Penerapan Pohon Untuk Menyelesaikan Masalah Labirin

Andru Putra Twinanda
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung 40135
email: ndrewh@students.itb.ac.id

Abstract – *Makalah ini membahas mengenai teori-teori dasar pohon dan penerapannya untuk menyelesaikan masalah labirin. Labirin merupakan teka-teki mencari jalan keluar yang sudah sering kita jumpai di banyak permainan. Masalah yang dibahas dalam makalah ini adalah bagaimana mencari jalan keluar dari labirin dan bagaimana pemain tidak tersesat dalam labirin baik yang berskala besar maupun yang berskala kecil.*

Kata Kunci*:pohon, labirin, jalan keluar..*

1. PENDAHULUAN

Pohon adalah graf yang khusus. Definisi dari pohon adalah graf tak berarah yang tidak mengandung sirkuit.

Berikut ini beberapa terminologi yang akan digunakan dalam makalah ini:

1. **Orang Tua (*parent*)**
   Sebuah simpul dikatakan sebagai anak dari simpul yang lainnya apabila terdapat sisi yang menghubungkan kedua simpul tersebut. Pada Gambar 1.1, simpul A merupakan orang tua dari simpul B dan simpul C merupakan orangtua dari simpul E.

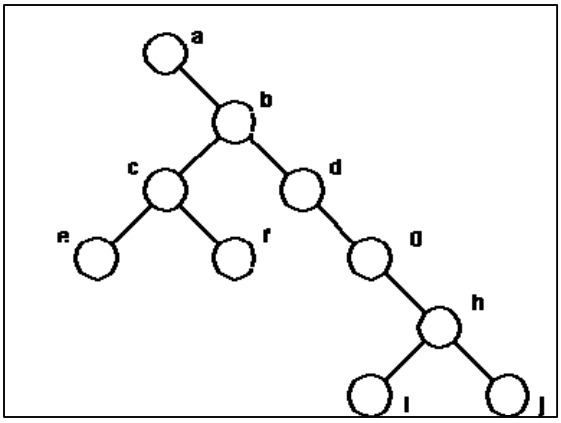
2. **Anak**
   Sebuah simpul dikatakan sebagai anak dari simpul yang lainnya apabila terdapat sisi yang menghubungkan kedua simpul tersebut. Pada Gambar 1.1, simpul B dikatakan sebagai anak dari simpul A dan simpul E anak dari simpul C.

3. **Saudara Kandung**
   Simpul yang memiliki orangtua sama disebut saudara kandung terhadap satu sama lain. Pada Gambar 1.1, D adalah saudara kandung dari C dan E adalah saudara kandung dari E.

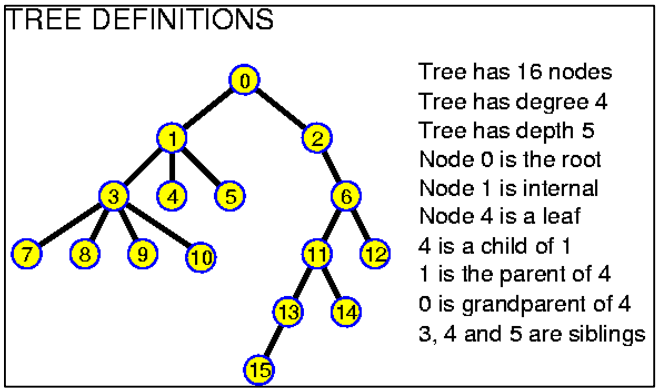
4. **Pohon *n-ary***
   Pohon *n-ary* merupakan pohon jumlah anaknya tidak tetap. Gambar 1.2 merupakan contoh dari pohon *n-ary*.

