PENERAPAN ALGORITMA *BACKTRACKING* DALAM PENCARIAN SOLUSI MENARA HANOI

Arif Nanda Atmavidya (13506083)

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10, Bandung 40132
e-mail: if16083@students.if.itb.ac.id

ABSTRAK

Menara Hanoi adalah sebuah permainan matematis atau teka-teki yang terdiri dari tiga tiang dan sejumlah cakram dengan ukuran berbeda-beda yang bisa dimasukkan ke tiang mana saja. Permainan dimulai dengan cakram-cakram yang tertumpuk rapi berurutan berdasarkan ukurannya dalam salah satu tiang, cakram terkecil diletakkan teratas, sehingga membentuk kerucut. Dalam permasalahan ini, solusi berusaha dipecahkan dengan algoritma *backtracking*, dimana jika dalam suatu langkah sudah tidak ditemukan jalan lagi, maka proses runut-balik dilakukan. Algoritma ini juga memanfaatkan algoritma *greedy* dalam proses mencari cakram terkecil di setiap tiang.

Kata kunci: Runut-balik, *Backtracking*, Menara Hanoi.

1. PENDAHULUAN

1.1 Algoritma Runut-balik (*Backtracking*)

Runut-balik (*backtracking*) adalah algoritma yang berbasis DFS (pencarian mendalam) untuk mecari solusi persoalan secara lebih mangkus. *Backtracking* yang merupakan perbaikan dari algoritma *bruteforce* secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah pada solusi saja yang dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat. *Backtracking* secara harfiah dinyatakan dalam algoritma rekursif, Kadang-kadang disebutkan pula bahwa runut-balik merupakan bentuk tipikal dari algoritma rekursif. [1]

Saat ini algoritma runut-balik banyak diterapkan untuk program *games* (seperti permainan *tic-tac-toe*, menemukan jalan keluar dalamsebuah labirin, catur, dll.) dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*). [1]

Untuk menerapkan metode runut-balik, properti berikut didefinikan:

1. Solusi persoalan
   Solusi dinyatakan sebagai vector dengan *n-tuple*:
   $X = (x_1, x_2, \ldots, x_n)$, $x_i \in$ himpunan berhingga S_i
   Mungkin saja $S_1 = S_2 = \ldots = S_n$
   Contoh: $S_i = \{0, 1\}$, $x_i = 0$ atau 1
2. Fungsi pembangkit nilai x_k
   Dinyatakan sebagai T(k)
   T(k) membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.
3. Fungsi pembatas (pada beberapa persoalan fungsi ini dinamakan fungsi kriteria)
   Dinyatakan sebagai $B(x_1,x_2,\ldots,x_k)$
   Fungsi pembatas menentukan apakah $(x_1, x_2, \ldots, x_k)$ mengarah ke solusi. Jika mengarah, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika tidak, maka $(x_1, x_2, \ldots, x_k)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

Langkah-langkah pencarian solusi adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar sampai daun. Aturan pembentukan yang diapai 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 ayng 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.

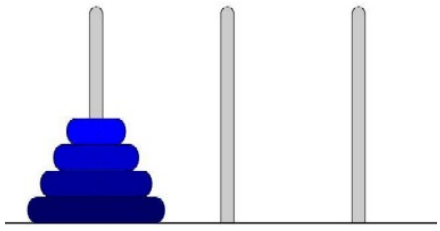
1.2 Menara Hanoi

Menara Hanoi adalah sebuah permainan matematis atau teka-teki. Permainan ini terdiri dari tiga tiang dan sejumlah cakram dengan ukuran berbeda-beda yang bisa dimasukkan ke tiang mana saja. Permainan dimulai dengan cakram-cakram yang tertumpuk rapi berurutan berdasarkan ukurannya dalam salah satu tiang, cakram terkecil diletakkan teratas, sehingga membentuk kerucut.

Gambar 1 Tiga buah contoh kasus menara Hanoi yang belum terselesaikan

Tujuan dari teka-teki ini adalah untuk memindahkan seluruh tumpukan ke tiang yang lain, mengikuti aturan berikut:

- Hanya satu cakram yang boleh dipindahkan dalam satu waktu.
- Setiap perpindahan berupa pengambilan cakram teratas dari satu tiang dan memasukkannya ke tiang lain, di atas cakram lain yang mungkin sudah ada di tiang tersebut.
- Tidak boleh meletakkan cakram di atas cakram lain yang lebih kecil.



