Penerapan Algoritma *Brute force* dalam PermainanTradisional Congklak

Christian 120140056[1]
Program Studi Teknik Informatika Jurusan
Teknik Elektro Informatika dan Sistem Fisis
Institut Teknologi Sumatera, Jl. Terusan Ryacudu Lampung 35365, Indonesia
[1]christian.120140056@student.itera.ac.id

***Abstrak*—Sebagai ciri khas bangsa, Indonesia dikenal sebagai negara yang beragam budayanya. Hal ini mengakibatkan banyaknya permainan tradisional yang Indonesia miliki, salah satunya adalah congklak. Permainan ini memiliki tujuan untuk mendapatkan biji sebanyak-banyaknya dengan cara yang diperbolehkan dalam permainan. Dalam makalah ini akan dijelaskan mengenai penerapan algoritma *brute force* untuk mendapatkan hasil yang optimal dengan harapan dapat memenangkan permainan. Selain itu, makalah ini juga dapat menjadi acuan dalam membangun *artificial intelligence* yang mampu mempertimbangkan dan memilih pilihan yang paling optimal.**

Kata kunci—permainan tradisional; congklak; optimal; algoritma brute force.

I. PENDAHULUAN

Seiring berjalannya waktu, perkembangan teknologi dan budaya di dunia semakin cepat. Budaya budaya lama atau tradisional mulai ditinggalkan karena sering dianggap tidak relevan dan tidak praktis oleh masyarakat. Hal ini tanpa terkecuali, juga berlaku untuk permainan permainan tradisional yang ada di Indonesia. Anak-anak di Indonesia, terutama yang tinggal di daerah perkotaan, sudah tidak lagi mengenal permainan tradisional Indonesia, padahal banyak permainan tradisional Indonesia yang seru, melatih reflek motorik, serta mendidik.

Sebut saja permainan papan tradisional, congklak. Permainan congklak dapat meningkatkan kemampuan berpikir pemain. Strategi diperlukan dalam permainan congklak untuk mengalahkan lawan. Dalam setiap putaran terdapat beberapa kemungkinan opsi yang dapat dieksekusi. Pemain harus membuat keputusan dari opsi yang tersedia, sehingga algoritma *brute force* dianggap cocok untuk masalah ini.

Algoritma *brute force* dapat digunakan untuk mencari solusi optimal dengan bayaran waktu dan tenaga yang banyak untuk melihat semua solusi yang mungkin. Dengan menggunakan algoritma *brute force*, pemain dipastikan dapat memilih opsi yang tepat setiap giliran untuk memenangkan permainan. Tentu saja, opsi ini mungkin bukan opsi terbaik ke depannya. Namun, algoritma ini cocok untuk sebuah *game* karena dapat memastikan bahwa langkah yang diambil merupakan yang terbaik untuk mencapai tujuan yaitu kemenangan.

II. DASAR TEORI

A. Congklak

Congklak merupakan salah satu permainan tradisional di Indonesia. Ada beberapa nama daerah yang terkait dengan permainan ini, misalnya Dakon di Jawa, Congkak di Sumatera, Makaotan di Sulawesi [1]. Seiring berjalannya waktu, permainan modern mulai bermunculan. , dari mana permainan tradisional muncul, semacam congklak sedikit demi sedikit ditinggalkan.

Congklak adalah permainan untuk dua pemain. Papan permainan dengan empat belas lubang kecil dan dua lubang besar juga diperlukan. Lubang besar atau lumbung atau Gudang ini berada di ujung papan, sedangkan lubang kecil dibagi dua dan saling berhadapan sehingga masing-masing sisi ada tujuh [2]. Berikut adalah foto papan Congklak [3].

Gambar 1 Papan permainan congklak

Setiap lubang samping harus diisi tujuh biji. Jumlah biji yang dibutuhkan sebanyak 98 biji. Biji yang biasa digunakan adalah biji batu kecil, cangkang atau biji buah.

Permainan dimulai dengan menentukan pemain mana yang lebih dulu.Pemain pertama memilih salah satu lubang di sisinya. Biji dikeluarkan dari lubang kemudian dimasukkan satu per satu ke lubang berikutnya searah jarum jam. Semua lubang kecuali lubang utama lawan terisi. Jika benih terakhir jatuh ke dalam lubang yang ada benih lain di dalamnya, maka benih yang ada di lubang tersebut diambil kembali untuk melanjutkan pengisian lubang berikutnya.

