PENERAPAN ALGORITMA DFS DAN BFS DALAM PERMAINAN ATLANTEINE

Rachmansyah Budi Setiawan (13507014)

Program Studi Teknik Informatika Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if17014@students.if.itb.ac.id

ABSTRAK

Perkembangan pengetahuan yang muncul membuat banyaknya buah pemikiran baru untuk membuat permainan. Salah satu jenis permainan yang berkembang adalah permainan dengan jenis strategi dan salah satu contoh permainannya adalah Atlanteine.

Permainan strategi menuntut pemainnya memutar otak untuk menyelesaikannya. Begitu pula dengan Atlanteine. Hal ini membuat para pemainnya memikirkan langkah – langkah penyelesaian yang sesuai untuk menyelesaikan permainan. Langkah – langkah ini kita kenal dengan istilah algoritma.

Contoh algoritma yang dapat digunakan untuk menyelesaikan permasalah dalam permainan ini adalah algoritma DFS dan BFS. Kedua algoritma ini memiliki kelebihan dan kekurangannya masing – masing dan cukup sulit untuk menentukan algoritma mana yang lebih efektif, efisien, dan optimal dalam pencarian solusi pada permainan ini karena pengaruh beberapa faktor.

Kata kunci: Atlanteine, strategi, DFS, BFS.

1. PENDAHULUAN

Perkembangan pengetahuan merangsang otak manusia dalam banyak hal, salah satunya adalah menghasilkan ide – ide baru. Ide – ide baru ini termasuk juga ide dalam dunia permainan. Hal ini menyebabkan kemunculan banyak permainan baru. Salah satu tipe permainan yang banyak berkembang adalah permainan dengan tipe strategi. Bagi para pencintanya, tentu saja hal ini merupakan hal yang menggembirakan. Mereka kembali dipaksa untuk memutar otak dalam menyelesaikan permainan – permainan tersebut dengan segala aturannya.

Salah satu jenis permainan dengan tipe strategi ini adalah permainan yang bernama Atlanteine. Permainan yang dapat dimainkan *online* pada situs www.kadokado.com ini merupakan permainan yang mirip dengan permainan yang meminta pemainnya untuk mengeluarkan karakternya dari dalam suatu labirin. Perbedaannya, jalan keluar pada permainan ini berupa sebuah lubang yang terletak di tengah – tengah peta. Karakter pada permainan ini berupa sebuah labu yang dapat digerakkan ke atas, bawah, kiri, dan kanan. Saat pemain memilih suatu arah untuk pergerakan labu, labu baru akan berhenti bergerak ke arah yang telah dipilih jika bertemu batu atau kotak kayu. Jika pada arah gerak labu tidak ada batu atau kotak kayu, maka labu akan terjatuh dan pemain akan dipaksa untuk mengulang dari posisi awal labu pada permainan tersebut. Dalam suatu permainan, pemain dapat menggerakkan paling banyak satu kotak kayu ke arah yang memungkinkan (masih di dalam peta dan tidak ada kotak kayu lain atau ada batu).

Gambar 1. Contoh gambar permainan Atlanteine

Dengan batasan permasalahan seperti dalam permainan ini, maka dapat dilihat bahwa algoritma DFS dan BFS dapat digunakan karena melibatkan beberapa langkah penyelesaian dan dapat menciptakan suatu pohon ruang status.

2. METODE

2.1 Pemecahan dengan Algoritma BFS

Secara sederhana, penerapan algoritma BFS adalah dengan memeriksa keempat arah yang mungkin (atas, kanan, bawah, kiri) dari tiap simpul awal yang disimpan dalam sebuah antrian sampai solusi ditemukan. Jadi, jika langkah pencarian solusi dibuat menjadi sebuah pohon, akan terbentuk pohon merentang yang urutan pembentukan daunnya adalah secara horizontal terlebih dahulu.

Untuk mempermudah simulasi, maka gambar permainan di atas diubah terlebih dahulu menjadi sebuah papan berukuran 13 x 13 yang dapat berisi B jika terdapat batu, K jika terdapat kotak kayu, T jika terdapat lubang tujuan, dan L jika terdapat karakter labu.

