Algoritma *Branch and Bound* dalam Penyelesaian Persoalan Matriks *Math Magic*

Hasanul Hakim[1], Ilham Fatoni[2], Nazar Iskandar Fajri[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14091@students.if.itb.ac.id[1], if14121@students.if.itb.ac.id[2], if14151@students.if.itb.ac.id[3]

Abstrak

Matriks adalah salah satu bentuk array yang berdimensi dua. Sampai saat ini, sudah banyak sekali persoalan matematika yang bisa diselesaikan dengan menggunakan konsep matriks. Salah satu contohnya adalah sistem persamaan linear. Penyelesain persoalan ini terbukti lebih cepat jika kita menggunakan matriks daripada diselesaikan secara biasa. Matriks itu sendiri mempunyai bentuk – bentuk tersendiri seperti layaknya sebuah bidang persegi atau persegi panjang. Khusus pada bentuk persegi terdapat suatu keajaiban (*math magic*) yang pada awalnya hanya dipakai untuk permainan mengolah otak biasa. Hal ini terjadi dengan cara menuliskan angka dari 1 sampai n x n untuk matriks berukuran n x n sehingga akan didapat jumlah baris perbaris, kolom perkolom, dan semua diagonal utamanya sama. Namun, tidak semua bentuk matriks n x n dapat menghasilkan keajaiban ini. Hanya matriks yang berukuran n x n di mana n-nya ganjil saja yang bisa. Salah satu syarat yang harus dipenuhi adalah angka yang dimasukkan itu tidak boleh berulang dan bilangan terbesar atau n x n itu tidak boleh terletak pada diagonal utama matriks. Pada makalah ini akan dibahas dan dianalisis kelebihan Algoritma *Branch and Bound* terhadap Algoritma *Brute Force* dalam pemecahan masalah ini.

***Kata kunci**: matriks,branch and bound,brute force, kolom,baris,diagonal utama, ganjil, kombinas, math magic*

1. Pendahuluan

Matriks adalah larik berdimensi dua yang dapat memberikan banyak keistimewaan dalam merepresentasikan persoalan-persoalan matematik dan komputasi. Salah satunya adalah permainan pengombinasian angka berbeda ke dalam matriks untuk mendapatkan kombinasi yang cocok sedemikian sehingga hasil penjumlahan untuk bilangan-bilangan yang terletak pada satu baris sama dengan hasil penjumlahan bilangan-bilangan pada baris lain, pada suatu kolom untuk setiap kolom, dan pada satu diagonal utama yang sama. Persoalan dibatasi untuk n=2k-1, k=1,2,3,4,.... , n adalah dimensi matriks. Diperlukan suatu algoritma efektif untuk memecahkan pesoalan ini alih-alih menggunakan algoritma yang lempang(*Brute Force*) yang memiliki kompleksitas besar.

2. Branch and Bound

2.1. Prinsip Branch and Bound

Sebagaimana pada algortima runut-balik, algoritma *Branch & Bound* juga merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang Solusi diorganisasikan ke dalam pohon ruang status. Pembentukan pohon ruang status. Pembentukan pohon ruang status pada algoritma B&B berbeda dengan pembentukan pohon pada algoritma runut-balik. Bila pada algoritma runut-balik ruang solusi dibangun secara Depth-First Search(DFS), maka pada algoritma B&B ruang solusi dibangun dengan skema Breadth-First Search (BFS).

Pada algoritma B&B, pencarian ke simpul solusi dapat dipercepat dengan memilih simpul hidup berdasarkan nilai ongkos (cost). Setiap simpul hidup diasosiasikan dengan sebuah ongkos yang menyatakan nilai batas (bound). Pada prakteknya, nilai batas untuk setiap simpul umumnya berupa taksiran atau perkiraan.

Fungsi heuristik untuk menghitung taksiran nilai tersebut dinyatakan secara umum sebagai :

$$\hat{c}(i) = f(i) + \hat{g}(i)$$

yang dalam hal ini,

$\hat{c}(i)$ = ongkos untuk simpul i
$f(i)$ = ongkos mencapai simpul i dari akar
$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul akar i

Nilai $\hat{c}$ digunakan untuk mengurutkan pencarian. Simpul berikutnya yang dipilih untuk diekspansi adalah simpul yang memiliki $\hat{c}$ minimum.

