PENGGUNAAN ALGORITMA BACKTRACKING DALAM PENCARIAN KOEFISIEN ROOK POLYNOMIAL

Arinta Primandini Auza
13505021

Program Studi Teknik Informatika
Institut Teknologi Bandung
Alamat : Jl Ganeca 10 Bandung
e-mail: if15021@students.if.itb.ac.id

ABSTRAK

Rook polynomial telah dipelajari sejak tahun 1946. Polinom ini menyediakan metode untuk menghitung permutasi dengan beberapa *constraint*. Sesuai dengan namanya, *rook polynomial* didasari oleh permainan catur dengan menempatkan k buah *rook* (benteng) pada sebuah papan catur yang besar berukuran hingga (missal n x n) sehingga tidak terdapat benteng yang menempati baris atau kolom yang sama. *Pada* bab-bab selanjutnya akan diberikan suatu permasalahan yang dapat diselesaikan dengan menggunakan *rook polynomial* dan bagaimana algoritma *backtracking* dapat mencari koefisien dari polinom yang merupakan solusi dari permasalah tersebut. *Rook polynomial* merupakan contoh lain dari *generating function*.

Seperti yang telah disebutkan sebelumnya, algoritma yang akan dibicarakan pada makalah ini adalah algoritma *backtracking* atau runut balik. Algoritma ini merupakan algoritma yang berbasis *Depth First Search (DFS)*. Dengan menggunakan algoritma ini program tidak harus mencari solusi dengan memeriksa semua kemungkinan solusi, seperti pada algoritma *Brute Force* tetapi hanya kemungkinan solusi yang mengarah pada solusi. Dengan demikian waktu untuk pencarian solusi dapat dihemat dan program menjadi lebih mangkus.

Kata kunci: *Rook polynomial*, *backtracking,* koefisien.

1. PENDAHULUAN

Rook (benteng) adalah salah satu pion pada permainan catur yang dapat bergerak secara horizontal atau vertikal. Sehingga jika terdapat benteng yang terletak pada kolom atau baris yang sama, maka benteng yang bergerak pada giliran selanjutnya akan mengalahkan benteng yang lain. Konsep dari *rook polynomial* ini mirip dengan konsep gerakan benteng pada permainan catur tersebut. Setiap benteng yang terdapat pada papan catur berukuran hingga diletakkan sedemikian rupa sehingga tidak terdapat benteng yang terletak pada kolom atau baris yang sama.

Rook polynomial menyediakan metode dalam menghitung permutasi dengan beberapa posisi yang terlarang. Studi mengenai metode ini dimulai pada tahun 1946 oleh Kaplansky dan Riordan dengan aplikasi pada permasalahan *card-matching* [1]. Metode ini sangat berhubungan dengan area lainnya dalam matematika seperti teori graf, deret *hypergeometric*, dan geometri aljabar.

Aplikasi dari metode ini sangat banyak seperti pada *Married Couples Problem, Dancing School Problem, Dinner-Diner Matching Problem, Job Allocation Problem, Derangement* dari suatu string, dan sebagainya. Sebagai contoh, misalkan terdapat 4 orang *A, B, C, D* akan diberikan tugas *a, b, c, d* sehingga setiap orang mendapatkan 1 pekerjaan. *A* tidak akan mengerjakan pekerjaan *b* dan *c*, *B* tidak akan mengerjakan pekerjaan *a*, *C* tidak akan mengerjakan pekerjaan *a*, *b*, dan *d*, *D* tidak akan mengerjakan pekerjaan *c* dan *d*. Dengan metode *rook polynomial* kita dapat mengetahui berapa banyak cara untuk mengalokasi 4 pekerjaan tersebut.

Pada makalah ini akan dijelaskan bagaimana mencari koefisien x^k dari *rook polynomial* yang berarti banyaknya cara menempatkan *k* buah benteng pada sebuah papan catur berukuran hingga *n* x *m* sehingga untuk *b* suatu benteng dari himpunan *k* buah benteng tidak terletak pada minimal 1 dari *k-1* benteng lainnya pada baris maupun kolom yang sama. Algoritma yang digunakan untuk menyelesaikan permasalahan ini adalah algoritma runut balik (*backtracking*) dengan mencari solusi dari ruang solusi dan menyimpan semua solusi pada himpunan solusi kemudian menghitung banyak anggota dari himpunan solusi tersebut. Perbedaan algoritma ini dengan algoritma *Exhaustive Search* adalah pada algoritma *Backtracking* tidak perlu mengecek semua kemungkinan pada ruang solusi, namun hanya mengecek kemungkinan solusi yang paling mendekati solusi.

