Aplikasi Algoritma BFS pada Permainan *Squarescape*

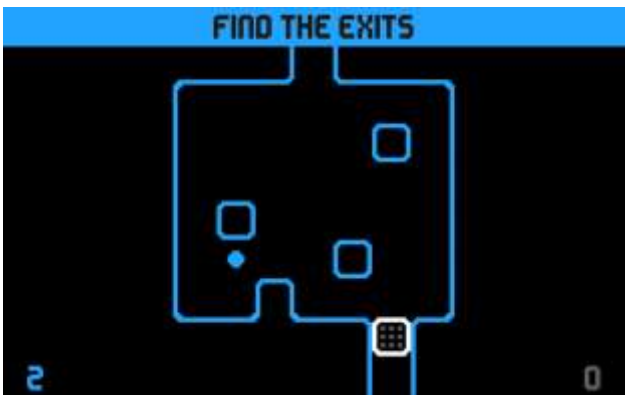
Cliff Jonathan 13513044[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13513044@std.stei.itb.ac.id

***Abstract*—Algoritma *Breadth First Search* (BFS) seringkali digunakan untuk mencari solusi terpendek dari sebuah permasalahan. Salah satu permasalahan yang memanfaatkan algoritma ini untuk mendapatkan langkah terpendek dalam mencari solusinya adalah pada permainan *Squarescape*. *Squarescape* adalah permainan di mana pemain diminta untuk menggeser sebuah kotak untuk mendapatkan bola dan keluar melalui pintu keluar yang telah ditentukan. Pergeseran hanya bisa dihentikan oleh objek lain, bisa berupa lantai pemberhentian, maupun dinding. Pada permainan *Squarescape*, jumlah langkah yang dilakukan untuk mencapai solusi dihitung, semakin pendek langkah yang dilakukan untuk menyelesaikan sebuah masalah semakin baik.**

***Index Terms*—squarescape, BFS, traversal, breadth, solusi terpendek.**

I. PENDAHULUAN

Algoritma traversal untuk mencari solusi dari sebuah permasalahan ada banyak, mulai dari DFS yang mentraversal sampai ke *dead-end*, BFS yang mentraversal semua simpul yang terbuka di suatu tahap, *branch and bound* yang membuka simpul-simpul dengan *cost* minimum/maksimum, dan masih banyak lagi. Setiap algoritma traversal memiliki kelebihannya masing-masing. Dengan kelebihan yang ada, kita bisa menentukan algoritma mana yang cocok untuk permasalahan yang kita hadapi.



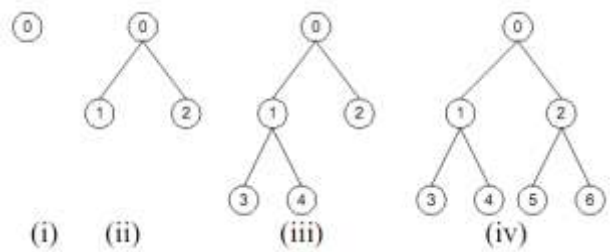
Gambar 1.1 Tampilan permainan *Squarescape*.

Pada permainan *Squarescape*, yang menjadi faktor permasalahan adalah menggeser kotak untuk mengambil bola, kemudian keluar melalui pintu keluar yang ditentukan dengan langkah terpendek. Dengan menggunakan algoritma yang tepat, maka permasalahan ini dapat diselesaikan dengan mudah. Algoritma yang dipilih adalah BFS karena BFS memiliki kelebihan yaitu lebih cepat untuk mencapai simpul solusi dibandingkan dengan DFS dan algoritma lainnya, walaupun membutuhkan ruang status yang lebih banyak.

II. DASAR TEORI

A. *Breadth First Search* (BFS)

BFS adalah algoritma traversal untuk mencari kemungkinan dengan cara membuka simpul-simpul dimulai dari simpul akar, kemudian semua simpul tetangga yang memungkinkan di level itu. Pada tahap berikutnya, dari semua simpul yang dibuka, akan dilakukan proses yang sama yaitu membuka semua simpul tetangga level berikutnya, dan seterusnya sampai simpul solusi tercapai.



Gambar 2.1 Ilustrasi proses BFS.