2.2. Algoritma Branch and Bound

Algoritma B&B jika dituliskan secara prosedural adalah sbb :

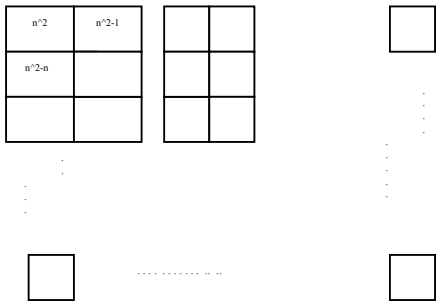
1. Masukkan simpul akar ke dalam antrian *Q*. Jika simpul akar adalah simpul solusi (*goal node*), maka solusi telah ditemukan. Stop.
2. Jika *Q* kosong, tidak ada solusi . Stop.
3. Jika *Q* tidak kosong, pilih dari antrian *Q* simpul *i* yang mempunyai $\hat{c}(i)$ paling kecil. Jika terdapat beberapa simpul *i* yang memenuhi, pilih satu secara sembarang.
4. Jika simpul *i* adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul *i* bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika *i* tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak *j* dari simpul *i*, hitung $\hat{c}(j)$, dan masukkan semua anak-anak tersebut ke dalam antrian *Q*.
6. Kembali ke langkah 2.

3. Solusi Algoritma

3.1. Algoritma Lempeng(*Brute Force*)

Bila persoalan ini bila diselesaikan dengan algoritma lempeng (*brute force)* maka variasi kombinasi penyelesaian banyak dan berorde faktorial.

Gambar 1.Matriks yang akan diisi bilangan



Mudah dilihat bahwa melalui teknik *brute force*, memerlukan kompleksitas sekitar n! dan karena adanya kesamaan dalam 4 arah, kompleksitas dikalikan dengan ¼.

Jadi, kompleksitas waktunya, $T(n)=0.25*n^2!$, n=2k-1, k=1,2,3,.... Contoh kasus matriks berukuran 3 x 3 maka kompleksitasnya sebesar
$¼*3^2!$ = 90720 satuan.

3.2. Algoritma *Branch and Bound*

3.2.1. Prinsip Dasar

Sebelum menentukan semua bilangan-bilangan kombinasi yang cocok untuk dimasukkan kedalam matriks, kita patut menentukan apakah bentuk tujuan dapat dicapai dari status awal. Penentuan bahwa pembentukan sudah tidak bisa dilanjutkan lagi dilakukan melalui matriks tereduksi, yakni dengan ketentuan bila dalam satu baris, satu kolom, atau satu diagonal tidak memungkinkan menggunakan bilangan yang baru ditentukan sehingga pengurangan bilangan jumlah tertentu matriks n x n dua bilangan yang berada dalam satu baris, kolom, atau diagonal yang sama tidak berada dalam *range* $0<\text{bilangan}<n^2+1$ atau jumlah dua bilangan tersebut melebihi jumlah konstanta matriks. Pereduksian juga dilakukan apabila jumlah bilangan-bilangan matriks dalam satu baris, kolom, dan diagonal tidak sama dengan jumlah konstanta matriks.

Dapat dibuktikan bahwa jumlah konstanta matriks n x n, $S=½(1+n^2)n$ dengan prinsip induksi. Juga dapat dibuktikan bahwa $x= ½(1+n^2)$ (bilangan tengah) harus selalu berada pada posisi matriks dengan $i,j=½ (n+1), ½ (n+1)$. Hal ini dikarenakan kombinasi angka-angka yang jumlahnya harus memenuhi suatu angka tertentu, maka angka yang paling sering muncul dalam kombinasi tersebut merupakan pembulatan dari hasil bagi jumlah yang harus dipenuhi dengan jumlah angka dalam kombinasi tersebut. Dalam kasus ini, jumlah yang harus dipenuhi adalah $S=½(1+n^2)n$, sedangkan jumlah angka dalam kombinasi tersebut adalah n, maka angka yang paling sering muncul dalam kombinasi tersebut adalah

$$S/n = (½(1+n^2)n) / n = ½(1+n^2)$$

Angka tersebut adalah sama dengan x, yaitu bilangan tengah, maka dari itu x harus berada pada posisi tengah matriks.

