3 1 M	3 0 3 1 T	3 3 3 1 M	4 2 3 1 M
0 0 3 2 K	2 0 3 2 T	3 0 3 2 K	3 3 3 2 M	4 2 3 2 T
0 0 3 3 K	2 0 3 3 T	3 0 3 3 K	3 3 3 3 T	4 2 3 3 T
0 0 4 0 K	2 0 4 0 M	3 0 4 0 K	3 3 4 0 T	4 2 4 0 M
0 0 4 1 K	2 0 4 1 T	3 0 4 1 T	3 3 4 1 M	4 2 4 1 T
0 0 4 2 K	2 0 4 2 M	3 0 4 2 T	3 3 4 2 T	4 2 4 2 T
0 0 4 3 K	2 0 4 3 T	3 0 4 3 K	3 3 4 3 T	4 2 4 3 T
0 0 4 4 K	2 0 4 4 T	3 0 4 4 T	3 3 4 4 T	4 2 4 4 T
1 0 1 0 K	2 1 1 0 M	3 1 1 0 M	4 0 1 0 M	4 3 1 0 M
1 0 1 1 K	2 1 1 1 T	3 1 1 1 T	4 0 1 1 T	4 3 1 1 M
1 0 2 0 K	2 1 2 0 K	3 1 2 0 M	4 0 2 0 T	4 3 2 0 M
1 0 2 1 K	2 1 2 1 T	3 1 2 1 M	4 0 2 1 M	4 3 2 1 M
1 0 2 2 K	2 1 2 2 K	3 1 2 2 T	4 0 2 2 T	4 3 2 2 T
1 0 3 0 K	2 1 3 0 M	3 1 3 0 M	4 0 3 0 T	4 3 3 0 M
1 0 3 1 K	2 1 3 1 M	3 1 3 1 T	4 0 3 1 M	4 3 3 1 M
1 0 3 2 K	2 1 3 2 T	3 1 3 2 T	4 0 3 2 T	4 3 3 2 M
1 0 3 3 K	2 1 3 3 K	3 1 3 3 T	4 0 3 3 T	4 3 3 3 T
1 0 4 0 M	2 1 4 0 M	3 1 4 0 M	4 0 4 0 M	4 3 4 0 M
1 0 4 1 M	2 1 4 1 M	3 1 4 1 M	4 0 4 1 T	4 3 4 1 T
1 0 4 2 K	2 1 4 2 T	3 1 4 2 T	4 0 4 2 T	4 3 4 2 T
1 0 4 3 K	2 1 4 3 T	3 1 4 3 T	4 0 4 3 T	4 3 4 3 T
1 0 4 4 K	2 1 4 4 T	3 1 4 4 T	4 0 4 4 T	4 3 4 4 T
1 1 1 0 M	2 2 1 0 M	3 2 1 0 M	4 1 1 0 M	4 4 1 0 M
1 1 1 1 T	2 2 1 1 T	3 2 1 1 T	4 1 1 1 T	4 4 1 1 M
1 1 2 0 M	2 2 2 0 T	3 2 2 0 M	4 1 2 0 T	4 4 2 0 M
1 1 2 1 M	2 2 2 1 T	3 2 2 1 M	4 1 2 1 T	4 4 2 1 T
1 1 2 2 T	2 2 2 2 T	3 2 2 2 T	4 1 2 2 T	4 4 2 2 T
1 1 3 0 T	2 2 3 0 M	3 2 3 0 M	4 1 3 0 M	4 4 3 0 T
1 1 3 1 T	2 2 3 1 M	3 2 3 1 M	4 1 3 1 T	4 4 3 1 T
1 1 3 2 T	2 2 3 2 T	3 2 3 2 M	4 1 3 2 T	4 4 3 2 T
1 1 3 3 K	2 2 3 3 T	3 2 3 3 M	4 1 3 3 T	4 4 3 3 T
1 1 4 0 M	2 2 4 0 M	3 2 4 0 M	4 1 4 0 M	4 4 4 0 T
1 1 4 1 M	2 2 4 1 T	3 2 4 1 T	4 1 4 1 T	4 4 4 1 T
1 1 4 2 T	2 2 4 2 M	3 2 4 2 T	4 1 4 2 T	4 4 4 2 T
1 1 4 3 M	2 2 4 3 T	3 2 4 3 T	4 1 4 3 T	4 4 4 3 T
1 1 4 4 T	2 2 4 4 T	3 2 4 4 T	4 1 4 4 T	4 4 4 4 T