5. **Pohon biner**
   Pohon biner adalah pohon *n-ary* yang khusus karena memiliki setiap simpulnya maksimal memiliki dua anak saja. Gambar 1.1 merupakan contoh dari pohon biner. Pohon biner memiliki keistimewaan karena merupakan struktur yang paling penting dalam komputer, banyak sekali terapannya dalam ilmu computer. Beberapa contohnya adalah kode Huffman dan pohon pencarian biner (*binary search tree*)

Gambar 1.1 Pohon Biner



Gambar 1.2 Pohon *n-ary*



Terjemahan:
Pohon memiliki 16 simpul, berderajat 4, berkedalaman 5. Simpul 0 adalah akar. Simpul 1

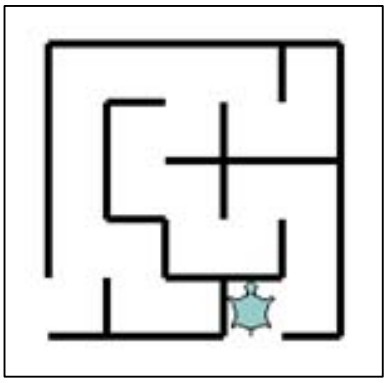
adalah simpul dalam. Simpul 4 adalah daun. Simpul 4 adalah anak dari Simpul 1. Simpul 1 adalah orang tua dari simpul 4. Simpul 0 adalah leluhur dari simpul 4. Simpul 3, 4, 5 adalah saudara kandung.

2. PEMBAHASAN MASALAH

Tentu kita mengenal permainan labirin, yaitu permainan mencari jalan keluar . Permainan ini biasa dilakukan dalam sekala besar, ataupun skala kecil yang biasa kita temukan di kertas. Permainan labirin ini juga bisa kita aplikasikan dalam kehidupan nyata. Seperti bagaimana kita dapat keluar dari sebuah hutan yang memiliki jalan bercabang.

Pada gambar 2.1, diperlihatkan contoh labirin 2 dimensi yang berskala kecil. Gambar 2.2 memperlihatkan contoh labirin 3 dimensi.

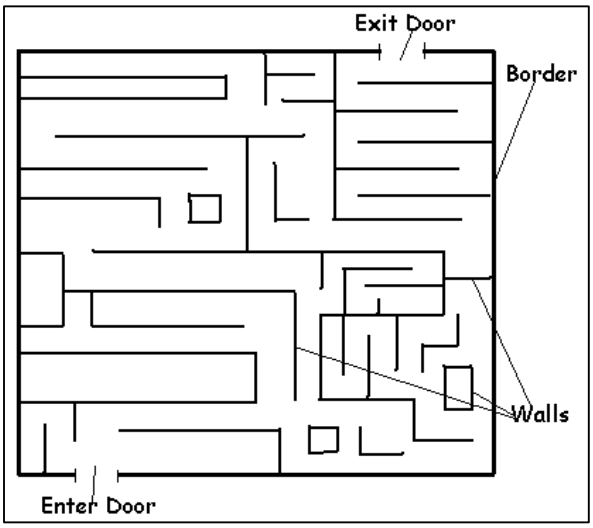
Gambar 2.1 Labirin 2 Dimensi Berskala Kecil



Gambar 2.2 Labirin 3 Dimensi

Dalam makalah ini, akan dibahas labirin 2 dimensi yang skalanya tidak terlalu besar, tetapi akan cukup menyulitkan apabila tidak diselesaikan dengan strategi tertentu. Gambar 2.3 menampilkan labirin yang akan dibahas cara penyelesaiannya dengan menerapkan pohon dalam makalah ini.

Gambar 2.3 Permasalahan Labirin



Dalam permainan ini, pemain masuk dari pintu masuk, lalu mencari jalan keluar dari labirin ini.

Bagaimana kita dapat mencari jalan keluar dalam labirin? Akan sangat mudah jika kita mencari jalan keluar labirin jika labirin yang dimaksud adalah permainan skala kecil yang biasa kita temukan di kertas, seperti yang kita lihat pada gambar permodelan labirin di atas. Dalam permainan labirin skala kecil ini, kita dapat melihat dengan jelas jalan masuk dan jalan keluarnya. Sedangkan, jika permainan labirin yang dimaksud adalah permainan labirin skala besar, yaitu permainan labirin 3 dimensi, kita tidak bisa melihat jalan keluar dari permainan ini, dan yang dapat kita lihat adalah petak jalan yang kita sedang lewati.