Gambar 2 Susunan menara Hanoi dalam keadaan terselesaikan (solusi)

Teka-teki ini ditemukan Édouard Lucas, ahli matematika Perancis di tahun 1883. Ada sebuah legenda tentang candi Indian yang berisi ruang besar dengan tiga tiang yang dikelilingi 64 cakram emas. Pendeta Brahma, melaksanakan tugas dari peramal di masa lalu, sesuai dengan aturan teka-teki ini. Menurut legenda ini, bila teka-teki ini diselesaikan, dunia akan kiamat. Tidak jelas benar apakah Lucas menemukan legenda ini atau terinspirasi olehnya. Bila legenda ini benar, dan pendeta itu bisa memindahkan satu cakram tiap detik, menggunakan pemindahan paling sedikit, maka akan memakan waktu $2^{64}-1$ detik atau kurang lebih 584,582 milyar tahun.

Permainan Menara Hanoi sering digunakan dalam penelitian psikologis dalam hal pemecahan masalah. Selain itu, juga sering digunakan dalam pengajaran algorima rekursif bagi pelajar pemrograman. Permainan ini juga digunakan sebagai ujian ingatan oleh ahli psikolog syaraf dalam berupaya mengevaluasi amnesia.

2. PENGUJIAN

Untuk menggambarkan pergerakan algoritma runut-balik secara terperinci, kita membagi lintasan menjadi sederetan langkah. Setiap langkah terdiri dari variasi perpindahan cakram dari satu tiang ke dua tiang lainnya. Arah perpindahan yang mungkin ialah menuju tiang dengan cakram teratas yang lebih besar daripada cakram yang ingin kita pindahkan.

2.1 Pembuatan Algoritma

Dari hasil, identifikasi masalah dalam menara Hanoi, penulis mengusulkan algoritma sebagai berikut:

```
while bukan solusi do
   if (cakram di bawah cakram terkecil di
   tiang tertinggi merupakan cakram terkecil
   jika dibandingkan dengan cakram-cakram
   tertinggi di tiang lainnya) && (masih ada
   pilihan pergerakan bagi cakram di tiang
   tertinggi lainnya)
   then memroses cakram terkecil kedua dan
      tertinggi di tiang lainnya
   else
     if terdapat cara bagi cakram terkecil
        untuk dipindahkan ke tiang lain
     then pindahkan cakram terkecil ke tiang
        dengan cakram tertinggi lebih besar
        yang paling sedikit selisih besarnya
     else backtrack langkah sampai terdapat
        kesempatan perpindahan cakram seperti
        di atas
     endif
  endif
endwhile
```

Dalam algoritma di atas, tampak adanya proses mencari cakram terkecil di setiap puncak tiang. Proses tersebut dilakukan tiap kali akan memindahkan cakram. Dan untuk mendapatkan hasil yang minimal (cakram terkecil), akan dapat digunakan algoritma *greedy*.

Seperti pada banyak penerapan algoritma *backtracking* lain, seperti pada persoalan labirin, terdapat permasalahan cara penjejakan kembali terhadap langkah terakhir yang dipilih (jika tidak ada solusi lagi). Seperti halnya persoalan labirin, terdapat dua solusi masalah: pertama, menyimpan semua langkah yang dilalui, kedua, menggunakan teknik rekursi (yang secara implisit menyimpan semua langkah). Rekursi adalah solusi yang lebih mudah. [1]

```
function SolveHanoi(input H : MenaraHanoi) →
boolean {mengembalikan true jika solusi
ditemukan, false jika tidak}

Kamus lokal
movement : integer {1 : tiang pertama,
2 : tiang kedua, 3 : tiang ketiga}

Algoritma :
begin
   if solusi ditemukan then
      return true
   else
      for setiap kemungkinan perpindahan
      (1,2,3) do
         Move(H, movement)
         if SolveHanoi(H) then
            return true
         else
            UnMove(M, movement) {melakukan
            proses backtracking}
         endif
      endfor
      return false
   endif
end
```

