Algoritma BFS dalam pencarian Jalur Terpendek Bidak Kuda pada Permainan Catur

Leo Lambarita Nadeak - 13515041
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung (ITB)
Bandung, Jawa Barat
lnadeak97@gmail.com

***Abstract*—Kuda menjadi salah satu bidak dalam permainan catur yang memiliki keunikan tersendiri. Dengan keunikan ini bagaimana cara mengetahui jalur terpendek bidak kuda tersebut di atas papan catur. Hal ini dapat diselesakan dengan banyak algoritma pencari jalur yang umum dipakai, salah satunya adalah algoritma BFS.**

Keywords—Catur, Bidak Kuda, Algoritma, BFS.

I. PENDAHULUAN

Dalam permainan catur, terdapat beberapa bidak, yang setiap bidaknya berbeda cara melangkahnya. Contohnya pion, hanya bisa melangkah ke depan sebanyak satu kotak, kuda yang bentuk langkahnya seperti huruf L, dan bidak – bidak lain.

Bidak catur yang memiliki cara melangkah paling unik adalah kuda, bentuk langkahnya menyerupai huruf L, dengan kata lain tidak akan pernah lurus. Dengan cara langkah seperti ini ada hal yang cukup menarik untuk dipermasalahkan. Bagaimana menemuka lintasan terpendek bidak kuda pada papan catur? Jika dihadapkan dengan objek lain, lintasan terpendek ini merupakan hal yang telah umum, namun untuk bidak kuda, lintasan terpendek menadi sebuah permasalahan yang lumayan rumit, dikarenakan beberapa hal, seperti jumlah kotak papan catur yang dilaluinya konstan, atau bentuk langkahnya yang sudah tetap, ataupun karena hal lainnya.

Dalam makalah ini, akan dibahas bagaimana menacari lintasan terpendek dari posisi awal bidak kuda, menuju kotak lain menggunakan algoritma yang umum digunakan dalam permasalhan lintasan, salah satunya adalah Algoritma *Breadth First Search* (BFS). Selain menjabarkan bagaimana algoritmanya, akan dilihat seberapa optimal algoritma tersebut dgunakan dalam menemukan lintasan terpendek bidak kuda di atas papan catur.

II. DASAR TEORI

2.1 Permainan Catur

Catur adalah salah satu contoh *board games* yang dimainkan oleh dua orang yang saling berlawanan. Kedua pemain direpresentasikan dengan dua warna, yaitu warna hitam dan warna putih.

Tujuan dari permainan catur ini adalah "memakan" bidak pemain lain, terutama bidak raja dari lawan. Memakan yang dimaksud adalah memindahkan bidak sendiri ke posisi bidak lawan sehingga bidak tersebut menimpa bidak lawan, dan bidak lawan harus keluar dari papan catur.

Gambar 1. Posisi awal keseluruhan bidak dalam permainan catur
Sumber : http://www.burung-net.com diakses pada 13 Mei 2017

Terdapat beberapa bidak dalam permainan catur, pion, benteng, kuda, gajah, menteri, dan raja untuk masing – masing warna hitam dan putih. Setiap bidak memiliki cara melangkah yang berbeda – beda, sehingga setiap bidak tidak bisa asal memakan bidak lain.

Tabel 1. Daftar bidak yang ada pada bidak catur

Bidak	Jumlah	Lambang
Raja	1	
Menteri	1	
Gajah	2	

Kuda	2	
Benteng	2	
Pion	8	

(Gambar lambang disadur dari : https://id.wikipedia.org/wiki/Papan_catur diakses pada 13 Mei 2017)

Dalam pergerakan semua bidak catur, bidak tidak boleh berpindah ke kotak yang telah ditempati bidak lain yang sewarna dengan bidak tersebut, namun diperblehkan ika di kotak tersebut terdapat bidak lain dengan warna berbeda, dengan kata lain proses "memakan".

Untuk mengidentifikasi lokasi dari bidak catur, digunakan sebuah kode yang menunjukkan letak baris dan kolom bidak catur berada. Jika bidak berada pada kolom pertama maka dinyatakan sebagai a, dan berada pada baris ketiga maka dinyatakan sebagai 3. Sehingga posisi bidak tersebut adalah a3.

2.2 Bidak Kuda

Bidak kuda adalah salah satu bidak yang digunakan dalam permainan catur. Dalam posisi awal permainan, kuda diletakkan pada posisi b1, g1, b8, dan g8, yang mana b1 dan g1 merupakan lokasi kuda dengan warna sama, dan b8 dan g1 merupakan lokasi kuda dengan warna berbeda.