Ternyata, permainan labirin dapat kita modelkan juga dalam bentuk pohon, atau biasa kita kenal dengan *n-ary*. Dengan memodelkan labirin dengan *n-ary*, kita akan lebih mudah menemukan jalan keluar dalam permainan labirin. Berikut ini adalah cara mencari jalan keluar dari labirin dengan memodelkan labirin dengan *n-ary*:

1. Anggaplah jalan masuk sebagai parent dari pohon *n-ary*

2. Caban-cabang jalan yang terlihat dari parent atau pintu masuk kita gambar dan dapat kita analogikan sebagai simpul.

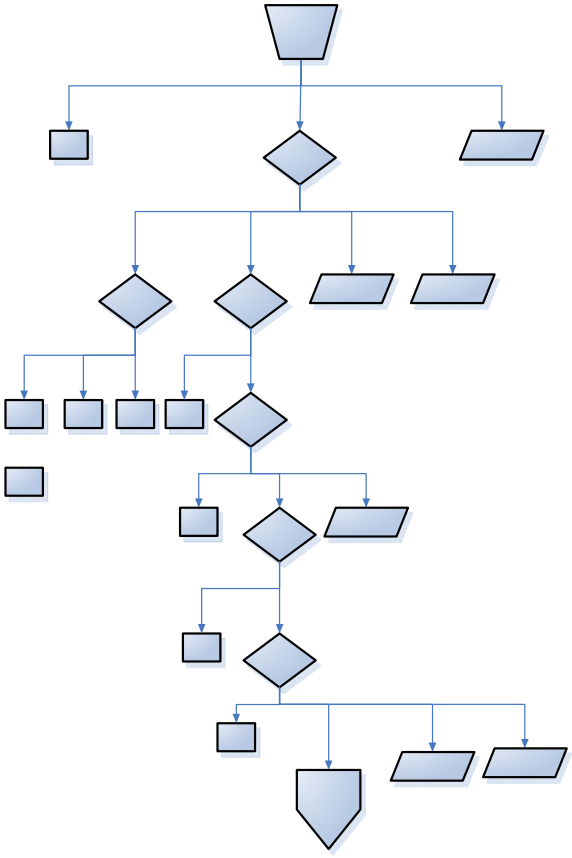
3. Dalam membuat pohon *n-ary* dari labirin ini, kita harus dapat membuat semua simpul yang dianalogikan sebagai cabang setiap jalan yang kita lewati pada labirin. Agar tidak tersesat, kita mulai merunutkan jalan dengan memilih jalan paling kiri atau jalan paling kanan di setiap cabang jalan permainan labirin.

4. Jalan yang terlihat dari cabang kita gambar tanpa perlu kita masuki terlebih dahulu. Jalan yang terlihat dari cabang dapat kita maski setelah percabangan sebelumnya telah kita masuki dan hasilnya adalah jalan buntu.

5. Jalan keluar , jalan yang terlihat dari percabangan dan jalan buntu adalah daun pada pohon *n-ary*. Jalan keluar adalah salah satu daun dalam pohon labirin *n-ary* ini.

Yang menjadi kunci utama dalam mencari pintu keluar adalah dengan cara memilih percabangan jalan secara berurutan. Kita dapat terus memilih percabangan jalan paling kiri di setiap percabangan sampai menemukan jalan buntu. Apabila menemukan jalan buntu, kembali ke tempat awal di mana kita memlih jalan dan mulai memilih cabang kedua paling kiri dari percabangan jalan ini.
\
Dengan penjelasan sebelumnya, kita dapat membuat pohon *n-ary* dari gambar labirin yang sedang kita lewati. Sketsanya adalah sebagai berikut.

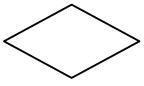
Gambar 2.4
Analogi labirin ke dalam pohon *n-ary*

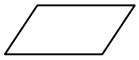


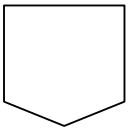
Keterangan :


: jalan masuk


:jalan buntu (terlihat langsung dari user)


: percabangan


: jalan yang terlihat dari cabang


: jalan keluar

Dengan begini, dapat dicegah pemain memasuki jalan yang jelas-jelas buntu (terlihat bahwa buntu). Pemain hanya perlu memasuki percabangan dan jalan yang terlihat dari cabang.