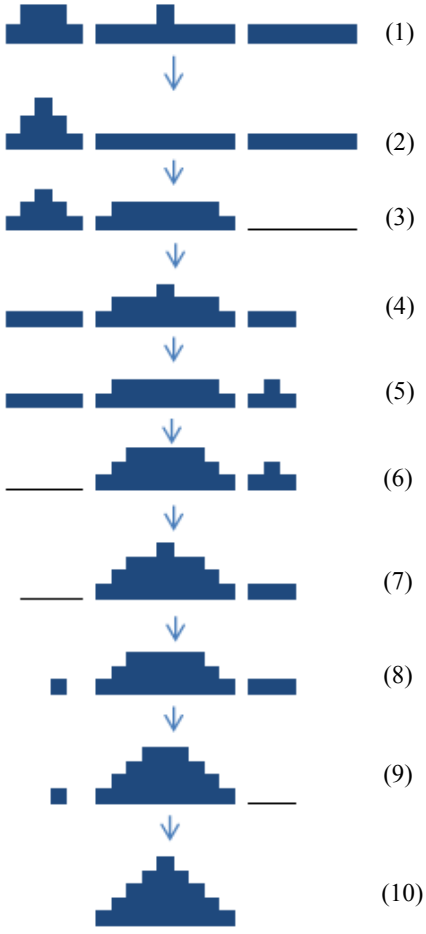
Algoritma di atas hanya memberikan *output* solusi ditemukan atau tidak, tanpa mampu menyimpan keseluruhan proses perpindahan cakram. Oleh karena itu, dibutuhkan suatu tumpukan (*stack*) untuk menyimpan keseluruhan perpindahan cakram yang dilakukan. Sehingga jika terjadi proses *backtracking*, hanya perlu berpindah ke keadaan sebelumnya yang telah tersimpan dalam tumpukan tersebut.

2.2 Pengujian Algoritma

Sebagai analisis sekaligus pembuktian, kita ambil sebuah contoh menara Hanoi yang belum terselesaikan.

Gambar 3 Sebuah contoh kasus Menara Hanoi yang belum terselesaikan

Berikut ini merupakan proses *backtracking* terhadap permasalahan di atas:



Gambar 4 Proses *backtracking* terhadap permasalahan pada gambar 3

3. ANALISIS PEMBAHASAN

Berikut analsis algoritma *backtracking* yang diterapkan pada menara Hanoi dalam contoh gambar 3:

(1) Kasus menara Hanoi yang masih dalam keadaan bukan solusi (masih teracak).
(2) Cakram terkecil yang terletak di tiang ke-2 berpindah ke tiang pertama karena cakram puncak tiang pertama lebih kecil setingkat daripada cakram terkecil tersebut.
(3) Cakram puncak terkecil kedua yang terletak di tiang ke-3 berpindah ke tiang ke-2.
(4) Karena tidak ada jalan lain selain memindahkan cakram puncak tiang pertama, maka cakram puncak tersebut dipindahkan ke tiang yang cakram puncaknya berselisih kecil, yaitu tiang ke-2. Sedangkan cakram di bawah cakram puncak tiang pertama dipindah ke tiang ke-3.
(5) Cakram terkecil yang terletak di puncak tiang ke-2 dipindah ke tiang ke-3 karena tiang tersebut lah yang memiliki cakram puncak berselisih terkecil.
(6) Cakram terkecil pertama di tiang ke-3 tidak mengalami perubahan karena cakram bawahnya memang cakram terkecil ke-2. Sedangkan cakram puncak terkecil ke-2 di tiang ke-1 dipindah ke tiang ke-2.
(7) Karena tidak ada jalan lain, maka cakram puncak di tiang ke-3 dipindah ke tiang ke-2.
(8) Karena tidak ada jalan lain pula, cakram puncak tiang ke-2 dipindah ke tiang pertama (jika dipindah ke tiang ke-3 maka tidak mengubah kondisi nomor 6).
(9) Cakram terkecil kedua di tiang ke-3 dipindah ke tiang ke-2 (jika yang dipindah cakram di tiang pertama, maka tidak mengubah kondisi nomor 7).
(10) Dengan memindah cakram terkecil di tiang pertama ke tiang ke-2 maka diperoleh solusi.

4. KESIMPULAN

Dari hasil pengujian, maka dapat disimpulkan hal-hal sebagai berikut:

1. Algoritma *backtracking* mampu memberikan solusi pada permasalahan menara Hanoi ini.
2. Algoritma *backtracking* memberikan solusi yang lebih mangkus dan efisien daripada solusi yang diberikan oleh algoritma lain, semisal *bruteforce* dan *exhaustic.*
3. Dalam permasalahan ini, sebenarnya terdapat pula algoritma *greedy* dalam kasus mencari cakram terkecil di setiap tiang.

REFERENSI

[1] Munir, Rinaldi. 2005. *Strategi Algoritmik.* Bandung : Penerbit Informatika.
[2] Munir, Rinaldi. 2005. Matematika Diskrit. Bandung : Penerbit Informatika.
[3] *http://id.wikipedia.org/wiki/Menara_Hanoi*. Diakses pada 18 Mei 2008.