Kuda memiliki cara berpindah yang cukup unik dibandingkan dengan bidak lain. Kuda bergerak dua petak ke satu arah, lalu bergerak satu kotak lagi kearah sudut 90^o, seperti bentuk L. Dengan bentuk langkah seperti ini, kuda diperbolehkan melompati bidak lain yang berada pada jalur langkah kuda tersebut, berbeda dengan bidak lain yang tidak bisa melompati bidak yang berada pada jalur perpindahan bidak tersebut.

Gambar 2. Bentuk langkah dari bidak kuda pada permainan catur
Sumber : http://www.grosircatur.com dakses pada 13 Mei 2017

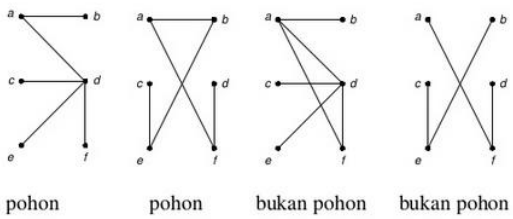
2.3. Graf dan Pohon

Graf merupakan sebuah konsep yang digunakan untuk merepresentasikan hubungan antara objek – objek. Representasi tersebut disimbolkan dalam bentuk simpul dan sisi. Yang mana simpul biasanya digambarkan dengan lingkaran menyatakan objek dan sisi biasanya digambarkan dengan garis menyatakan hubungan dari objek – objek tersebut. Secara matematis graf dinyatakan sebagai G = (V,E) denga V adalah himpunan tidak kosong dari simpul dan E adalah himpunan tidak kosong dari sisi[2].

Salah satu bentuk graf yang tidak mengandung sirkuit didalamnya adalah pohon, yang dalam artian tidak ada hubungan yang kembali ke objek tersebut setelah berhubungan dengan objek – objek lain.

Misalkan G = (V,E) adalah graf sederhana dengan jumlah simpul n, G disebut graf apabila[2]:

- Setiap pasang simpul G terhubung dengan lintasan tunggal.
- G terhubung dan memiliki n – 1 buah sisi.
- G tidak memiliki sirkuit
- Jika G ditambah dengan 1 sisi maka akan terbentuk sebuah sirkuit
- G terhubung dan semua sisinya adalah jembatan



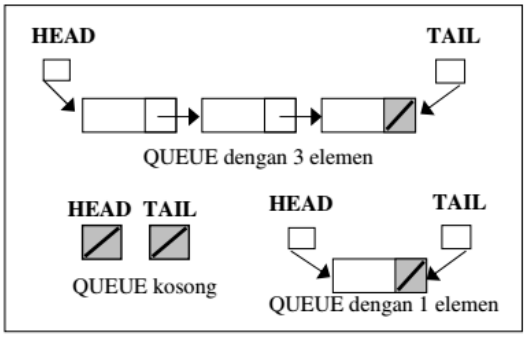
Gambar 3. Contoh graf yang merupakan pohon dan yang tidak merupakan pohon

Sumber : https://www.slideshare.net/KuliahKita/matematika-diskrit-10-pohon-01 diakses pada 15 Mei 2017

2.4. Struktur Data Queue

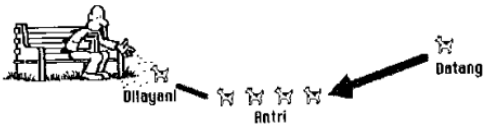
Queue adalah salah satu struktur data berupa sebuah list linier yang[3]:

- Diketahui elemen pertama (*head)* dan elemen terakhirnya *(tail)*.
- Data yang ingin ditambahkan selalu ditambahkan setelah elemen terakhir
- Data yang ingin dihapus selalu dilakukan pada elemen pertama.
- Elemen selanjutnya dapat diakses dengan "*Next"*, sama seperti list linier pada umumnya.



Gambar 4. Struktur Lojik Queue
Sumber : [3]

Dalam struktur data queue, data yang pertama masuklah data yang akan diproses, sama seperti bagaimana orang – orang antri pada umumnya. Data yang belum diproses akan diantrikan di belakang data yang lebih dulu masuk.



Gambar 5. Gambaran queue/antrian pada dunia nyata.
Sumber : Draft Struktur Data

2.5 Algoritma Breadth First Search (BFS)

Algoritma BFS merupakan salahs atu algoritma yang digunakan untuk menyelusuri objek – objek pada graf secara traversal. Karakteristik utama dari algoritma BFS adalah menyelusuri seluruh simpul yang bertetangga dengan simpul tersebut, jika sudah semua dan tujuan belum dicapai kemudian simpul lain dikunjungi, dalam artian penelusuran dilakukan secara melebar.