Pada gambar 2.1, algoritma BFS dimulai dari simpul 0. Simpul 0 memiliki 2 anak yaitu simpul 1 dan 2, maka kedua anak simpul tersebut dijelajahi. Pada langkah berikutnya, untuk simpul 1 dan 2 yang baru dijelajahi akan dibuka dan dijelajahi semua simpul anaknya lagi. Pada langkah (iii) terlihat bahwa simpul 1 memiliki simpul anak 3 dan 4, dan pada langkah (iv) terlihat bahwa simpul 2 memiliki simpul anak 5 dan 6. Simpul 3, 4, 5, dan 6 kemudian dijelajahi.

BFS berbeda dengan DFS yang menjelajahi salah satu anak pohon terlebih dahulu sampai penjelajahan bertemu dengan jalan buntu (*dead-end*) kemudian melakukan runut

balik dan menjelajahi simpul yang belum dijelajahi. BFS membutuhkan lebih banyak ruang status daripada DFS karena simpul yang dibuka di satu level banyak (semua simpul tetangga yang memungkinkan), sedangkan DFS hanya membuka simpul yang sedang dijelajahinya dan kemudian menjelajahi salah satu anak simpulnya terlebih dahulu. Namun, karena BFS membuka semua simpul tetangga yang ada, BFS akan lebih cepat dalam menemukan solusi untuk permasalahan karena tidak perlu melakukan traversal sampai penjelajahan bertemu dengan jalan buntu (*dead-end*) seperti yang dilakukan oleh algoritma DFS.

Algoritma BFS memiliki kompleksitas waktu yaitu :

$$1 + b + b^2 + b^3 + b^4 + ... + b^d = O(b^d)$$

Sedangkan kompleksitas ruangnya adalah :

$$O(b^d)$$

Dengan b adalah *branching factor* yaitu maksimum pencabangan yang memungkinkan dari sebuah simpul, dan d adalah *depth* yaitu kedalaman dari solusi terbaik.

B. *Squarescape*

Squarescape adalah permainan mengasah otak yang dikembangkan oleh Noodlecake Studios Inc dengan genre *Puzzle* untuk OS Android dan iOS. Pada permainan ini, pemain diminta untuk menggeser kotak putih untuk mendapatkan bola yang ada di sebuah *stage*, kemudian mencari jalan keluar dari stage itu melalui pintu keluar yang ditentukan. Penggeseran kotak dilakukan dengan melakukan *swipe* pada layar sentuh ke arah yang diinginkan. Yang menarik dari permainan ini adalah bahwa pemain tidak bisa menggeser kotak ke koordinat posisi yang diinginkan secara mudah karena kotak hanya akan berhenti saat menabrak dinding atau berada di atas lantai pemberhentian. Kotak juga tidak bisa digeser sebelum berhenti, sehingga bila pemain melakukan *swipe* terhadap layar sentuhnya di saat kotak masih bergerak, kotak akan tetap terus bergerak ke arah yang ditujunya. Pemain harus berpikir bagaimana cara untuk sampai ke posisi yang diinginkan dengan memilih langkah-langkah yang tepat dengan menabrakkan kotak ke dinding, dan/atau memberhentikan kotak tersebut di lantai perhentian. Selain itu, terdapat juga berbagai rintangan seperti lantai X di mana kotak pemain akan kembali ke posisi semula ketika *stage* baru dimulai.

Adapun keterangan tampilan permainan *Squarescape* adalah sebagai berikut :

 = kotak pemain

 = kotak dinding

 = bola yang harus diambil pemain

 = lantai pemberhentian. Kotak pemain akan berhenti bila menginjak lantai ini.

 = lantai rintangan X. Kotak pemain akan kembali ke posisi semula ketika *stage* baru dimulai bila menginjak lantai ini.

Contoh *stage* :

Gambar 2.2 *Screenshot stage* 1 dari permainan *Squarescape*.

Pada gambar 2.2, terlihat bahwa kotak pemain berada di bawah dan satu-satunya jalan yang bisa dilakukan adalah menggeser kotak tersebut ke atas karena kiri, kanan, dan bawah kotak merupakan dinding.

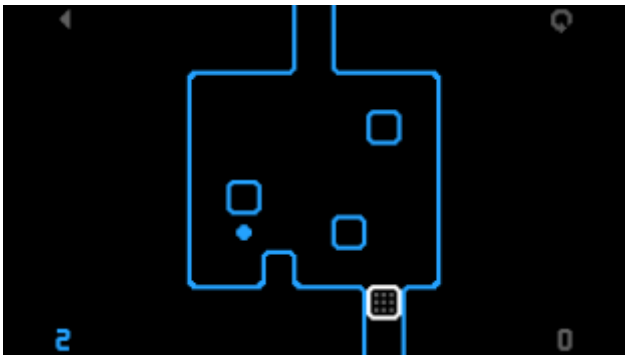
Gambar 2.3 Hasil pergeseran kotak ke atas. Kotak pemain akan berhenti saat menabrak dinding.

