ALGORITMA RUNUT-BALIK UNTUK MENGGANTIKAN ALGORITMA *BRUTE FORCE* DALAM PERSOALAN N-RATU

Nur Cahya Pribadi-NIM: 13505062

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail: if15062@students.if.itb.ac.id

ABSTRAK

Dalam kehidupan seringkali menemui banyak masalah. Kadang-kadang masalah tersebut kita selesaikan dengan cara yang paling cepat ,tetapi belum tentu tepat. Hal ini juga dapat terjadi di dunia komputer yaitu pemograman. Seringkali untuk menyelesaikan suatu masalah pemograman dengan cepat digunakan algoritma *Brute Force* , tetapi sayangnya algoritma ini kurang begitu mangkus. Dalam aplikasi yang kecil mungkin algoritma ini mungkin bisa digunakan , tetapi dalam aplikasi yang besar algoritma ini tidak tepat digunakan karena dapat membuat kompleksitas waktu dan kompleksitas ruang aplikasi menjadi besar. Oleh karena itu untuk menggantikan algoritma Brute Force dalam beberapa masalah digunakan algoritma runut-balik atau sering disebut *backtracking*.

Algoritma *Brute Force* adalah sebuah pendekatan yang lempang untuk memecahkan suatu masalah, biasanya didasarkan langsung pada pernyataan masalah dan definisi konsep yang dilibatkan. Algoritma runut-balik adalah algoritma yang berbasis DFS untuk mencari solusui persoalan yang lebih mangkus. Kompleksitas waktu adalah jumlah operasi yang dilakukan untuk menjalankan sebuah algoritma sebagai fungsi dari ukuran masukan n. Sedangkan kompleksitas ruang adalah ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n.

Kata kunci: Algoritma *Brute Force*, Algoritma *runut-balik*, Kompleksitas waktu.

1. PENDAHULUAN

Ada banyak algoritma alternatif untuk menggantikan algoritma *brute force* yaitu algoritma *greedy* dalam pemecahan persoalan penukaran uang, algoritma *Divide and Conquer* dalam persoalan pencarian minimum dan maksimum, dan algoritma Knuth-Morris-Pratt pada persoalan pencarian string. Tapi dalam persoalan N-Ratu algoritma yang cocok untuk menggantikan algoritma *Brute Force* adalah algoritma runut-balik .

Tentu saja dengan menggunakan algoritma runut-balik sebagai alternatif pengganti algoritma *Brute Force* pada persoalan N-Ratu membuat kompleksitas waktu menjadi semakin kecil.

2. METODE

Dalam persoalan N-Ratu ini digunakan algoritma runut-balik untuk menggantikan algoritma *Brute Force*. Istilah runut-balik pertama kali diperkenalkan onleh D.H Lehmer pada tahun 1950. Uraian umum dan penerapannya pada berbagai persoalan selanjutnya disajikan oleh R.J. Walker, Golomb, dan Baumert. Algoritma runut-balik merupakan perbaikan dari algoritma *Brute Force,* secara sistematis mencari solusi persoalan diantara semua kemungkinan solusi yang ada. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya kompleksitas wak tu menjadi semakin kecil.

2.1 Properti Umum

Untuk menerapkan metode runut-balik, properti berikut didefinisikan:

1. Solusi persoalan.
   Solusi dinyatakan sebagai vektor *n-tuple*:
   X=(x1, x2, x3,....,xn) x1€ himpunan berhingga S1. Mungkin saja S1=S2=...=Sn.
   Contoh Si={0,1}
   Xi= 0 atau 1
2. Fungsi Pembangkit nilai xk
   Dinyatakan sebagai:
   T(k)
   T(k) membangkitkan nilai untuk xk, yang merupakan komponen vektor solusi.

3. Fungsi kriteria
   Dinyatakan sebagai B(x1, x2, ...,xk)

2.2 Prinsip Pencarian Solusi

Pencarian solusi ditinjau pada pohon ruang status yang dibangun secara dinamis. Langkah-langkah pencarian solusi adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti metode pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup. Simpul hidup yang sedang diperluas dinamakan simpul-E. Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya.

2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi simpul mati. Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas. Simpul yang sudah mati tidak akan pernah diperluas lagi.

3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut-balik ke simpul hidup terdekat. Selanjutnya simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.

4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik.

2.3 Skema Umum

Dibawah ini disajikan skema umum algoritma runut-balik dalam dua versi yaitu versi rekursif dan versi iteratif.