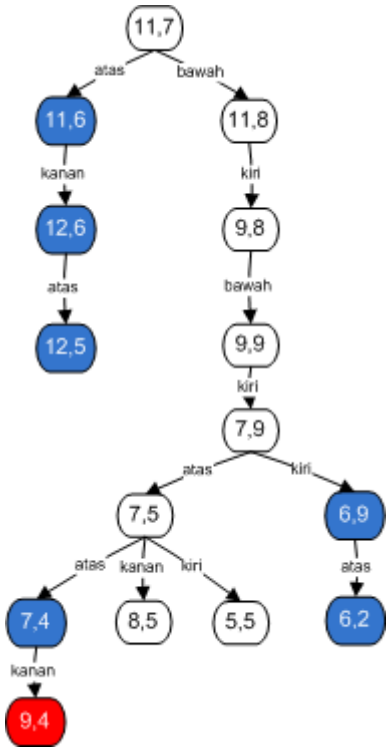
y\x	1	2	3	4	5	6	7	8	9	10	11	12	13
1		B				B			B				
2								K	B				
3	B	K							B	B			
4							K		T			B	
5	B			K					K				
6											K		B
7			B	K				B	B	B	L	B	B
8								B					
9						K				B	B		
10								K	B				
11										B			
12				B									
13										B			

Gambar 2. Hasil pengubahan kondisi permainan pada Gambar 1

Algoritma BFS yang dapat diterapkan untuk menyelesaikan permainan ini memerlukan beberapa elemen, yaitu sebuah antrian yang berisi posisi, penanda apakah penggeseran kotak kayu sudah pernah dilakukan, dan matriks yang menandai suatu posisi sudah pernah dikunjungi atau belum. Algoritmanya sendiri secara garis besar sebagai berikut (asumsi yang digunakan adalah permainan selalu dapat diselesaikan atau dengan kata lain minimal ada satu cara untuk mencapai lubang tujuan dari posisi awal labu):

1. Masukkan posisi awal labu pada antrian.
2. Untuk tiap empat arah mulai dari simpul paling awal dalam antrian, periksa apakah sepanjang arah tersebut terdapat batu, kotak kayu, atau lubang tujuan kemudian hapus simpul tersebut dari antrian. Tandai posisi tersebut pada matriks penanda sebagai posisi yang sudah dikunjungi.
3. Jika ada batu atau kotak kayu dan letaknya tidak bertetangga langsung dengan posisi simpul dalam antrian yang sedang diperiksa serta posisi berhenti labu belum pernah dikunjungi, maka tambahkan posisi berhenti labu ke antrian.
4. Jika ada kotak kayu, letaknya bertetangga langsung dengan posisi simpul dalam antrian yang sedang diperiksa, dan belum pernah dilakukan penggeseran kotak kayu serta posisi berhenti labu belum pernah dikunjungi, maka geser kotak kayu ke arah yang dipilih kemudian tambahkan posisi berhenti labu ke antrian serta ubah penanda yang menyatakan sudah pernah menggeser kotak kayu.
5. Ulangi pemeriksaan mulai langkah 2 sampai 3 untuk simpul berikutnya dalam antrian sampai lubang tujuan ditemukan.

Penerapan algoritma BFS di atas untuk menyelesaikan permainan dengan kondisi seperti pada gambar 1 akan menghasilkan sebuah pohon ruang status seperti pada Gambar 3. Urutan pemeriksaannya adalah ke atas, kanan, bawah, barulah yang terakhir ke kiri. Warna biru menandakan bahwa pada untuk mencapai posisi tersebut sudah pernah dilakukan penggeseran kotak kayu, sementara itu warna merah menandakan lubang tujuan. Selain itu, langkah atau pemeriksaan yang tidak valid tidak digambarkan pada pohon ruang status ini.



Gambar 3. Hasil eksekusi algoritma BFS untuk kondisi permainan seperti pada Gambar 1

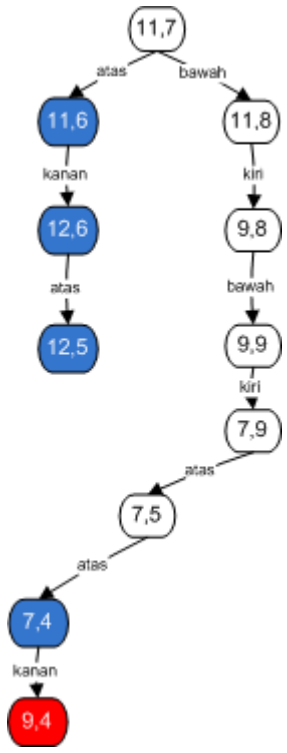
2.2 Pemecahan dengan Algoritma DFS

Dengan algoritma DFS, pencarian solusi ditekankan pada satu arah sampai memang tidak memungkinkan lagi untuk melanjutkan pencarian pada arah tersebut. Berbeda dengan algoritma BFS, jika langkah pencarian solusinya dibuat menjadi pohon, maka urutan pembentukan daun – daunnya adalah secara vertikal terlebih dahulu.