Dengan didapatkannya bahwa jumlah yang harus dipenuhi adalah $S=½(1+n^2)n$, maka ukuran matriks yakni n harus ganjil dapat dibuktikan sebagai berikut :

$S = \Sigma a + x$,

S = jumlah yang harus dipenuhi yakni $½(1+n^2)n$
Σa = jumlah angka dalam baris, kolom, atau diagonal yang sama selain nilai tengah.
x = nilai tengah, yakni $½(1+n^2)$

$$\tfrac{1}{2}(1+n^2)n = \Sigma\, a + \tfrac{1}{2}(1+n^2)$$
$$\Sigma\, a = \tfrac{1}{2}(1+n^2)n - \tfrac{1}{2}(1+n^2)$$
$$\Sigma\, a = \tfrac{1}{2}(1+n^2)(n-1)$$
$$2\,\Sigma\, a = (1+n^2)(n-1)$$

Dari persamaan diatas dapat disimpulkan bahwa :

1. kedua ruas harus bernilai sama, yakni bernilai genap. Disebabkan pada ruas kiri terjadi perkalian dengan 2.
2. karena ruas kanan harus bernilai genap, maka nilai $(1+n^2)(n-1)$ harus bernilai genap. Karena merupakan perkalian dua buah konstanta, maka kemungkinan menghasilkan genap adalah
   genap x genap, atau
   genap x ganjil, atau
   ganjil x genap
3. Kita tinjau kasus perkasus
   a. Untuk genap x genap :
      $(1+n^2)$ genap, berarti n haruslah ganjil. (1)
      $(n-1)$ genap, berarti n haruslah ganjil. (2)
      karena (1) dan (2) cocok, maka kasus ini memenuhi dimana n ganjil.
   b. Untuk genap x ganjil :
      $(1+n^2)$ genap, berarti n haruslah ganjil. (1)
      $(n-1)$ ganjil, berarti n haruslah genap. (2)
      karena (1) dan (2) tidak cocok, maka kasus ini tidak memenuhi dimana n bisa ganjil atau genap.
   c. Untuk kasus ganjil x genap :
      $(1+n^2)$ ganjil, berarti n bisa ganjil atau genap. (1)
      $(n-1)$ genap, berarti n haruslah ganjil. (2)
      karena (1) dan (2) cocok, maka kasus ini memenuhi dimana n ganjil.
4. Karena n bernilai ganjil dan n merupakan ukuran matriks, maka terbukti bahwa matriks yang memenuhi sifat ini haruslah berukuran ganjil.

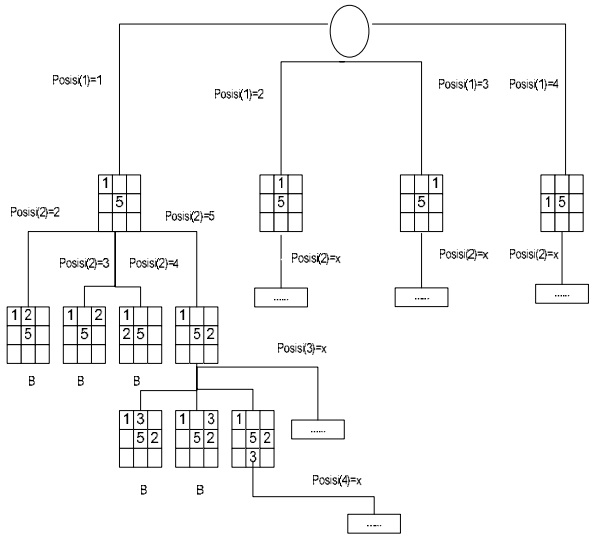
3.2.2 Algoritma

Misalkan posisi(i)=j menandakan pembangkitan bilangan i diposisi j pada matriks secara singuler. Lebih jelasnya, tinjau gambar 2, pada awal penempatan posisi 1, bilangan 1 di bangkitkan pada posisi matriks 0,0 lalu pada posisi 0,1 kemudian 0,2 dan terakhir 1,0. Secara berurutan posisi(1)=1, posisi(1)=2, posisi(1)=3, posisi(1)=4 (posisi lainnya tidak diperhitungkan karena merupakan sekawan dari posisi yang disebutkan).

Pertama kali membangun matriks selalu menempatkan bilangan tengah pada tengah matriks. Bilangan tengah adalah bilangan senilai dengan $\tfrac{1}{2}*(1+n^2)$ dan tengah matriks didefinisikan $i_,j=\tfrac{1}{2}(n+1)$, $\tfrac{1}{2}(n+1)$ -- $i_,j$ posisi bilangan pada baris ke-i dan kolom ke-j.

Statick (*static*) i ditambah satu dan membangkitkan posisi(i) pada posisi yang belum ditempati. Jika matriksnya adalah matriks tereduksi, seperti yang telah dijelaskan sebelumnya, matriks anaknya tidak dibangkitkan (dituliskan pada gambar 2 dengan B *notation*), kemudian proses dilanjutkan ke matriks berikutnya.