Secara umum, algoritma dari BFS adalah :

- Traversal dimulai dari simpul v, yang mana v adalah simpul yang dipilih.
- Traversal dilanjutkan dengan mengunjungi semua simpul yang bertetangga dengan simpul v.
- Traversal dilanjutkan lagi dengan mengunjungi simpul yang belum dikunjungi dan simpul – simpul yang bertetangga dengan simpul tersebut, dan lakukan seterusnya hingga tujuan dicapai atau semua simpul sudah dikunjungi.

Untuk memastikan setiap tetangga dari simpul awal dikunjungi pula tetangga – tetangganya, maka setiap simpul yang dikunjungi akan disimpan dalam sebuah queue. Setelah suatu simpul selesai diproses, setiap tetangganya telah dikunjungi, maka proses dilanjutkan dengan memproses simpul pada *head* queue. Berikut *pseudocodedari* algoritma BFS[1].

```
Procedure BFS (input v : integer)
{v adalah simpul awal yang dikunjungi}
Deklarasi
   W : integer
   q : queue;

   procedure BuatAntrian(input/ouput q
   : queue)
   {Membuat antrian kosong}

   procedure MasukAntrian(input/output
   q : queue, input v : integer)
   {Menambahkan v kedalam queue}

   procedure HapusAntrian(input/output
   q : queue, output v : integer)
   {Menghapus data dari queue dan
   mengembalikan nilainya sebagai v}

   Function AntrianKosong(q : queue) :
   boolean
   {Menghasilkan true jika queue kosong
   dan false jika queue tidak kosong}

Algoritma :
   BuatAntrian(q)
   output(v)
   dikunjungi[v] ← true
   {Nyatakan simpul ke-v sudah
   dikunjungi}
   MasukAntrian(q,v)
   {travelsal ke tiap tetangga}
   while not AntrianKosong(q) do
```

```
      HapusAntrian(q,v)
      for tiap simpul w tetangga dari v
      do
         if not dikunjungi[w] then
            output(w)
            MasukAntrian(q,w)
            dikunjungi[w] ← true
         endif
      endfor
   endwhile
```

III. BATASAN PERMASALAHAN

Pada permasalahan yang sudah dijelaskan sebelumnya diberikan batasan – batasan unutk mempermudah pembentukan rancangan algoritma penyelesaian masalah. Untuk menyelesaikan permasalahan yang umum (di luar batasan – batasan yang diberikan) dapat diselesaikan di kemudian hari jika memungkinkan.

Batasan yang dibeikan adalah sebagai berikut.

- Di atas papan catur hanya terdapat satu bidak kuda. Sehingga tidak ada kemungkinan ada bidak lain berwarna sama pada posisi tertentu membuat bidak kuda tidak dapat berpindah ke posisi tersebut.
- Papan catur sudah pasti berukuran 8 x 8 kotak.
- Permasalahan mencakup node posisi awal dari bidak kuda dan node tujuan yang ingin dicapai kuda. Dan hasil yang diberikana adalah satu atau lebih jalur terpendek yang dapa dilakukan oleh bidak kuda.

IV. PEMBAHASAN

Untuk mengimplementasikan algoritma penyelesaian masalah, ada beberapa hal yang harus diperhatikan, yaitu sebagai berikut.

- Posisi dari kuda dinyatakan sebagai node, yang merupakan tipe bentukan dan terdiri dari baris dan kolom dimana kuda itu berada.
- Arah dinyatakan sebagai integer, yaitu 1 untuk atas, 2 untuk kanan, 3 untuk bawah, dan 4 untuk kiri.
- Jika posisi bidak setelah melakukan perpindahan ternyata diluar papan catur, maka posisi hasil akan diubah menjadi 00.

