Penerapan Algoritma Backtracking dalam program N-Queens Problem

I Kadek Meganjaya (10112203)
Faddikha Irza Putra (10112189)

Program Studi Teknik Informatika
Universitas Komputer Indonesia

Abstract - Runut balik (*Backtracking*) merupakan algoritma yang berbasis pada DFS (*Depth First Search*) untuk mencari solusi dari suatu persoalan secara optimal. Runut balik merupakan perbaikan dari algoritma *Brute-force*, secara sistematis mencari solusi dari suatu persoalan diantara semua kemungkinan solusi yang ada. Di zaman sekarang, algoritma *Backtracking* banyak digunakan dalam program-program kecerdasan buatan, salah satu contohnya adalah pada program N-Queens problem. N-queens problem adalah penempatan N buah ratu pada catur ukuran NxN yang nantinya diantara ratu-ratu tersebut tidak boleh berada dalam satu baris, satu kolom dan tidak saling memakan.

1.Pendahuluan

Dewasa ini, teknologi informasi adalah salah satu cabang yang ruang pengembangannya sangat luas, salah satunya adalah dibidang pemrograman. Sebagai salah satu bagian dari teknologi informasi, tentunya bidang pemrograman banyak mengalami perkembangan.

Semakin banyak fasilitas yang ditawarkan oleh aplikasi-aplikasi komputer yang ada saat ini, maka semakin banyak pekerjaan manusia yang di permudah oleh adanya aplikasi tersebut.

Lahirnya *Artificial Intelligence* (A.I.) atau kecerdasan buatan dapat membantu suatu program untuk bekerja dengan pemikiran program itu sendiri, sehingga meminimalisir campur tangan pengguna. Kecerdasan buatan saat ini umum digunakan dalam permainan-permainan komputer yang memungkinkan *mode player* atau *versus computer*, seperti permainan catur yang dapat berjalan dengan sendirinya. Disini akan dijelaskan penerapan algoritma *Backtracking* pada program N-Queens problem.

2.Definisi

Sebelum membahas lebih jauh mengenai N-Queen Problem, disini akan dijelaskan terlebih dahulu tentang algoritma *Backtracking* yang akan digunakan untuk meyelesaikan permasalahan ini.

Backtracking

Algoritma runut balik (*Backtracking*) pertama kali diperkenalkan oleh D.H Lehmer pada tahun 1950. Algoritma ini cukup mangkus untuk digunakan dalam penyelesaian beberapa masalah dan juga untuk memberikan kecerdasan buatan dalam *game*. Beberapa *game* populer semisal *Sudoku*, *Labyrinth* dan *Chess* juga bisa diimplementasikan dengan menggunakan algoritma runut balik.

Algoritma runut balik (*backtracking*) merupakan algoritma yang digunakan untuk mencari solusi persoalan secara lebih mangkus daripada menggunakan algoritma *Brute-force.* Algoritma ini akan mencari solusi berdasarkan ruang solusi yang ada secara sistematis, namun tidak semua ruang solusi akan diperiksa, hanya pencarian yang mengarah kepada solusi yang akan diproses.

Kelemahan dari algoritma *Backtracking* ini adalah algoritma ini hanya bisa diaplikasikan terbatas pada tipe permasalahan yang memiliki solusi yang dapat dicari secara sistematis dan bertahap. Ada beberapa masalah yang tidak bisa diselesaikan dengan menggunkan algoritma *Backtracking*, misalnya menemukan suatu nilai yang diminta pada tabel yang tidak terurut. Namun, ketika algoritma ini dapat diaplikasikan, *Backtracking* dapat bekerja jauh lebih cepat dari algoritma *Brute-force* karena jumlah kandidat solusi yang dapat dibuang dengan algoritma *Backtracking* cukup besar.

Algoritma runut balik berbasis pada DFS (*Depth First Search*), sehingga aturan pencariannya akan mengikut kepada aturan pencarian DFS yaitu dengan mencari solusi dari akar ke daun (dalam pohon ruang solusi) dengan pencarian mendalam. Simpul-simpul yang sudah dilahirkan (diperiksa) dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E atau *Expand Node.*

Algoritma *Backtracking* mempunyai prinsip dasar yang sama seperti algoritma *Brute-force*, yaitu mencoba segala kemungkinan solusi. Perbedaan utamanya terletak pada ide dasarnya, semua solusi dibuat dalam bentuk pohon solusi (pohon ini tentunya berbentuk abstrak) dan algoritma akan menelusuri pohon tersebut secara *DFS* (*Depth Field Search)* sampai ditemukan solusi yang layak.