Untuk mencegah tersesatnya pemain di dalam labirin, dapat digunakan metode pewarnaan pohon, yaitu dengan cara mewarnai cabang (*distinguishing*) yang sudah dilewati. Dengan cara seperi ini, kemungkinan pemain tersesat dapat dihindari.

Masalah labirin inipun juga dapat diselesaikan dengan menggunakan pohon biner. Pohon *n-ary* yang sudah kita buat dapat kita buat menjadi pohon biner. Dengan mengubah menjadi pohon biner, penyelesaian masalah di komputer dapat diselesaikan dengan lebih mudah.

Proses pengubahan pohon *n-ary* menjadi pohon biner adalah sebagai berikut:

1. Kita ambil satu simpul sebagai parent
2. Percabangan pertama kita anggap sebagai anak kiri
3. Percabangan kedua kita anggap sebagai anak kanan dari percabangan pertama, percabangan ketiga kita anggap sebagai anak kanan dari percabangan kedua, dan seterusnya.
4. Jalan buntu tidak memiliki anak sama sekali

Dengan mentranslasi pohon *n-ary* menjadi pohon biner, dapat mempermudah pengambilan keputusan karena hanya perlu memilih antara anak kanan atau anak kiri.

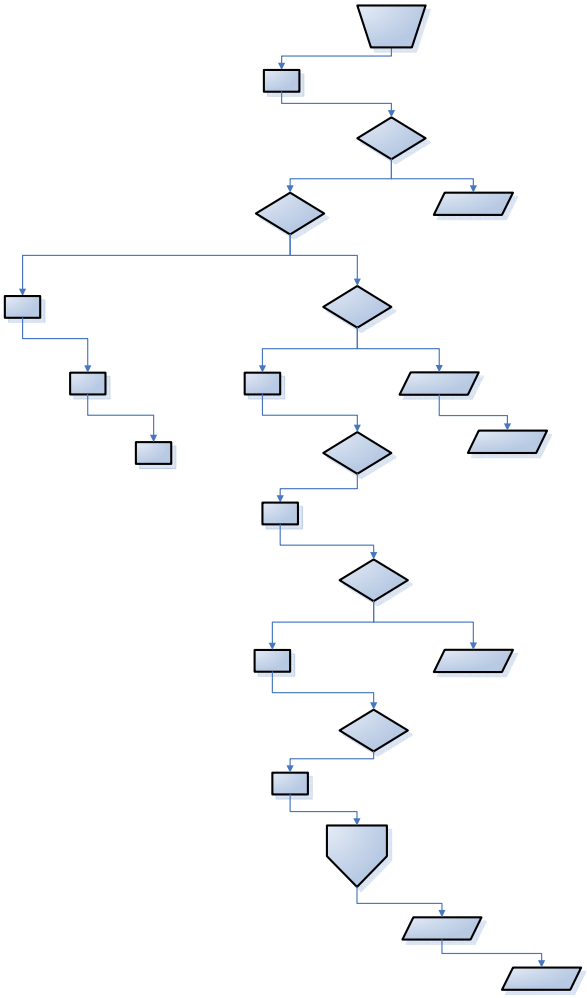
Meski terlihat lebih mudah, penggunaan pohon biner untuk memecahkan masalal labirin ini tetap memerlukan pewarnaan pada simpul yang telah dilewati agar pemain dapat kembali ke simpul yang masih memiliki anak kiri dan anak kanan, tetapi anak kirinya sudah dimasuki (karena kita selalu memasuki anak kiri terlebih dahulu).

Dilihat dari segi algoritmik, penggunaan pohon biner akan lebih menguntungkan karena jumlah anak yang konsisten, sehingga pemain tak perlu memperhitungkan kemungkinan perubahan jumlah anak.

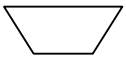
Namun, dilihat dari segi permainan dan pengsketsaan akan lebih mudah dengan menggunakan pohon *n-ary* karena apabila pemain menemukan jalan buntu, pemain cukup kembali ke simpul orang tua dan mengambil jalan yang merupakan saudara kandung dari simpul pemain berasal.