3.1 Algoritma Langkah Kuda

Untuk menjalankan kuda dapat dilakukan dengan cara memindahkan kuda ke kanan, atas, bawah, atau kiri (lurus) sejauh 2 kotak kemudian memindahkannya 1 kotak 90^o dari arah pergerakan awalnya. Hal ini dapat dilakukan dengan menambahkan atau mengurangi baris atau kolom pada posisi kuda sebanyak 2 satuan kemudian menambahkan atau mengurangi sebanyak 1 kotak pada baris (jika sebelumnya yang diubah adalah kolom) atau kolom (jika sebelumnya yang diubah adalah baris). Algoritma tersebut dalam pseudocode berikut.

```
function moveKuda(node Awal, integer
arah) : node
{Mengembalikan posisi akhir dari kuda
setelah melakukan perpindahan ke arah
yang diinginkan.
Mengembalikan 00 jika langkah tidak bisa
dilakukan.}

Algoritma :
if arah = 2 then
   tambahkan kolom 2 kotak
   tambahkan/kurangkan baris 1 kotak
elseif arah = 3 then
   kurangi baris 2 kotak
   tambahkan/kurangkan kolom 1 kotak
elseif arah = 4 then
   kurangi kolom 2 kotak
   tambahkan/kurangkan baris 1 kotak
else
   tambahkan baris 2 kotak
   tambahkan/kurangkan kolom 1 kotak

if not posisi masih dalam papan then
   node.baris ← 0
   node.kolom ← 0

→ node
```

3.2 Mengambil seluruh node tetangga dari node yang diproses

Untuk mendapatkan seluruh tetangga dari node yang sedang diproses dapat dilakukan dengan cara memindahkan kuda ke seluruh arah, kemudian mengecek apakah hasil perpindahan masih benar, dalam artian masih di dalam papan catur atau tidak menimpa bidak yang sewarna. Dengan algoritma ini akan ditemukan maksimal 8 node tetangga yang mungkin. Nilai maksimal ini adapa dicapai jika hasil perpindahan tidak di luar papan catur atau tidak ada bidak dengan warna sama pada posisi hasil perpindahan kuda tersebut (hal ini ttidak dipertimbangkan sesuai dengan batasan masalah yang digunakan).

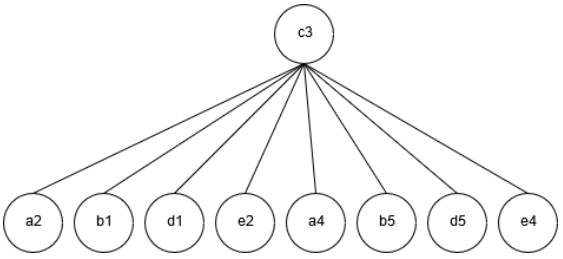
Algoritma tersebut dapat dituliskan dalam *pseudocode* berikut.

```
function getTetangga(awal : node) : array [0..7] of node
{Menyimpan seluruh tetangga node dari node awal yang benar}

Deklarasi :
   NHasil : array [0..7] of node
   i, j : integer

Algoritma :
   {isi NHasil dengan 00 = Deklarasi}
   {Pindahkan kuda dari posisi awal ke delapam arah yang mungkin}
   i = 0
   while (i < 8)
      for j = 1 to 4 do
        NHasil[i] = moveKuda(awal, j)
        {j merupakan representasi arah}
        if NHasil[i] = 00 then
           NHasil[i] = 00
        else
           i ← i + 1
        {Lakukan moveKuda berbeda sekali lagi, karena perpindahan di satu arah, bisa ada dua hasil, 90 derajat ke kiri atau 90 derajat ke kanan arah}
        NHasil[i] = moveKuda(awal, j)
        if NHasil[i] = 00 then
           NHasil[i] = 00
        else
           i ← i + 1
   → NHasil
```

Gambar 6. Delapan kemungkinan perpindahan bidak kuda
Sumber : http://www.grosircatur.com dakses pada 17 Mei 2017

Node – node tetangga yang diperoleh merupakan simpul – simpul tetangga dari node yang diproses, yang dalam artian level dari node – node tersebut merupakan level node yang diproses ditambahkan satu. Misalkan node yang diproses adalah node awal/pertama posisi kuda berada, maka tetangga yang diperoleh dengan fungsi getTetangga merupakan simpul – simpul pohon dengan level 2 dan tetangga dari node pertama tersebut.

Setelah node – node tetangga ditemukan, node – node tersebut haru dimuat dalam struktur data pohon yang nantinya akan digunakan dalam pencarian BFS.



Gambar 7. Representasi perpindahan kuda pada gambar 6 menjadi sebuah pohon.