Jika biji terakhir menjadi lubang utama yang sama, pemain dapat melanjutkan gerakan dimulai dengan salah satu lubang di sisinya. Ketika unggulan terakhir berada di lubang samping kosong lawan, giliran mereka berakhir.Jika biji terakhir mendarat di lubang kosong di sisi Anda, putaran saat ini berakhir, tetapi biji apa pun di lubang lawan yang menjadi pasangan lubang terakhir dapat ditempatkan di lubang utama pemain [2].

Setelah semua lubang kosong, permainan berakhir. Pemenang ditentukan dengan menghitung jumlah biji di setiap lubang utama. Pemain yang mengumpulkan biji paling banyak adalah pemenangnya.

B. Algoritma Brute force

Algoritma *brute force* adalah algoritma yang dapat menghasilkan solusi yang selalu optimal terhadap suatu permasalahan dengan cara mencoba semua kemungkinan solusi yang ada. Algoritma ini selalu dijadikan dasar untuk mendapatkan solusi optimal meskipun tidak se-efisien algoritma lain. Hampir semua permasalahan dapat diselesaikan menggunakan algoritma *brute force* termasuk dalam menemukan langkah terbaik dalam permainan congklak.

Algoritma *brute force* sering disebut sebagai algoritma naif karena mengecek semua solusi yang ada untuk memilih solusi optimal dan memiliki waktu yang sebanding dengan langkah atau pengecekan solusi yang dilakukan.

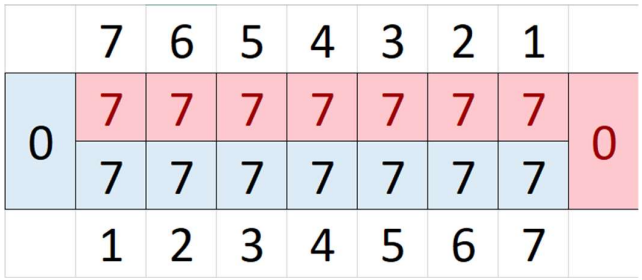
Contoh Algoritme *Brute force* dalam algoritma searching normal dalam bahasa python:

```
#python code
def max(list): #list berisi angka integer
        temp = 0
        for i in range(len(list)):
                if i == 0:
                        temp = list[i]
                elif list[i] > temp:
                        temp = list[i]
        return temp
```

dari contoh tersebut bisa diketahui bahwa semakin banyak isi dari *array* x dan semakin besar *index* posisi *value* didalam *array* maka waktu komputasinya juga akan semakin lama. Sudah jadi sifat dari algoritma *brute force* ini untuk mencari solusi dari suatu permasalahan dengan cara yang tidak efisien karena mengecek semua kemungkinan, namun ada beberapa persoalan yang hanya bisa diselesaikan dengan algoritma ini.

III. PEMBAHASAN

A. Gambaran Model



Gambar 2 Model congklak

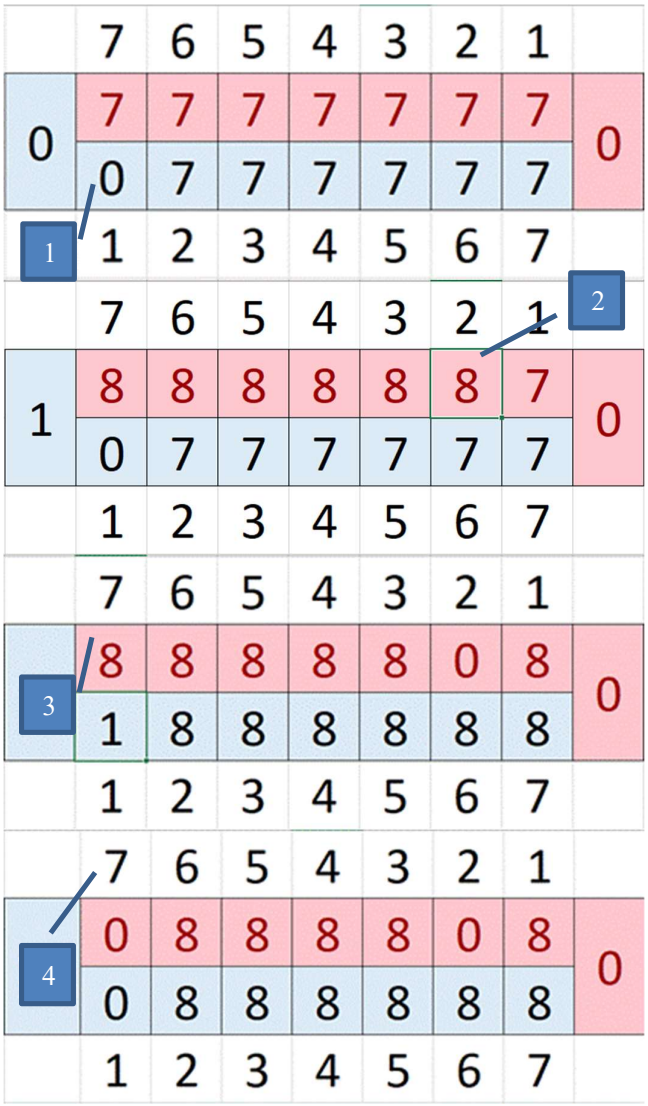
Gambar di atas adalah sebuah contoh penggambaran kondisi permainan congklak yang akan digunakan dalam penjelesan kedepannya. Gambar di atas menunjukan kondisi awal permainan, yang mana kotak berangka 7 adalah lubang pemain sedangkan yang berangka 0 adalah lumbung pemain. Lalu, dari pewarnaan, warna merah menggambarkan lubang lawan dan warna biru menggambarkan lubang pemain.