Pergeseran ini terus dilakukan sampai kotak pemain mendapatkan bola dan keluar melalui pintu keluarnya. Pada contoh *stage* 1 di atas, pintu keluar adalah jalan menuju ke atas tempat berada bola yang harus didapatkan juga, maka pemain akan menggeser kotak pemain ke kanan, lalu ke atas untuk menyelesaikan *stage* tersebut.

III. ANALISIS PERMASALAHAN

A. Representasi Graf dari *Stage*

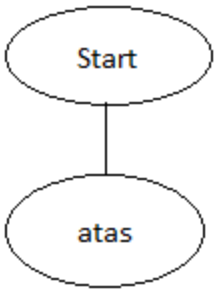
Misalnya untuk *stage* 2 ini :



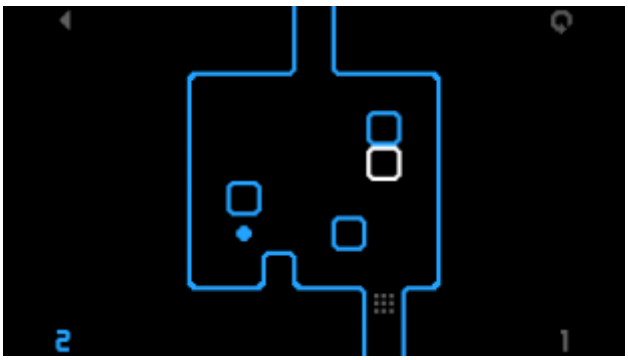
Gambar 3.1 *Screenshot stage* 2.

Pada saat *stage* ini baru dimulai, kotak pemain berada di bawah, terletak di lantai pemberhentian. Kondisi permulaan ini kita tambah ke graf sebagai akar yang paling atas, misalnya bernama "Start".

Kemudian, dapat kita lihat bahwa kotak pemain hanya bisa digeser ke atas karena kanan dan kiri kotak adalah dinding, sedangkan kotak pemain tidak bisa digeser ke bawah karena itu adalah pintu asal kotak pemain dari *stage* sebelumnya. Maka, ditambahkan anak sesuai dengan nama arah pergeseran yaitu "atas".

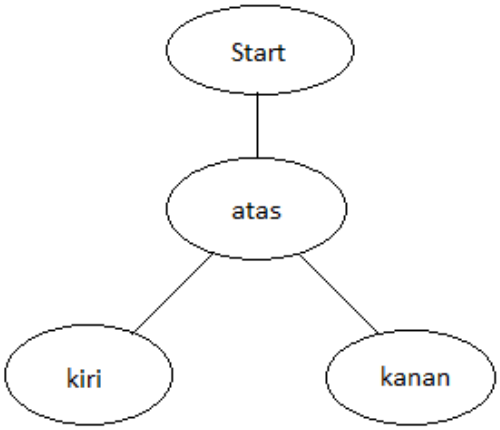


Dari simpul anak "atas" ini, kemudian kita akan mendapati kondisi *stage* sebagai berikut.