1.Versi Rekursif

```
Procedure RunutBalikR(input k:integer)

Algoritma:
    For tiap x[k] yang belum dicoba sedemikian sehingga
      ( x[k]←T[k]) and B(x[1], x[2],...,x[k])=true do
    If (x[1], x[2],..., x[k]) adalah lintasan dari akar ke
    daun then
      CetakSolusi(x)
    endif
    RunutBalikR(k+1)
endfor
```

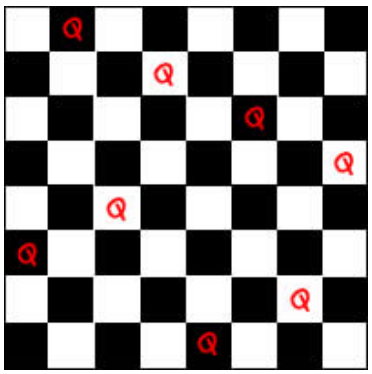
2. Versi iteratif

```
Procedure RunutBalikI(input n:integer)

Deklarasi:
    K:integer;

Algoritma:
  k←1
  while k>0 do
  if(x[k] belum dicoba sedemikian sehingga x[k]←T[k])
and (B(x[1], x[2]..., x[k])=true)then
    If (x[1], x[2],..., x[k]) adalah lintasan dari akar ke
    daun then
      CetakSolusi(x)
    endif
    k←k+1
  else
    k←k+1
  endif
  endwhile
```

3. PERSOALAN N-RATU

Diberikan sebuah papan catur berukuran NxN dan delapan buah ratu. Bagaimanakah menempatkan N buah ratu itu terletak pada papan catur sedemikian sehingga tidak ada dua ratu atau lebih yang terletak pada satu baris yang sama, atau pada satu kolom yang sama, atau pada suatu diagonal yang sama ?

Contoh solusi persoalan 8-ratu ditunjukan oleh gambar berikut:



Gambar 1. Salah satu contoh solusi dari persoalan n-ratu.

3.1 Penyelesaian dengan Algoritma *Brute Force*

Masalah persoalan ini dapat diselesaikan dengan algoritma *Brute Force* :

a) *Brute Force* 1

Mencoba semua kemungkinan solusi penempatan 8 buah ratu pada petak-petak catur. Jika dimisalkan solusinya dijadikan pohon ruang maka algoritma ini seperti pohon ruang status statis. Taruh 8 buah ratu pada setiap petak sembarang, dan diperiksa apakah posisi semua ratu merupakan sebuah solusi. Ketidakmangkusan pada metode ini pada pencarian yang sudah tidak mungkin mendapatkan solusi lagi masih diteruskan. Banyaknya kemungkinan solusi adalah

C(64, 8)= 4.426.165.368

64 merupakan banyaknya petak pada papan catur dan 8 merupakan 8 buah ratu.

b) *Brute Force* 2

Pada metode ini lebih maju satu langkah dari metode *Brute Force* sebelumnya yaitu meletakkan satu ratu pada setiap baris yang berbeda. Kemudian diperiksa posisi ratu-ratu tersebut merupakan sebuah solusi.

```
procedure Ratu1

Deklarasi:
   i1, i2, i3, i4, i5 ,i6 ,i7 ,i8 : integer;

Algoritma:
  for i1←1 to 8 do
    for i2←1 to 8 do
      for i3←1 to 8 do
        for i4←1 to 8 do
          for i5←1 to 8 do
            for i6←1 to 8 do
              for i7←1 to 8 do
                for i8←1 to 8 do
                  if Solusi(i1, i2, i3, i4, i5 ,i6 ,i7 ,i8) then
                      CetakSolusi();
                endif
              endfor
            endfor
          endfor
        endfor
      endfor
    endfor
  endfor
endfor
```

Dengan algoritma Brute Force ini, jumlah kemungkinan solusi yang diperiksa semakin berkurang menjadi

8x8x8x8x8x8x8x8=16.777.216

Kelemahan pada algoritma ini adalah tidak memperhatikan penempatan n-ratu pada kolom yang sama dan diagonal yang sama serta terus melanjutkan penempatan n-ratu padahal solusi sudah jelas tidak mungkin ditemukan.

c) *Brute Force* 3

Metode *Brute Force* kali ini hampir sama dengan metode *Brute Force* sebelumnya hanya pada metode *Brute Force* kali ini kita menempatkan ratu pada baris dan kolom yang berbeda. Misalkan solusinya dinyatakan delam vektor 8-*tupple* :

X=(x1, x2,..., x8)

Dalam hal ini x menyatakan kolom kedudukan ratu pada baris ke-i. Dengan mengamati bahwa vektor solusi merupakan permutasi dari bilangan 1 sampai 8. maka banyak kemungkinan solusi adalah P(8,8)=8!=40.230.

```
procedure Ratu1

Deklarasi:
  X:vektor solusi
  n, i: integer

Algoritma:
  n←40320
  i←1
  repeat
    X←Permutasi(8)
    if Solusi(X) then
      CetakSolusi(X);
    endif
    i←i+1
  until i> n
```

Kelemahan pada Brute Force ini adalah tidak memperhatikan penempatan ratu pada posisi diagonal dan masih meneruskan penempatan ratu ketika solusi tidak mungkin ditemukan.

3.2 Penyelesaian dengan Algoritma Runut-Balik

Dalam algoritma runut-balik ini hanya memperbaiki algoritma *Brute Force* 3. Dimana algoritma *Brute Force* 3 hanya mempermutasi n-ratu pada baris dan kolom berbeda dan juga penempatannya tidak memperhatikan persyaratan diagonal (hanya ada fungsi solusi). Dan pada algoritma *Brute Force* 3 meneruskan penempatan n-ratu ketika solusi sudah jelas tidak ditemukan lagi. Berbeda dengan algoritma runut-balik. Algoritma ini memperhatikan penempatan n-ratu pada posisi diagonal dan tidak meneruskan penempatan n-ratu ketika solusi sudah jelas tidak dapat ditemukan. Karena itu algoritma ini lebih lebih mangkus dan mempunyai kompleksitas waktu yang lebih kecil dari algoritma *Brute Force* pada persoalan n-ratu.