3.3 Algoritma BFS

Untuk menyelesaikan permasalahan, implementasi algoritma BFS yang digunakan tidak jauh berbeda dengan algoritma BFS pada umumnya. Namun terdapat sedikit perubahan untuk penyesuaian dengan algoritma – algoritma sebelumnya.

```
Procedure BFScatur (input v : node, input w : node)
{v adalah simpul awal yang dikunjungi, w adalah simpul tujuan yang ingin dikunjungi}
Deklarasi
   q : queue;
   t : array [0..7] of node
   {t merupakan array dari tetangga dari sebuah simpul yang diproses}
   i : integer

   procedure BuatAntrian(input/ouput q : queue)
   {Membuat antrian kosong}

   procedure MasukAntrian(input/output q : queue, input v : integer)
   {Menambahkan v kedalam queue}

   procedure HapusAntrian(input/output q : queue, output v : integer)
   {Menghapus data dari queue dan mengembalikan nilainya sebagai v}

   Function AntrianKosong(q : queue) : boolean
```

```
    {Menghasilkan true jika queue kosong
    dan false jika queue tidak kosong}

Algoritma :
    BuatAntrian(q)
    output(v)
    dikunjungi[node.baris][node.kolom]
    ← true
    {Nyatakan kotak pada posisi baris-
    kolom sudah dikunjungi}
    MasukAntrian(q,v)
    {travelsal ke tiap tetangga}
    while not AntrianKosong(q) or
    ditemukan node w pada level tersebut
    do
       HapusAntrian(q,v)
       t = getTetangga(v)
       for tiap anggota array t dengan
       indeks i do
          if not dikunjungi[t[i].baris]
          [t[i].kolom] then
             output(t[i])
             MasukAntrian(q,t[i])
             dikunjungi[t[i].baris]
             [t[i].kolom] ← true
          endif
       endfor
    endwhile
```

Algoritma tersebut akan berhenti jika ditemukan node tujuan pada level tertentu, atau semua jalur sudah dikunjungi satu persatu. Algoritma yang berhenti ketika menemukan tujuan pada level tertentu, membuat algoritma ini cukup efisien, karena sistem tidak perlu lagi mencari ke seluruh jalur yang mungkin ada.

Ketika sudah mencapai level yang memuat minimal satu node tujuan, sistem/program akan berakhir. Namun sistem tidak langusng dihentikan ketika sudah menemukan node tujuannya. Semua node pada level tersebut harus dikunjungi terlebuh dahulu kemudian berhenti, karena ada kemungkinan ada lebih dari satu jalur untuk mencapai node tujuan pada level poshon yang sama.

V. KESIMPULAN

Kuda merupakan salah satu bidak catur yang unik, dikarenakan bentuk langkahnya yang menyerupai huruf “L” dan dapat melompati bidak yang ada pada jalur langkahnya. Dengan metode langkah yang unik tersebut, muncul permasalahan bagaimana menemukan lintasan terpendek dari bidak tersebut.

Permasalahan dapat diselesaikan dengan algoritma *Breadth First Search* (BFS). Dan algoritma ini sangat cocok dengan permasalahan tersebut, karena dalam proses pencariannya, tidak perlu mencari hingga ke seluruh jalur yang mungkin dilalui oleh Kuda, hanya perlu mencari node posisi kuda pada level tertentu dimana posisi tujuan sudah ditemukan. Namun tetap saja ada kemungkinan *worst case*, dimana pencarian dilakukan ke seluruh jalur yang mungkin dilalui oleh Kuda.

VI. SARAN

Untuk memperoleh hasil yang lebih baik, permasalahan dapat diperluas dengan mempertimbangkan batasan – batasan yang diberikan pada makalah ini. Sehingga algoritma dapat digunakan dalam permasalahan langkah kuda pada umumnya.

Algoritma dapat dirancang dengan mempertimbangkan adanya bidak lain yang mungkin dapat menghalangi langkah kuda. Sistem dapat juga dibuat mampu menerima inputan user berupa ukuran papan catur,s eprti papan catur dengan ukuran 4 x 8 (ukuran yang sering digunakan dalam permainan halma) dan semacamnya.

V. REFERENSI

[1] Munir, Rinaldi. 2009.*Diktat Kuliah IF2211 Strategi Algoritma*. Bandung : Penerbit Informatika.

[2] Munir, Rinaldi.2006.*Diktat Kuliah IF2120 Matematika Diskrit*. Bandung : Penerbit Informatika.

[3] Liem, Inggriani.2008.*Draft Struktur Data*. Bandung : Penerbit Infomatika.

[4] https://www.chess.com/id/learn-how-to-play-chess, diakses pada tanggal 13 Mei 2017 pukul 14.55

[5] Saputra, Amri.2011.*Rancang Bangun Simulasi Langkah Kuda dalam Papan Catur*.Laporan Tugas Akhir.Jurusan Teknik Informatika Universitas Islam Negeri Sultan Syarif Kasim Riau.

VI. 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, 19 Mei 2017

Leo Lambarita Nadeak
13515041