2. METODE

Pada bab ini akan dijelaskan mengenai apa itu *rook polynomial* dan bagaimana cara merepresentasikan suatu persoalan dalam papan catur dan benteng, serta metode *backtracking* dalam membantu menghitung koefisien dari *rook polynomial* pada suatu permasalahan.

2.1 Rook Polynomial

Rook Polynomial adalah polinomial yang berbentuk

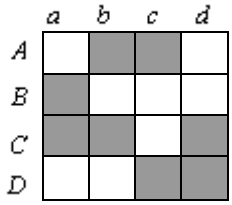
$$R_{mn}(x) = 1 + r_1^{(m,n)}x + r_2^{(m,n)}x^2 + ... + r_k^{(m,n)}x$$

atau

$$R_{mn}(x) = \sum_k r_k^{(m,n)} x^k$$

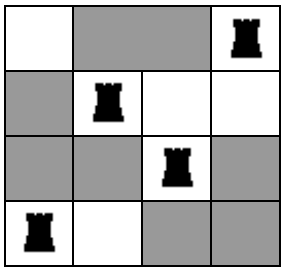
dimana koefisien $r_k^{(m,n)}$ adalah banyaknya cara menempatkan k buah benteng pada papan catur m x n sehingga tidak terdapat benteng yang saling menyerang [2].

Pada permasalahan pengalokasian pekerjaan yang telah disebutkan pada bab Pendahuluan di atas, maka kita dapat merepresentasikan permasalahan tersebut dengan papan catur berukuran 4 x4 pada gambar berikut.



Gambar 1. Papan catur permasalahan alokasi pekerjaan

Sel papan yang berwarna gelap merepresentasikan pekerjaan yang tidak boleh dikerjakan oleh orang yang berada pada bagian paling kiri. Contohnya pada sel (*b, A*) yang berarti *A* tidak boleh melakukan pekerjaan *b*. Kemudian untuk menggambarkan pekerjaan yang dikerjakan oleh masing-masing orang agar tiap orang mendapatkan pekerjaan yang berbeda dapat ditempatkan benteng pada masing-masing sel yang tidak berwarna gelap sesuai dengan *constraint* yang telah disebutkan di atas. Salah satu solusi dari permasalahan alokasi pekerjaan di atas adalah :



Gambar 2. Solusi permasalahan alokasi pekerjaan

Untuk mengetahui *rook polynomial* dari permasalahan di atas hitung banyaknya kemungkinan meletakkan 1 buah benteng, 2 buah benteng, 3 buah benteng, dan 4 buah benteng. Untuk nilai $k > 4$ tidak dapat ditempatkan pada papan karena banyak baris = banyak kolom = 4. Pada masing-masing baris maupun kolom ditempatkan maksimal satu benteng untuk mencegah penyerangan. Maka maksimal terdapat 4 tempat untuk menempatkan benteng-benteng tersebut. Berdasarkan *Pigeon Hole Principle* maka akan terdapat baris atau kolom yang berisi lebih dari 1 benteng.

Dengan penghitungan secara manual dapat diperoleh banyaknya cara menempatkan 1 benteng adalah 8 cara, untuk 2 benteng 19 cara, untuk 3 benteng 14 cara, dan untuk 4 benteng sebanyak 2 cara. Dengan demikian *rook polynomial* dari permasalah tersebut adalah :

$$1 + 8x + 19x^2 + 14x^3 + 2x^4$$

Untuk papan berukuran 4 x 4 dapat dengan mudah menentukan koefisien-koefisien dari *rook polynomial*. Untuk papan berukuran yang lebih besar akan lebih sulit. Karena itu untuk mempermudah penghitungan digunakan program yang dapat menghitung secara otomatis dengan menggunakan algoritma *backtracking*.

2.2 Algoritma Backtracking

Dalam penerapan algoritma runut balik (*backtracking*) dibutuhkan properti-properti berikut :

1. Solusi Persoalan
   Solusi dari persoalan dinyatakan dalam sebuah vektor

   $Y = (y_1, y_2, ..., y_n)$, $y_i \in S_i$. S_i tidak harus berbeda.

   S_i merupakan himpunan nilai y_i yang merupakan kemungkinan dari solusi. Misalkan $S_i = \{a, b\}$ maka $y_i = a$ atau b.