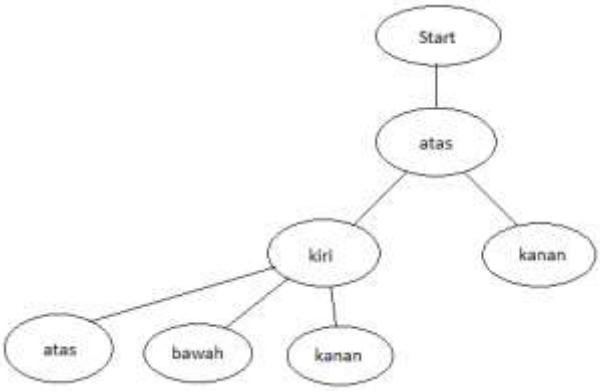


Gambar 3.2 Kondisi *stage* setelah kotak pemain digeser ke atas

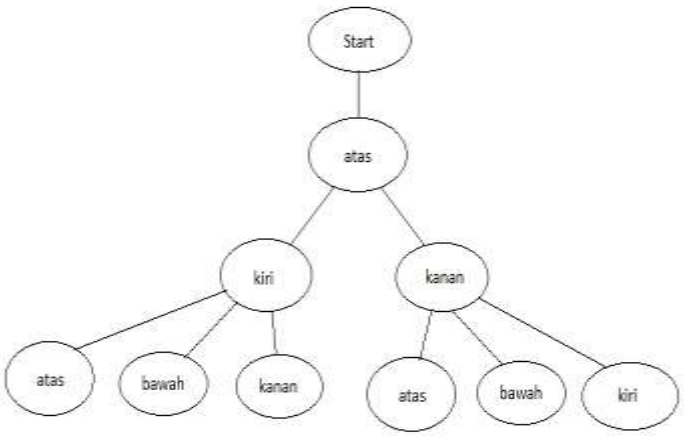
Dari kondisi ini, kita bisa melihat bahwa kotak hanya bisa digerakkan ke kiri, ke kanan, atau ke bawah. Pergeseran kotak ke bawah berarti kembali ke akar simpul (menuju ke simpul Start, karena hanya melakukan invers dari pergeseran sebelumnya), berarti anak simpul yang perlu ditambahkan berikutnya adalah "kiri" dan "kanan".



Bila kotak pemain digeser ke kiri, maka pemain akan terbentur dengan dinding kiri, maka arah pergeseran yang bisa dilakukannya hanyalah atas, bawah, dan kanan (arah ke kanan dimasukkan karena bila kotak digeser ke kanan, kotak bukan berada di posisi sebelum kotak digeser ke kiri, melainkan kotak akan menabrak dinding kanan).



Kemudian kita akan menelusuri kemungkinan dari simpul "kanan" di *level* pohon kedua. Bila kotak digeser ke kanan, maka akan terbentuk kemungkinan berikutnya yaitu kotak bisa digeser ke atas, bawah, dan kiri.



Simpul "atas", "bawah", dan "kanan" di *level* 3 dari

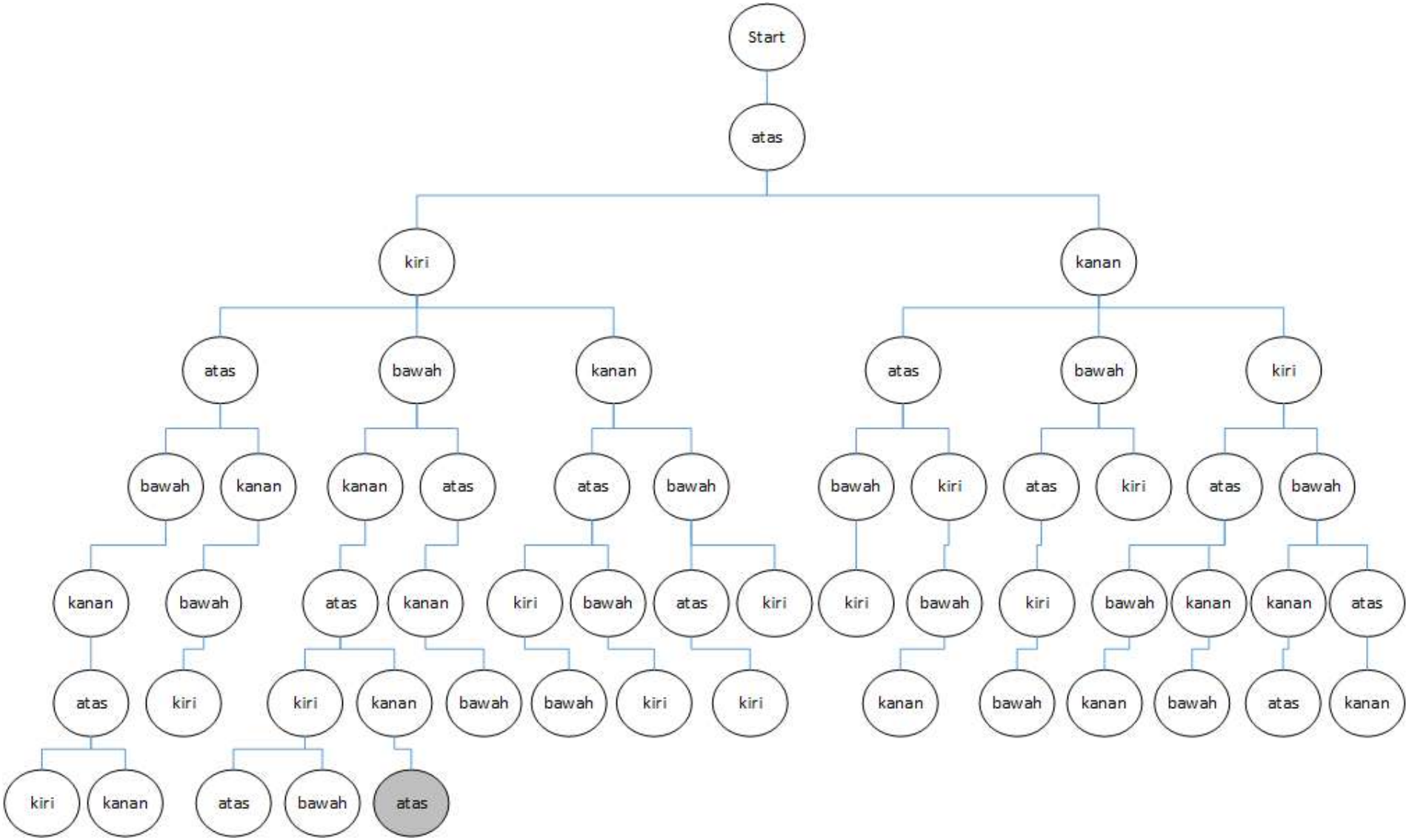
simpul "kiri" kemudian akan ditraversal juga anak simpulnya. Untuk simpul "atas" akan terbentuk simpul "kanan" dan "bawah", untuk simpul "bawah" akan terbentuk simpul "atas" dan "kanan", sedangkan untuk simpul "kanan" akan terbentuk simpul "atas" dan "bawah".