Properti Umum Metode Backtracking :

a.Solusi Persoalan

Solusi dinyatakan sebagai vektor dengan *n-tuple*:

$X = (x1, x2, \ldots, xn)$, $xi \in Si$.

Mungkin saja $S1 = S2 = \ldots = Sn$.

Contoh: $Si = \{0, 1\}$, $xi = 0$ atau 1

b. Fungsi pembangkit nilai *xk*

Dinyatakan sebagai: $T(k)$

$T(k)$ membangkitkan nilai untuk *xk*, yang merupakan komponen vektor solusi.

c. Fungsi pembatas (pada beberapa persoalan fungsi ini dinamakan fungsi kriteria).

Dinyatakan sebagai$B(x1, x2, \ldots, xk)$.

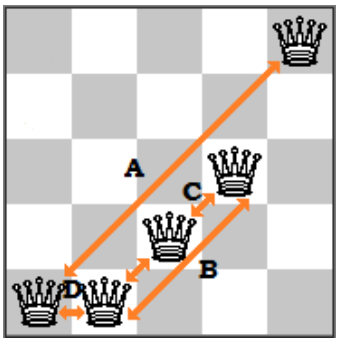
B bernilai *true* jika $(x1, x2, \ldots, xk)$ mengarah ke solusi. Jika *true*, maka pembangkitan nilai untuk $xk+1$dilanjutkan, tetapi jika *false*, maka $(x1, x2, \ldots, xk)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

3. N-QUEENS PROBLEM

3.1. Definisi

N-Queens merupakan salah satu bentuk permainan *puzzle* yang pertama kali diperkenalkan pada tahun 1848 oleh seorang pemain catur Max Bezzel. Dari tahun ke tahun, banyak matematikawan termasuk Gauss dan George Cantor telah bekerja keras untuk dapat menyelesaikan masalah N-Queen ini. Solusi pertama kali dibentuk oleh Franz Nauck pada tahun 1850. Nauck juga memperluas *puzzle* ke bentuk N-Queens. Pada tahun 1874, S. Gunter mengusulkan suatu metode dengan menggunakan metode determinan dan J.W.L. Glaisher menyaring pendekatan tersebut.

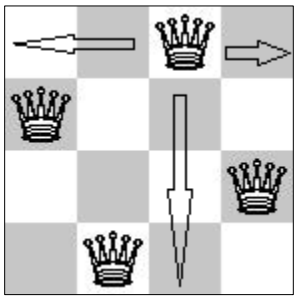
Cara kerja dari N-Queens problem ini adalah bagaimana kita menempatkan n buah ratu pada papan catur NxN, dimana setiap ratu tersebut tidak saling memakan, serta tidak ada 2 ratu yang terletak dalam satu baris, satu kolom, maupun satu diagonal.



Gambar 1

Pada **Gambar 1** di atas, terdapat 5 buah ratu yang terletak pada papan berukuran 5x5. Dan kita dapat melihat dari kelima ratu tersebut, pada posisi A terdapat dua buah ratu dalam satu diagonal, pada posisi B terdapat tiga buah ratu dalam satu diagonal, pada posisi C ratu saling memakan dan berada dalam satu diagonal, dan pada posisi D terdapat dua buah ratu dalam satu baris. Maksud dari pencarian solusi N-Queens

problem ini adalah mencari solusi penempatan N ratu dimana tidak ada ratu yang terletak seperti pada **gambar 1** diatas.



Gambar 2

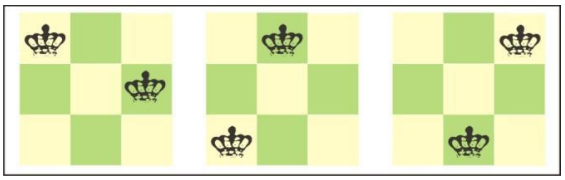
Pada **Gambar 2** terlihat contoh solusi dari penempatan empat buah ratu pada papan berukuran 4x4. Pada gambar tersebut, kita tidak menemukan dua buah ratu yang terletak dalam satu baris, kolom, maupun diagonal atau saling memakan. Dalam program N-Queens ini, kita akan mencari solusi bagaimana agar ratu bisa terlihat seperti **gambar 2** dan mencari berapa solusi yang didapatkan dari N ratu yang diinputkan, serta menampilkan pergerakan dari setiap satu solusi.

