Aplikasi Algoritma Runut Balik Dengan *Threshold* untuk Pencarian Jalan di Pusat Perbelanjaan

Dhammavadi Metta

Laboratorium Ilmu dan Rekayasa Komputasi
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if13100@students.if.itb.ac.id

ABSTRAK

Pada zaman sekarang, pusat perbelanjaan dapat dijumpai di mana-mana, mulai dari yang kecil sampai dengan yang sangat luas. Pusat perbelanjaan yang luas dapat menampung jumlah toko dan konsumen yang banyak, akan tetapi konsumen mungkin akan mengalami kesulitan dalam mencari toko yang dituju. Selain itu, bagi konsumen yang sedang terburu-buru misalnya, aplikasi ini akan sangat membantu karena konsumen hanya perlu mengikuti arah yang sudah ditunjukkan dan tidak membuang-buang waktu untuk mencari toko yang dimaksud. Makalah ini akan mencoba membahas tentang aplikasi algoritma runut balik untuk pencarian jalan. Algoritma runut balik adalah suatu algoritma yang berbasis pada algoritma pencarian mendalam (DFS). Dengan menggunakan algoritma ini, dapat dibuat suatu sistem aplikasi untuk pencarian jalan seperti halnya penemuan jalan keluar dalam sebuah labirin. Akan tetapi, pada algoritma runut balik biasa untuk mencari lintasan terpendek (*shortest path*) akan diperlukan waktu yang cukup lama, yaitu dengan membandingkan semua lintasan yang mungkin (pengaplikasian *brute force*). Oleh karena itu, algoritma tersebut sedikit dimodifikasi dengan menambahkan *threshold* sehingga didapatkan jalan yang relatif pendek dalam waktu yang relatif singkat. Algoritma yang telah dimodifikasi ini kemudian akan dibandingkan dengan algoritma runut balik biasa.

Kata kunci: pusat perbelanjaan, *backtracking algorithm*, pencarian jalan.

1. PENDAHULUAN

Pada pusat perbelanjaan seperti *mall*, terdapat berbagai kasus. Salah satu kasusnya yaitu adanya kesulitan pengunjung untuk mencari lokasi suatu tempat atau toko dan cara mencapai tempat atau toko yang ingin dituju. Jelasnya, ketika seseorang masuk ke dalam pusat perbelanjaan, terdapat dua kemungkinan. Kemungkinan pertama, dia tidak tahu ada toko apa saja yang terdapat di pusat perbelanjaan tersebut. Kemungkinan kedua, dia sudah mengetahui toko yang ingin dituju tetapi tidak tahu di mana lokasinya dan bagaimana mencapai toko yang dicari. Kemungkinan-kemungkinan ini membuat diperlukannya sistem yang membantu mempermudah pengunjung pusat perbelanjaan untuk menghemat waktunya dengan menyediakan peta keseluruhan pusat perbelanjaan sekaligus lokasi toko-toko yang ada dan menampilkan rute jalan terdekat menuju lokasi yang ingin dituju.

Metode pencarian jalan di pusat perbelanjaan ini mengambil konsep pencarian jalan keluar dari suatu labirin. Toko atau lokasi yang dituju dapat diumpakan sebagai *exit point*. Akan tetapi, untuk mencapai '*exit point*' tersebut perlu diperhatikan faktor praktis tidaknya terutama menyangkut masalah efisiensi waktu. Apabila jalur yang diusulkan malah membawa calon konsumen ataupun pengunjung mengitari seluruh pusat perbelanjaan untuk mencapai tujuan, hal ini juga merugikan. Oleh karena itu, diperlukan *threshold* atau batasan-batasan tertentu seperti yang akan dijelaskan lebih lanjut pada algoritma di bab berikutnya. Kemudian akan dibandingkan dengan algoritma runut balik biasa.

Aplikasi yang akan dibangun, memiliki fitur-fitur antara lain:

- Tampilan peta keseluruhan (semua lantai) pusat perbelanjaan pada layar.
- Informasi daftar toko yang ada berdasarkan kategori-kategori tertentu (misal toko buku, toko perhiasan, toko sepatu, restoran, dsb).
- Pencarian rute jalan ke toko yang dituju dan ditampilkan di layar.
- Pencetakan rute jalan ke toko yang dituju (jika pengunjung memilih untuk mencetak pada kertas).

Fitur terakhir yang disebutkan di atas yaitu mencetak peta di atas kertas, diambil idenya dari sistem toko buku Kinokuniya yang ada di Singapura di mana pengunjung dapat melihat lokasi rak tempat buku yang dicari berada dan sekaligus mencetak di atas kertas peta rak-rak yang ada dan menunjukkan lokasi buku tersebut berada di rak yang mana sehingga memudahkan pengunjung untuk melakukan pencarian buku tertentu. Hal ini dapat menghemat waktu pencarian terutama karena toko buku tersebut sangat luas dan akan sangat membuang waktu untuk mencari buku secara manual.