Struktur data *list* atau *array* juga dapat digunakan untuk menggambarkan permainan congklak. Dengan list sepanjang 15 dimana list indeks ke-0 adalah lumbung, list indeks ke-1 sampai ke-7 adalah lubang pemain, dan list indeks ke-8 sampai ke-14 adalah lubang lawan. Untuk lumbung lawan tidak perlu digambarkan karena tidak akan dipakai oleh pemain.

Tabel 1 Visualisasi evaluasi lubang

Lubang	Penambahan	Lumbung
1		
2		
3		
4		
5		
6		
7		
	Best Move = 1	Max = 0

Tabel di atas digunakan untuk memudahkan penjelasan dan pencatatan setiap hasil dari langkah yang diambil agar dapat menentukan langkah yang terbaik berdasarkan total biji yang dapat dimasukan ke lumbung. Variabel *best move* digunakan untuk mencatat langkah terbaik sejauh ini, dan *max* adalah jumlah biji terbanyak yang dapat dihasilkan sejauh ini.



Gambar 3 Kondisi congklak setelah mengambil langkah lubang 1

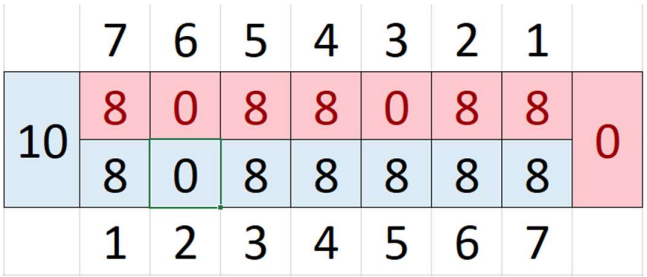
Saat mengambil lubang 1, kita mendistribusikan 7 biji ke lumbung kita sampai ke lubang 2 milik lawan. Selanjutnya kita mengambil 8 biji di lubang 2 tadi dan mendistribusikannya hingga kembali ke lubang nomor 1 milik kita. Karena kita

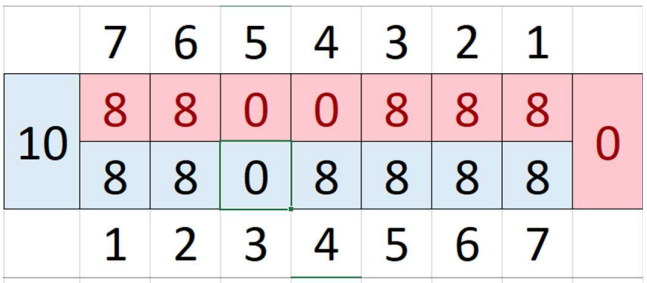
berhenti dilubang yang kosong, sesuai peraturan maka kita akan mengambil semua biji yang ada dilubang yang berseberangan yaitu lubang nomor 7 milik lawan dan 1 biji di lubang nomor 1 tadi. Maka hasil total dari giliran ini adalah 10 biji dan berikut adalah evaluasinya dalam tabel.