3.2. Contoh Kasus Pencarian Untuk N<5

Dalam contoh kasus ini, kita akan coba mencari solusi secara manual dari ratu yang berjumlah satu (N=1) hingga ratu yang berjumlah empat (N=4).

Untuk kasus N=1, disini sudah jelas kita ketahui bahwa ratu berjumlah satu dan solusinya pun pasti satu karena ratu tidak bisa pindah kemana-mana. Untuk N=2, dapat dipastikan tidak adanya solusi karena dalam penempatannya pasti kedua ratu membentuk diagonal atau keduanya terletak dalam satu baris atau dalam satu kolom dan yang pasti, keduanya saling memakan.

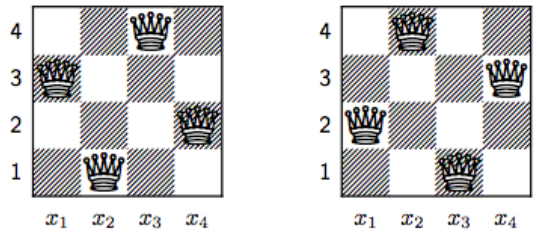
Berdasarkan konsep kombinatorial, untuk penempatan 3 bidak ratu dalam suatu papan catur berukuran 3x3 terdapat 504 kemungkinan solusi "$504 = \frac{(3*3)!}{((3*3)-3)!}$" . Jumlah kemungkinan ini dapat dikurangi apabila kita tambahkan syarat bahwa ketiga bidak tersebut tidak dapat berada dalam suatu kolom yang sama. Dengan mengaplikasikan syarat tersebut, maka jumlah kemungkinan solusi menjadi 3^3 = 27 kemungkinan. Namun demikian, dari 27 kemungkinan yang ada, tidak satupun yang mengarah pada solusi akhir. Buktinya adalah sebagai berikut:



Gambar 3

Pada **Gambar 3,** kita tidak akan menemukan solusi. Pada saat dimasukkan dua ratu dengan tiga kombinasi yang berbeda, ratu yang ke tiga tidak bisa masuk dikarenakan salah satu ratu pasti akan saling memakan.

Sedangkan untuk N=4, kita akan mendapatkan dua buah solusi.



Gambar 4

Pada **gambar 4** di atas, kita menemukan dua buah solusi untuk N=4. Dimana, kita bisa melihat dari kedua buah solusi tersebut sudah memenuhi syarat untuk N-Queens problem. Dan untuk jumlah ratu yang lebih dari empat (N>4) pasti akan memiliki solusi.

Proses penempatan ratu dimulai dari baris pertama kolom $x_1$4, dan akan di lanjutkan pada baris selanjutnya. Pada baris kedua, secara otomatis kolom $x_1$3 dan $x_2$3 akan dilewati dan langsung menempati kolom $x_3$3. Ratu langsung melewati dua kolom tersebut karena jika ditempatkan pada dua kolom sebelumnya, maka ratu akan saling memakan dan berada pada kolom dan diagonal yang sama.

Dan untuk penempatan pada baris selanjutnya, sama seperti cara dibaris kedua. Jika hingga baris terakhir masih belum mendapatkan solusi, penempatan ratu akan terus diulang hingga baris yang paling atas tidak bisa berpindah kemana-mana.

3.3 Algoritma N-Queen Problem

```
Procedure cetak_solusi(input x : Array,
input n_ratu : integer)
Kamus :
    i,j : integer
    c  : array[0..50] of char
Algoritma:
    for i←1 to n_ratu do
        for j←1 to n_ratu do
                c[i,j] ←'karakter papan catur'
        endfor
    endfor
    for i←1 to n_ratu do
                c[i,x[j]] ←'karakter untuk ratu'
    endfor
    for i←1 to n_ratu do
        for j←1 to n_ratu do
                output(c[i,j])
        endfor
    endfor
endprocedure
```

```
    for i←1 to n_ratu do
        for j←1 to n_ratu do
                c[i,j] ←'karakter papan catur'
        endfor
        endfor
```

Penjelasan dari algoritma di atas, yaitu menggunaan array dua dimensi. Pada perulangan yang pertama "for i←1 to n_ratu do" perulangan dilakukan untuk baris papan catur,

sedangkan perulangan for j←1 to n_ratu do yaitu melakukan perulangan untuk mengisi kolom pada papan catur. c[i,j] ←'karakter papan catur' kita mengisi nilai dari c[i,j] dengan karakter yang kita inginkan untuk simbol setiap kolom papan catur.