Alasan diperlukannya aplikasi pencarian rute jalan dengan algoritma runut balik ini adalah :

- Lebih memudahkan dalam belanja, supaya tidak membuat pengunjung stres dalam menemukan toko yang dituju secara langsung (asumsi pengunjung tersebut terburu-buru, tidak untuk hanya cuci mata dan jalan-jalan).
- Lebih cepat dan praktis dalam mencari toko yang diinginkan sehingga memudahkan pengunjung dan menghemat waktu.
- Dapat mengetahui rute jalan menuju lokasi toko yang dicari, apalagi dengan adanya fasilitas pencetakan peta dan rutenya di atas kertas pada sistem ini, sehingga pengunjung tidak perlu menghapal rute yang ditunjukkan pada layar.

2. METODE

2.1 Algoritma Runut Balik

```
procedure RunutBalik(input n: integer)
// Mencari semua solusi persoalan dengan
// metode runut balik skema iteratif
// Masukan: n, yaitu panjang vektor solusi
// Keluaran:
//     solusi x = x[1], x[2], ..., x[n]

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[n]) = true) then
                     if (x[1], x[2], ...,
x[n]) adalah lintasan dari akar ke daun
then
                            CetakSolusi(x)
                     endif

                     k <- k + 1
              else
                     k <- k - 1
              endif
       endwhile

// k = 0
```

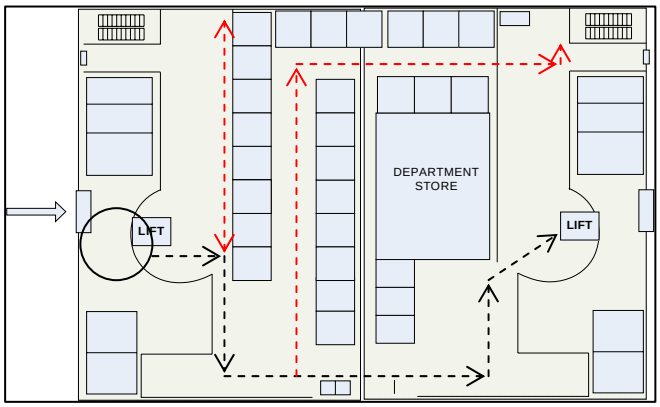
2.2 Algoritma Runut Balik dengan *Threshold* untuk Pencarian Jalan

```
Deskripsi Global
const up = 1
const tlaut = 2
const timur = 3
const tenggara = 4
const selatan = 5
const bdaya = 6
const barat = 7
const blaut = 8

// Point adalah suatu tipe bentukan untuk
// merepresentasikan posisi dari objek
type Point = record
                     x : integer
                     y : integer
              end

// Blok adalah suatu tipe bentukan yang
// merupakan satuan terkecil dari peta

type Blok =  record
                     p : Point
                     status : boolean
              end

// Tabel_of_Blok adalah representasi dari
// suatu jalan
type Tabel_of_Blok = array [1..n] of Blok

// Map adalah suatu tipe bentukan sebagai
// representasi dari peta (pusat
// perbelanjaan)
type Map = array [1..n] of
              array [1..m] of Blok

// jalan adalah variabel tempat untuk
// menampung solusi hasil
jalan : Tabel_of_Blok

// map_in adalah variabel yang menyimpan
// peta pusat perbelanjaan
map_in : Map

// threshold adalah jumlah langkah / blok
```