2. Fungsi Pembangkit nilai y_k.
   Fungsi untuk membangkitkan nilai untuk y_k yang merupakan komponen dari vektor solusi.

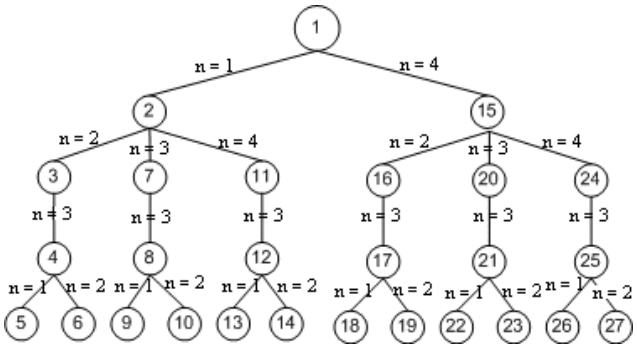
3. Fungsi Pembatas
   Fungsi yang menentukan apakah (y_1 , y_2, ..., y_k) mengarah pada vektor solusi atau tidak. Jika ya, maka pembangkitan untuk nilai y_{k+1}, tetapi jika tidak maka (y_1 , y_2, ..., y_k) dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi[3].

Sekarang mari tinjau persoalan pencarian koefisien bagi x^k pada *rook polynomial.* Sebagai contoh tinjau kembali permasalahan alokasi pekerjaan. Papan catur tempat meletakkan benteng direpresentasikan sebagai matriks berukuran 4 x 4. Setiap sel dari papan diberi index mulai (1, 1) sampai dengan (4 , 4). Index di depan koma merupakan kolom dan index di belakang koma adalah baris dari masing-masing sel. Index dimulai dari kiri atas sampai dengan kanan bawah.

(1,1)	(2,1)	(3,1)	(4,1)
(1,2)	(2,2)	(3,2)	(4,2)
(1,3)	(2,3)	(3,3)	(4,3)
(1,4)	(2,4)	(3,4)	(4,4)

Gambar 3. Representasi matriks papan catur

Misalkan n = kolom dan m = baris. Pohon ruang status untuk mencari solusi permasalahan tersebut adalah :



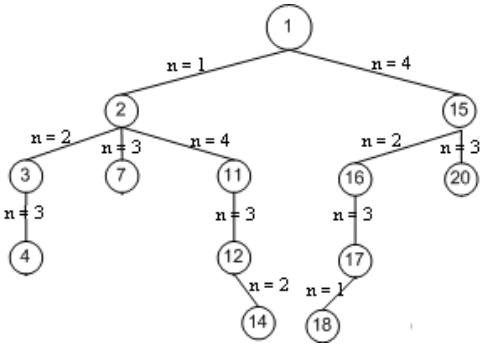
Gambar 4. Pohon ruang status permasalahan alokasi pekerjaan

Setiap benteng akan diletakkan mulai dari baris ke-1 hingga baris ke-4. Simpul pada level ke-*i* menandakan status penempatan benteng ke-i. Karena banyaknya benteng adalah 4 sama dengan banyak baris, maka setiap baris memiliki satu benteng. Karena itu pada pohon ruang status di atas nilai m tidak dicantumkan. Asumsikan nilai m = level. Sel yang berwarna gelap merupakan sel yang tidak bisa ditempati benteng. Karena itu pada level ke-1 hanya terdapat 2 kemungkinan n yaitu n = 1 atau n = 4 lalu pada level ke-2 hanya terdapat 3 kemungkinan, dan seterusnya hingga level terakhir. Jika menggunakan algoritma *Exhaustive Search* semua kemungkinaan solusi akan dicek kebenarannya sehingga memakan waktu lebih lama. Sedangkan jika menggunakan *Backtracking* jika terdapat nilai yang tidak sesuai dengan fungsi pembatas maka akan kembali ke status sebelumnya untuk melanjutkan pada simpul yang lain. Pada persoalan di atas dapat ditentukan nilai-nilai berikut :

$S_1 = \{1, 4\}$
$S_2 = \{2, 3, 4\}$
$S_3 = \{3\}$
$S_4 = \{1, 2\}$

Fungsi Pembangkit = $T(k + 1)$ = pembangkit nilai x_{k+1} sesuai dengan nilai yang terdapat pada S_{k+1}.
Fungsi Pembatas = $B(x_1, x_2, ..., x_k)$ = nilai x_{k+1} tidak sama dengan nilai $x_1, x_2, ..., x_k$.