Contoh pohon pembentukan status matriks 3x3:
Gambar 2. Pembentukan matriks yang akan diisi bilangan



Berikut contoh penyelesaian kasus untuk matriks berukuran 3 x 3 dengan teknik *branch and bound.* Dengan cara yang sama dapat juga dilakukan untuk matriks sisi ganjil lainnya.

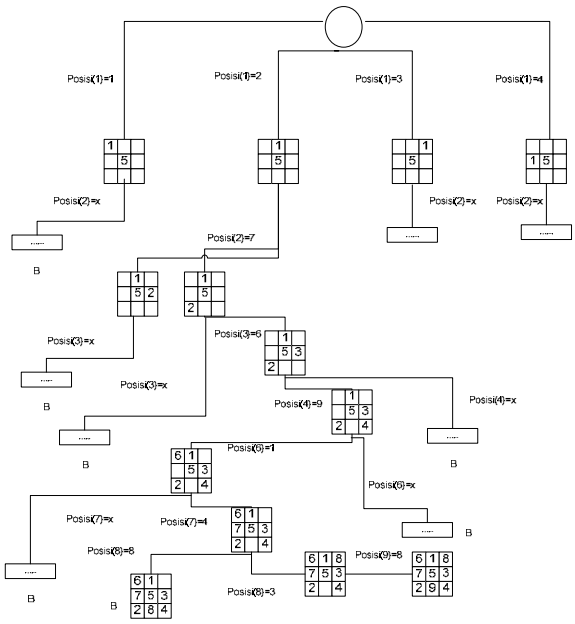
Awal program adalah membangkitkan empat matriks bernilai tengah $\tfrac{1}{2}(n^2+1)=5$ yang diposisikan ditengah matriks di kolom ke-dua dan baris ke-dua. Angka satu diposisikan di posisi pertama,kedua, ketiga, keempat (telah dijelaskan mengenai pengertian posisi(x)=y). Lalu *static* i di-*increment*, i sekarang bernilai dua. Angka dua ditempatkan di posisi dua, tiga. Kemuanya diberi B *notation* karena pengurangan bilangan jumlah matiks 3 x 3 (15) dengan penjumlahan satu dengan dua (dijumlahkan karena sama sama dalam satu baris, satu kolom) sama dengan tiga belas. Padahal, bilangan harus

berada dalam *range* $\leq n^2$ dan $n^2 \geq 13$, n=3. Hal yang sama terjadi pada anakan matriks dua pada posisi enam, muaranya mengharuskan menuliskan B *notation*.

Solusi mulai mengarah pada anakan-anakan matriks dalam penempatan posisi satu, posisi(1)=2.

Setelahnya gunakan alur yang sama untuk memangkas anakan. Gambar 3 menggambarkan skema menuju penemuan solusi

Gambar 3. Pembentukan matriks yang akan diisi bilangan menuju solusi



4. Kesimpulan

Banyak sekali persoalan matematika yang dapat diselesaikan dengan menggunakan matriks. Dalam matriks itu sendiri terdapat *math magic* dimana jumlah angka-angka yang terdapat pada baris, kolom atau diagonal utama yang sama berjumlah sama yaitu $\frac{1}{2}(1+n^2)n$, dengan n adalah ukuran matriks dan n haruslah bernilai ganjil.

Persoalan ini dapat diselesaikan dengan banyak cara. Salah satunya dengan mengimplementasikan strategi algoritmik, tepatnya algoritma *Branch & Bound*.

Dalam permasalahan ini, algoritma B&B dipakai untuk menentukan cost dari tiap simpul dengan menimbang apakah pada garis, kolom, atau diagonal utama yang sudah tebentuk apakah jumlahnya memenuhi yang sudah ditetapkan sebelumnya atau tidak. Jika tidak, maka anaknya tidak akan dibangkitkan alias di-*kill*.

Penerapan algoritma B&B ini lebih efektif dan kompleksitasnya lebih rendah dibandingkan dengan algoritma *Brute Force* yang mencoba semua kemungkinan.

5. Daftar Pustaka

1. **http://64.233.187.104/search?q=cache:JNRuZChXE64J:www.hpc2n.umu.se/para06/papers/paper_2.pdf+branch+and+bound+extended&hl=en&ct=clnk&cd=5**
2. Munir, Rinaldi, *Strategi Algoritmik,* Bandung, 2006.