Algoritma DFS yang dapat diterapkan untuk menyelesaikan permainan ini tidak jauh berbeda dengan algoritma BFS yang telah dijelaskan sebelumnya. Algoritma ini juga memerlukan matriks penanda suatu posisi sudah pernah dikunjungi atau belum dan juga penanda apakah penggeseran kotak kayu sudah pernah dilakukan. Hanya saja, karena algoritma yang digunakan bersifat rekursif, maka tidak diperlukan antrian penyimpanan posisi untuk posisi simpul awal pada pengulangan pemeriksaan. Meskipun begitu, antrian dapat digunakan untuk menyimpan langkah penyelesaian (sebenarnya cukup dengan *array* biasa). Secara garis besar, algoritmanya sebagai berikut:

1. Periksa apakah posisi labu sekarang sama dengan posisi labu tujuan (basis). Jika sama masa pencarian solusi selesai.
2. Jika tidak sama, maka tambahkan posisi tersebut ke antrian. Untuk suatu arah yang dipilih dari posisi labu sekarang dan arah tersebut belum diperiksa, periksa apakah sepanjang arah tersebut terdapat batu, kotak kayu, atau lubang tujuan. Tandai posisi tersebut pada matriks penanda sebagai posisi yang sudah dikunjungi.
3. Jika ada batu atau kotak kayu dan letaknya tidak bertetangga langsung dengan posisi simpul dalam antrian yang sedang diperiksa serta posisi berhenti labu belum pernah dikunjungi, maka tambahkan posisi berhenti labu ke antrian.
4. Jika ada kotak kayu, letaknya bertetangga langsung dengan posisi simpul dalam antrian yang sedang diperiksa, dan belum pernah dilakukan penggeseran kotak kayu serta posisi berhenti labu belum pernah dikunjungi, maka geser kotak kayu ke arah yang dipilih kemudian tambahkan posisi berhenti labu ke antrian serta ubah penanda yang menyatakan sudah pernah menggeser kotak kayu.
5. Jika dari keempat arah itu tidak ada langkah yang memungkinkan lagi dan labu belum mencapai lubang tujuan, maka pemeriksaan mundur ke posisi pada antrian yang memiliki arah yang belum dicek sambil menghapus posisi – posisi yang sudah diperiksa ke semua arah dari antrian.
6. Ulangi pemeriksaan mulai langkah 2 sampai 5 secara rekursif sampai untuk suatu posisi semua arah selesai diperiksa atau lubang tujuan ditemukan.

Penerapan algoritma DFS di atas untuk menyelesaikan permainan dengan kondisi seperti pada gambar 1 akan menghasilkan sebuah pohon ruang status seperti pada Gambar 4. Urutan pemeriksaannya adalah ke atas, kanan, bawah, barulah yang terakhir ke kiri. Warna biru menandakan bahwa pada untuk mencapai posisi tersebut sudah pernah dilakukan penggeseran kotak kayu, sementara itu warna merah menandakan lubang tujuan. Selain itu, langkah atau pemeriksaan yang tidak valid tidak digambarkan pada pohon ruang status ini.



Gambar 4. Hasil eksekusi algoritma DFS untuk kondisi permainan seperti pada Gambar 1

3. ANALISIS

Jika melihat contoh kasus di atas, terlihat bahwa penggunaan algoritma DFS lebih efisien dalam menyelesaikan permainan ini dibandingkan dengan algoritma BFS. Namun sebenarnya ini hanyalah kasus rata – rata. Terdapat beberapa kasus yang lebih efisien jika dipecahkan menggunakan algoritma BFS dibandingkan algoritma DFS. Selain itu, hal ini juga tidak terlepas dari pemilihan urutan atau prioritas pemeriksaan. Pada contoh

di atas, urutan pemeriksaannya adalah atas, kanan, bawah, dan yang terakhir kiri. Jika menggunakan urutan yang berbeda, mungkin saja akan didapatkan efisiensi yang berbeda dan mungkin saja BFS akan lebih efisien daripada DFS.

4. KESIMPULAN

Permainan seperti Atlanteine ini dapat dipecahkan dengan algoritma DFS dan BFS yang sesuai. Namun untuk masalah jumlah langkah pemeriksaan (yang berarti efisiensi waktu) akan berbeda – beda dan tergantung kepada beberapa hal, misalnya urutan arah yang diperiksa. Oleh karena itu, algoritma mana yang secara umum lebih baik untuk menyelesaikan permainan seperti ini masih sulit ditentukan.

REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika, 2006.

[2] http://www.kadokado.com
Waktu akses: Sabtu, 02 Januari 2010 pukul 18.17

[3] http://en.wikipedia.org/wiki/Depth-first_search
Waktu akses: Sabtu, 02 Januari 2010 pukul 21.52

[4] http://en.wikipedia.org/wiki/Breadth-first_search
Waktu akses: Sabtu, 02 Januari 2010 pukul 21.52