Sehingga pohon dinamis yang terbentuk selama pencarian solusi persoalan tersebut adalah :



Gambar 5. Pohon dinamis pencarian permasalahan alokasi pekerjaan

Dari gambar dapat dilihat dengan jelas bahwa solusi dari persoalan alokasi pekerjaan adalah :

$Y_1 = \{1, 4, 3, 2\}$
$Y_2 = \{4, 2, 3, 1\}$
Himpunan solusi = $\{Y_1, Y_2\}$

Banyaknya anggota pada himpunan solusi adalah 2, dengan demikian koefisien x^4 dari *rook polynomial* pada permasalahan tersebut adalah 2. Untuk mencari koefisien dari x, x^2, dan x^3 dapat dibentuk pohon ruang status dan pohon dinamis dengan cara yang sama seperti pada pencarian koefisien x^3.

2.3 Pseudo Code

Struktur data yang digunakan dalam menyimpan sel-sel papan catur adalah matriks bertipe integer berukuran m x n yang akan dialokasi sesuai dengan kebutuhan. Untuk menandai daerah yang tidak boleh ditempati oleh benteng,

matriks akan diisi nilai 1 dan sisanya 0. Untuk menandai matriks yang telah ditempati benteng sel akan diisi nilai 2.

Sedangkan untuk menyimpan koefisien dari x^k digunakan *array* bertipe integer berukuran *n. Array* ke-*i* berisi koefisien dari x^i. Program akan mengecek mulai dari simpul pertama. Jika pada suatu simpul tidak terdapat nilai yang mengarah ke solusi maka kembali ke simpul sebelumnya hingga diperoleh nilai y_k yang mendekati solusi. Selama fungsi pembatas bernilai true maka program akan melanjutkan ke simpul selanjutnya hingga semua benteng ditempatkan pada papan catur. Pencarian solusi akan terus dilanjutkan hingga kembali ke akar dan tidak terdapat simpul yang mengarah ke solusi.

Berikut ini adalah algoritma *backtracking* pencarian koefisien *rook polynomial* dalam bahasa algoritmik :

```
Kamus Global :
  papanCatur : array of array of integer
  m, n : integer {n = #kolom, m = #baris}
  nbBenteng : integer
  resArea : array of <integer, integer>
  koefisien : array of integer
  i : integer

procedure InitMatriks(n,m:integer, rArea:array of
<integer, integer>)
{
  mengalokasikan matriks berukuran m x n dan
  mengisi setiap sel dengan angka 1 pada resArea
  yang merupakan sel yang tidak dapat diisi
  benteng. Sedangkan sel lainnya diisi 0
}

procedure Maju()
{
  melanjutkan ke simpul berikutnya dan mengisi
  sel matriks yang bersesuaian dengan nilai yang
  diberikan fungsi pembangkit dengan 2
}

procedure Backtrack()
{
  sel matriks yang merupakan current sel diisi
  dengan 0 dan kemudian kembali pada simpul
  sebelumnya
}

function Hitung(nbBent:integer) : integer
{
  menghitung solusi dari koefisien x^k sesuai
  dengan jumlah benteng
}

  Kamus lokal :
    benteng : integer = 2
    k : integer = 0
    nbSolusi : integer = 0

  Algoritma :
    while simpul > 0 do
    {simpul = 0 yaitu akar dari pohon ruang
    solusi}
      if k = nbBent then
        nbSolusi = nbSolusi + 1
      else
        x[simpul] = T(simpul)
        if terdapat x[simpul] and
        B(x[1],x[2],...,x[simpul]) = true then
          Maju()
          k = k + 1
        else
          Backtrack()
          k = k - 1

  return nbSolusi

Algoritma :
  Inisialisasi resArea
  InitMatriks(n,m,resArea)
  nbBenteng = 1

  koefisien[0] = 1
  i = 1
  while Hitung(nbBenteng)!= 0 do
    koefisien[i] = Hitung(nbBenteng)
    i = i + 1
```