Untuk penjelasan pada baris bawahnya hampir sama penjelasannya seperti penjelasan diatas.

```
Function tempat(input x : Array, input k : integer)
Kamus:
    i : integer
    kedudukan,stop : Boolean
Algoritma:
    kedudukan←true
    i←1
    stop←false
    while (i<k) and (not stop) do
        if x[i] = x[k] or abs(x[i])=abs(i-k) then
                kedudukan←false
                stop←true
        else
                i←i+1
        endif
    endwhile
    return kedudukan
endfunction
```

Fungsi diatas berfungsi untuk memberikan nilai true atau false bila ada dua ratu yang posisinya berada dalam satu baris, kolom ataupun dalam satu diagonal.

```
Procedure nRatuI(input n_ratu : integer)
Kamus:
    x : Array[0..50] of integer
    k : integer
Algoritma:
    k←1
    x[k] ←0
    while (k>0) do
        x[k] ← x[k]+1
        while (x[k] ≤ n_ratu) and (not tempat(x,k)) do
                x[k] ←x[k]+1
        endwhile
        if x[k] ≤ n_ratu then
                if k=n_ratu then
                        cetak_solusi(x,n_ratu)
                else
                        k←k+1
                        x[k] ←0
                endif
        else
                k←k-1
        endif
    endwhile
endprocedure
```

Procedure di atas berfungsi untuk memindahkan ratu dari satu kolom ke kolom berikutnya dan selanjutnya akan diperiksa lagi apakah ratu bisa ditempatkan pada kolom yang ditempati sekarang.

4. Hasil Eksekusi Program

Setelah program sudah selesai dibuat, kami membandingkan dua buah laptop dengan spesifikasi yang berbeda untuk menjalankan program tersebut. Tujuannya yaitu untuk mengetahui seberapa jauh perbedaan waktu yang di perlukan untuk mendapatkan total solusi dari program tersebut.

Tabel 1, laptop acer (Intel Pentium 2,1Ghz dengan RAM 2 Gb)

N	Jumlah Solusi	Waktu Eksekusi
5	10 solusi	0.2490 detik
6	4 solusi	0.1210 detik
7	40 solusi	1.4140 detik
8	92 solusi	3.3480 detik
9	352 solusi	14.3350 detik
10	724 solusi	35.0560 detik

Tabel 2, laptop toshiba (Intel Core i5 2.4GHz dengan RAM 4 Gb)

N	Jumlah Solusi	Waktu Eksekusi
5	10 solusi	0.2250 detik
6	4 solusi	0.0930 detik
7	40 solusi	1.0090 detik
8	92 solusi	2.5650 detik
9	352 solusi	11.6130 detik
10	724 solusi	29.0330 detik

5. Kesimpulan

Program N-Queens Problem termasuk ke dalam *Artificial Intelligence* (A.I.) yang memiliki kemampuan untuk menentukan jalan keluar dari suatu permasalahan. Dalam program ini, penggunaan algoritma *Backtracking* sangat tepat daripada menggunakan algoritma *Brute-force*. Dalam program ini, kita juga dapat melihat kecepatan eksekusi dari program dalam mencari solusi dengan spesifikasi laptop yang berbeda.

Daftar Pustaka

[1] Munir, Rinaldi, *Diktat Kuliah IF3051 Strategi Algoritma* Prodi Teknik Informatika Institut Teknologi Bandung, 2009.

[2]http://www.scribd.com/doc/93612673/Algoritma-Backtracking

[3]http://widiantogilangramadhan.wordpress.com/2010/ 12/23/makalah -backtracking/

[4]http://www.ittelkom.ac.id/staf/zka/Materi%20Desain%20Analisis%20Algoritma/M14Algoritma%20Runut-balik.pdf

[5]http://repository.gunadarma.ac.id/bitstream/123456789/2747/1/21-PENYELESAIAN%20MASALAH%20NQUEEN%20DENGAN%20TEKNIK%20BACKTRACKING.pdf

[6] http://letstalkdata.com/2013/12/n-queens-part-1-steepest-hill-climbing/