Simpul "atas", "bawah" dan "kiri" di *level* 3 dari simpul "kanan" juga akan ditraversal anak simpulnya. Untuk simpul "atas" akan terbentuk simpul "bawah" dan "kiri", untuk simpul "bawah" akan terbentuk simpul "kiri" dan "atas", sedangkan untuk simpul "kiri" akan terbentuk simpul "atas" dan "bawah".

Traversal ini akan dilakukan terus menerus sampai akhirnya solusi untuk mengambil bola dan mencapai pintu keluar ditemukan. Solusi pertama yang ditemukan berarti merupakan solusi dengan langkah terpendek. Solusi adalah ketika keadaan kotak memiliki bola di saat memasuki pintu keluar. Langkah yang dilakukan jumlahnya adalah sama dengan kedalaman BFS yang dilakukan. Setelah solusi ditemukan, algoritma BFS dapat dihentikan karena solusi dengan langkah terpendek sudah ditemukan.

Untuk kasus lantai X, walaupun kotak pemain bisa digeser ke arah tersebut, tetapi arah tersebut tidak ditambahkan ke anak simpul karena lantai X akan mengulang *stage* kembali ke keadaan semula ketika *stage* baru dimulai.

Untuk kasus bila kotak pemain digeser ke pintu keluar tetapi masih belum memiliki bola, kotak pemain akan kembali ke posisi semula ketika *stage* baru dimulai, sehingga tidak dibuat anak simpulnya, sama seperti kasus lantai X.



Gambar 3.3 Graf untuk *stage* 2. Solusi dengan langkah terpendek dari Start adalah atas, kiri, bawah, kanan, atas, kanan, atas.

IV. KESIMPULAN

Dengan mengimplementasi algoritma BFS, segala kemungkinan masalah pada permainan *Squarescape* bisa diselesaikan dengan langkah yang terpendek.

VII. ACKNOWLEDGMENT

Kami mengucapkan terima kasih sebesar-besarnya kepada Tuhan Yang Maha Esa, karena atas rahmat-Nya, kami bisa menulis makalah ini. Kami juga mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi atas bimbingannya selama mengajar di kelas IF2211 Strategi Algoritma yang telah memberikan banyak ilmu kepada kami.

REFERENCES

[1] Munir,Rinaldi, BFS and DFS (2015).pptx.
[2] https://play.google.com/store/apps/details?id=com.noodlecake.squarescape&hl=en diakses pada 5 Mei 2015 pukul 03:38.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Mei 2015

Cliff Jonathan
13513044