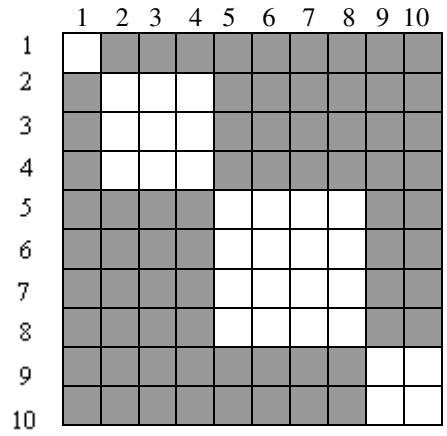
2.4 Beberapa Contoh Permasalahan

Terdapat banyak permasalahan yang dapat diselesaikan dengan menggunakan metode *rook polynomial*. Contoh dari permasalahan tersebut adalah :

a. Dinners-Diners Matching Problem

Terdapat N orang memesan makanan dan M jenis makanan untuk makan malam yang dipesan oleh N orang tersebut($M \leq N$). Kemudian pelayan akan memberikan makanan kepada pemesan secara random. Yang menjadi permasalahan adalah berapa peluang makanan yang dipesan diberikan pada pemesan yang tepat.

Misalkan terdapat 1 orang memesan *pasta vegetarian,* 3 orang memesan *chicken cordon bleu*, 4 orang memesan *prime rib*, 2 orang memesan *filet of sole*. Maka terdapat 10 orang yang memesan makanan. Perhatikan gambar berikut :



Gambar 6. Papan catur pada *Dinner-Diner Matching Problem*

Kolom menyatakan orang yang memesan makanan dan baris menyatakan makanan yang dipesan. Makanan ke-1 adalah *pasta vegetarian*, makanan 2 sampai dengan 4 adalah *chicken cordon bleu*, makanan 5 sampai dengan 8 adalah *prime rib*, dan 9 hingga 10 adalah *filet of sole*. Kotak berwarna hitam berarti makanan yang bukan merupakan pesanan dari orang pada kolom yang bersesuaian (*restricted area*), sedangkan kotak putih sebaliknya. Banyaknya permutasi makanan yang diterima adalah $\frac{10!}{2!.3!.4!} =$ 12600. Sedangkan banyak kemungkinan makanan diberikan pada pemesan yang tepat adalah dengan menghitung banyaknya cara menempatkan 10 buah benteng pada papan catur di atas sehingga memenuhi tidak ada benteng yang saling menyerang. Sehingga peluangnya adalah banyak kemungkinan menempatkan benteng / banyak permutasi makanan yang diterima.

Untuk mencari *rook polynomial* dari persoalan di atas, tentukan area terlarang kemudian inisialisasi ukuran matriks dan area terlarang tersebut. Kemudian program dapat dijalankan untuk menghitung semua koefisien dari polinom.

b. Married Couples Problem

Persoalan ini dikenal juga dengan sebutan ménage problem. Tujuannya adalah menghitung banyaknya cara menempatkan *n* pasangan suami-istri pada sebuah meja bundar sehingga selalu terdapat seorang pria di antara dua wanita dan tidak satupun dari suami duduk bersebelahan dengan istrinya[4].

c. Dancing School Problem

Terdapat sebuah grup beranggotakan *n* orang gadis berbaris dengan jarak suatu bilangan bulat dari *m* sampai dengan $m + n - 1$ cm dan grup $n + h$ orang pemuda yang berbaris dengan jarak dari *m* hingga $m + n + h - 1$ cm. Dapat dilihat dengan jelas bahwa *m* adalah jarak minimal bagi gadis maupun pemuda.

Lokasinya adalah pada suatu sekolah menari. Guru tari memilih sebuah grup beranggotakan *n* orang dari $n + h$ orang pemuda. Seorang gadis dengan jarak *l* dapat memilih teman menari dari grup *n* pemuda tersebut, pemuda pada jarak yang sama dengan dirinya, atau jarak yang lebih besar tapi lebih kecil atau sama dengan $l + h$. Tujuan dari permasalahan ini adalah menghitung berapa banyak pasangan yang dapat terbentuk.

3. KESIMPULAN

Metode *rook polynomial* dapat menyelesaikan banyak permasalahan kombinatorika yang menyangkut penghitungan permutasi dari suatu persoalan dengan beberapa area terlarang. Terdapat banyak sekali cara untuk menentukan koefisien dari polinomial ini, namun untuk mempermudah dan mempercepat pekerjaan dibuat suatu program komputer yang dapat digunakan dalam memecahkan persoalan. Algoritma *backtracking* telah cukup mangkus dalam penyelesain permasalahan tersebut karena tidak semua kemungkinan solusi diperiksa kebenarannya.

REFERENSI

[1] arxiv.org/pdf/math.CO/0407007

[2] http://mathworld.wolfram.com/topics/RooksProblem.html

[3] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", ITB, 2005.

[4] http://mathworld.wolfram.com/topics/MarriedCouplesProblem.html