Setelah mengalami proses pengubahan ke pohon biner, maka analogi permasalahannya akan berubah menjadi:

Gambar 2.5 Pohon Biner Hasil Translasi

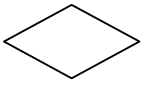


Keterangan :

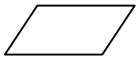


: jalan masuk

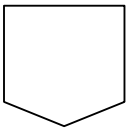
:jalan buntu (terlihat langsung dari user)



: percabangan



: jalan yang terlihat dari cabang



: jalan keluar

Algoritma yang dapat digunakan untuk menyelesaikan permasalahan labirin dengan aplikasi pohon adalah algoritma *recursive backtracking*. Algoritma ini pasti akan menemukan sebuah jalan keluar, akan tetapi lintasan yang ditemukan belum tentu merupakan lintasan yang terpendek.

Algoritma ini berjalan dengan proses masuk ke salah satu simpul anak. Apabila mengalami kebuntuan kembali ke simpul orang tuanya (sekarang menjadi *current node*) dan menghapus lintasan dari *current node* ke simpul sebelumnya. Dan secara rekursif berjalan terus hingga mencapai jalan keluar.

```
function BackTracking (P1 : PohonBiner) → boolean
{Mengembalikan True apabila sudah berada dalam finish}

Deklarasi:
function finish (…)
{Mengembalikan True apabila simpul input merupakan jalan keluar}

function buntu (…)
{Mengembalikan True apabila simpul merupakan jalan buntu}

procedure HapusPath (…)
{Kembali ke simpul yang masih memiliki anak kanan yang belum dijelajahi, dan menghapus path antara simpul awal ke simpul akhir}

Realisasi:
if (finish(Akar(P)))
    return (True)
else
    if (IsOneElmt(P))
        BackHapusPath()
    endif

    if (buntu(P))
        BackTracking(Right(P))
    else
        BackTracking(Left(P))
    endif
endif
```

3. KESIMPULAN

Pohon memiliki kegunaan yang sangat luas. Banyak masalah yang dapat diselesaikan dengan menggunakan pohon. Salah satunya adalah penyelesaian labirin. Terlebih lagi di ilmu computer, pohon merupakan salah satu tipe data yang sering digunakan.

Untuk beberapa masalah tertentu, akan lebih mudah disintesis penyelesaiannya apabila dibuat ke dalam bentuk pohon biner. Algoritma untuk penyelesaian dari masalah pohon binerpun lebih mudah dibandingkan dengan penyelesaian algoritma untuk pohon *n-ary*.

Pada kasus nyata, akan lebih mudah untuk menggunakan pohon *n-ary* karena mempermudah penggambaran dan pembentukan pohon. Lintasan pun lebih enak untuk dibaca.

Kekurangan dari proses penyelesaian masalah labirin dengan menggunakan pohon adalah pemain harus benar-benar teliti dalam proses pengubahan atau penganalogian labirin ke bentuk pohon karena tanpa ketelitian yang penuh akan sangat mudah terjadi kesalahan dalam pengsketsaan pohon.

Sekian makalah ini, untuk segala kekurangannya harap dimaklumi. Apabila algoritma tidak sesuai, harap dimaklumi karena algoitma dibuat sesuai dengan pemikiran sendiri hanya dengan membaca penjelasan dari referensi. Terima kasih.

DAFTAR REFERENSI

[1] Munir,Rinaldi. “Bab 9 Pohon”,*Matematika Diskrit*, Edisi Kempat, 2006, hal IX 1 –IX 31

[2] Liem, Inggriani. “Pohon”.*Diktat Kuliah IF221 Dasar Pemrogramnan Pemrograman Fungsional*, 2000, hal 99-113

[3] http://users.informatik.uni-halle.de , Tanggal akses: 1 Januari 2008

[4] http://pages.cpsc.ucalgary.ca , Tanggal akses: 1 Januari 2008

[5] http://www.bfoit.org, Tanggal akses: 2 Januari 2008

[6]http://en.wikipedia.org, Tanggal akses: 2 Januari 2008