(a) Versi Iteratif

Dua buah ratu terletak pada baris yang sama, berarti
i=k
Dua buah ratu terletak pada kolom yang sama, berarti
j=l
Dua buah ratu terletak pada diagonal yang sama, berarti
untuk diagonal kanan bawah:
i-j=k-l. (1)
untuk diagonal kiri bawah:
i+j=k+l (2)
dari (1) dan (2) kita dapat jadikan
- i-k=j-l atau k-i=j-l
- | j-l |=| i-k |

Algoritma pemecahan persoalan n-ratu dengan versi iteratif

```
procedure N_RATU_I(input N:integer)

Deklarasi:
  k:integer;
  x:array[1..N]of integer

Algoritma:
  k←1
  x[1] ←0
  while k >0 do
   x[k] ←x[k]+1
   while(x[k]<=N)and(not TEMPAT(k))do
    x[k] ←x[k]+1
   endwhile
   if(x[k]<=n)then
    if k=N then
     CetakSolusi(x,N)
    else
     k←k+1
     x[k] ←0
    endif
   else
    k←k+1
   endif
  endwhile
```

Fungsi TEMPAT mengembalikan *true* jika penempatan ratu tepat sesuai persyaratan yaitu ratu ditempatkan pada baris, kolom, dan diagonal yang berbeda.

Pertama-tama ratu ditempatkan pada baris kesatu (k←1) dan pada kolom ke-0(x[1] ←0). terus terjadi *looping* selama solusi belum ditemukan(while k >0 do). Ratu ditempatkan pada kolom selanjutnya(x[k] ←x[k]+1). Ketika ratu tidak bisa ditempatkan pada kolom yang sedang berlangsung(tidak sesuai dengan persyaratan baris, kolom, diagonal) maka ratu akan ditempatkan pada kolom selanjutnya dan ini terjadi *looping* sampai kolom yang ratu tempatkan tidak melanggar persyaratan baris, kolom, dan diagonal. Ketika kolom penempatan ratu ditemukan maka dicek lagi apakah semua ratu pada setiap baris sudah ditempatkan. jika semua ratu sudah ditempatkan maka solusi dicetak. jika tidak maka pergi ke baris berikutnya dan kolom diinisialisasikan menjadi nol kembali. Dan ketika penempatan ratu pada kolom tidak ditemukan maka dilakukan *backtracking* ke baris sebelumnya.

(b) Versi Rekursif

```
procedure N_RATU_R(input k,n:integer)

Deklarasi:
   x:array[1..N]of integer
   stop:boolean

Algoritma:
  stop←false
  while not stop do
    x[k] ←x[k]+1
    while(x[k]<=N)and(not TEMPAT(k))do
      x[k] ←x[k]+1
    endwhile
    if x[k]<=N then
      if k=N then
        CetakSolusi(x,N)
       else
         N_RATU_R(k+1,N)
    else
     stop←true
```

```
      x[k] ←0
    endif
  endwhile
```

Pada skema rekursif ini tidak banyak berubah dari skema iteratif perbedaannya hanya ada pada penempatan ratu pada baris berikutnya dilakukan dengan cara rekursif.

IV. KESIMPULAN

Kesimpulan yang dapat diambil adalah:

1. Algoritma *Brute Force* adalah sebuah pendekatan yang lempang untuk memecahkan suatu masalah, biasanya didasarkan langsung pada pernyataan masalah dan definisi konsep yang dilibatkan. Algoritma runut-balik adalah algoritma yang berbasis DFS untuk mencari solusui persoalan yang lebih mangkus. Kompleksitas waktu adalah jumlah operasi yang dilakukan untuk menjalankan sebuah algoritma sebagai fungsi dari ukuran masukan n.

2. Ada beberapa persoalan yang dapat menggantikan algoritma *Brute Force* seperti algoritma *greedy* dalam pemecahan persoalan penukaran uang, algoritma *Divide and Conquer* dalam persoalan pencarian minimum dan maksimum, algoritma Knuth-Morris-Pratt , dan algoritma runut-balik pada persoalan n-ratu.

3. Pada persoalan n-ratu algoritma runut-balik memperhatikan penempatan n-ratu pada posisi diagonal dan tidak meneruskan penempatan n-ratu ketika solusi sudah jelas tidak dapat ditemukan. Sedangkan pada algoritma *Brute Force* 3, 2 hal ini tidak diimplementasikan. Karena itu algoritma runut-balik lebih lebih mangkus dan mempunyai kompleksitas waktu yang lebih kecil dari algoritma *Brute Force*.

REFERENSI

[1] Munir, Rinaldi, “Strategi Algoritmik”, ITB, 2007.