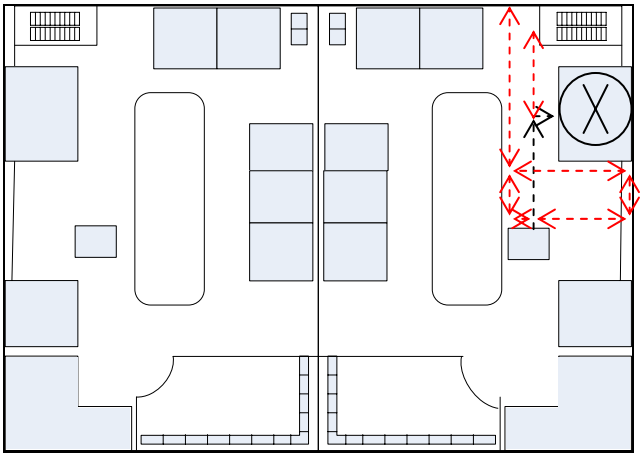
```
// maksimal yang dapat dilalui oleh
// konsumen untuk mencapai blok tujuan
threshold : integer

// findPath adalah prosedur pencarian jalan
// dari blok awal tempat konsumen berada
// hingga blok tempat tujuan yang ingin
// dicapai konsumen pada suatu peta
procedure findPath
        (Pk: Point, Pt: Point, Peta: Map)
        // Input:
        //    Pk : posisi / blok tempat
        //         konsumen berada
        //    Pt : posisi / blok tempat yang
        //         ingin dituju oleh konsumen
        //
        // Proses:
        //   Cari jalan yang mengarah ke
        //   blok tempat tujuan dari blok
        //   awal tempat konsumen berada
        //   dengan batasan panjang jalan
        //   harus lebih kecil atau sama
        //   dengan threshold
        //
        // Output:
        //   Jalan yang dapat ditempuh oleh
        //   konsumen untuk mencapai tempat
        //   tujuan


Deklarasi:
        pathFound: boolean
        i: integer
        b: Blok

Algoritma:
        pathFound <- false
        i <- 0

        jalan[i] = blok awal tempat konsumen
                   berada
        b = blok awal tempat konsumen berada

        // calculateThreshold adalah fungsi
        // untuk menghitung jumlah blok
        // dari Pk ke Pt
        calculateThreshold(Pk, Pt)

        while !(pathFound) do
        // Syarat pencarian blok jalan:
        // b adalah jalan dan bukan
        //           bangunan,
        // b belum pernah dilalui sebelumnya
              cariJalan(b, Peta)
              if b = Pt
              then
                    pathFound <- true
              else
              // isJalan adalah fungsi yang
              // mengecek apakah b adalah
              // blok yang dapat dilalui
              // oleh konsumen atau tidak
                    if isJalan(b) AND
                       panjang(jalan) <= threshold
                          then
                                simpan(jalan)
                          else
                                backtrack(jalan)
                          endif
              endif
        endwhile
endprocedure
```

3. ANALISIS KASUS

Kasus yang akan dianalisis menggunakan peta seperti tampak pada **Gambar 1** dan **Gambar 2**.

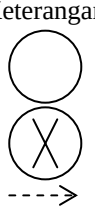


Gambar 1 Gambar Contoh Denah Pusat Perbelanjaan Lantai 1



Gambar 2 Gambar Contoh Denah Pusat Perbelanjaan Lantai 2

Keterangan Gambar:



Blok awal tempat konsumen berada

Blok tujuan yang ingin dicapai konsumen

Jalan yang ditempuh oleh konsumen

Arah pencarian jalan pada algoritma runut balik ini dimulai dari arah utara searah jarum jam dan arah yang terakhir dicoba adalah arah barat laut. Garis panah berwarna merah adalah jalan yang dapat ditempuh oleh konsumen dengan menggunakan algoritma runut balik biasa. Garis panah berwarna hitam adalah jalan yang ingin diberikan apabila konsumen bergerak dari tempat asal ke tempat tujuan. Dapat dilihat bahwa algoritma runut balik, memiliki kompleksitas waktu yang sangat tinggi karena semua kemungkinan jalan akan diuji coba sebelum akhirnya solusi jalan ditemukan.

Dengan menambahkan penggunaan *threshold* pada algoritma runut balik biasa, algoritma ini dapat menemukan solusi yang lebih cepat dan relatif lebih baik daripada solusi yang diberikan oleh algoritma runut balik biasa. Yang dimaksud dengan *threshold* di sini adalah jumlah blok maksimal yang dapat dilalui oleh konsumen untuk mencapai blok tujuan. Penggunaan *threshold* ini mempercepat algoritma untuk menemukan solusi dengan cara memangkas kemungkinan-kemungkinan jalan yang akan memperpanjang jalan yang harus ditempuh oleh konsumen. Algoritma runut balik dengan *threshold* untuk pencarian jalan ini akan relatif lebih cepat dan menghasilkan solusi jalan yang relatif lebih baik dibandingkan dengan algoritma runut balik biasa.

4. KESIMPULAN

Dari hasil perbandingan kedua algoritma, baik dari pseudo-code maupun gambar di atas, dapat dilihat bahwa algoritma yang telah dimodifikasi menghasilkan:

1. Rute yang relatif lebih pendek
2. Dalam waktu yang lebih singkat (lebih sedikit *backtracking*)

REFERENSI

[1] Munir, Rinaldi, "Diktat kuliah IF2251 Strategi Algoritmik", ITB: Bandung, 2006.

[2] http://en.wikipedia.org/wiki/Global_Positioning_System diakses pada tanggal 20 Mei 2007 pukul 12.00

[3] http://www.cs.bu.edu/teaching/alg/maze/ diakses pada tanggal 20 Mei 2007 pukul 12.00