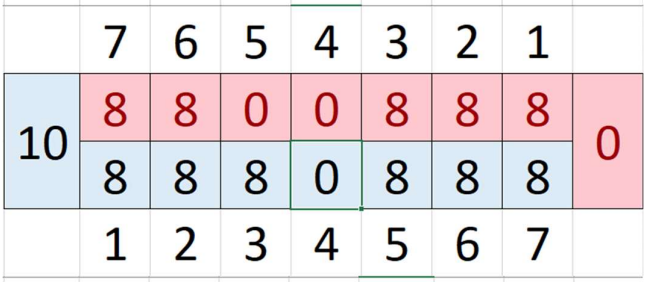
Tabel 2 Evaluasi setelah mengambil langkah lubang 1

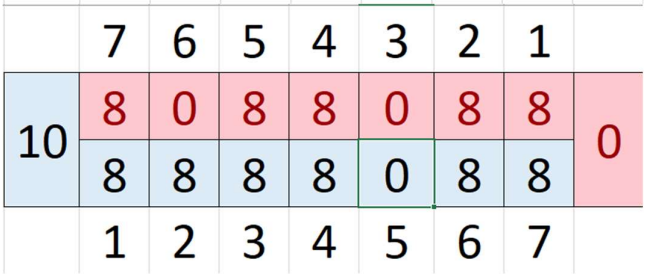
Lubang	Penambahan	Lumbung
1	1+9	10
2		
3		
4		
5		
6		
7		
	Best Move = 1	Max = 10

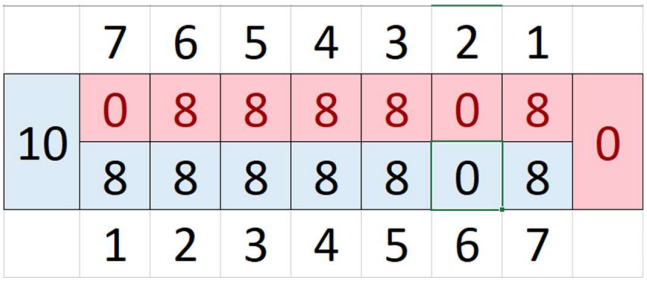
Dengan cara yang sama, kita dapat menentukan hasil dari lubang berikutnya yang dapat dijalankan dan mendapatkan hasil berikut.









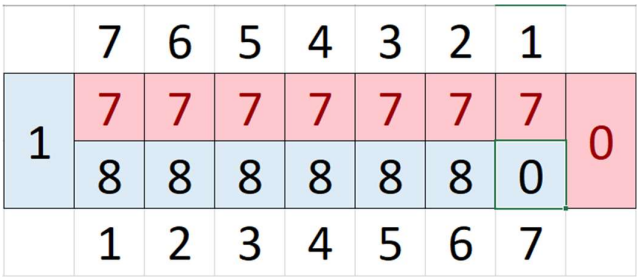


Gambar 4 Kondisi congklak setelah mengambil langkah lubang 2 sampai 6

Tabel 3 Evaluasi setelah mengambil langkah lubang 2 sampai 6

Lubang	Penambahan	Lumbung
1	1+9	10
2	1+9	10
3	1+9	10
4	1+9	10
5	1+9	10
6	1+9	10
7		
	Best Move = 1	Max = 10

Karena hasil dari lubang nomor 2 sama seperti hasil dari lubang nomor 1, maka kita tidak perlu mengganti best move untuk saat ini. Dan karena hasil dari lubang nomor 3 sampai 6 juga menghasilkan hal yang sama, maka tidak ada yang berubah untuk saat ini.



Gambar 5 Kondisi congklak setelah mengambil langkah lubang 7

Dari gambar diatas, langkah lubang 7 berakhir di lumbung. Sesuai peraturan, maka pemain akan mendapat giliran kedua untuk memilih lubang kembali.

Tabel 4 Evaluasi setelah mengambil langkah lubang 7

Lubang	Penambahan	Lumbung
1	1+9	10
2	1+9	10
3	1+9	10
4	1+9	10
5	1+9	10
6	1+9	10
7	1	1
7-1		

7-2		
7-3		
7-4		
7-5		
7-6		
7-7	-	-
	Best Move = 1	Max = 10

Lubang 7-1 sampai 7-7 adalah lubang yang diambil setelah mengambil lubang 7 terlebih dahulu. Hal ini merupakan langkah algoritma *brute force* untuk mencari langkah terbaik dengan mencari semua kemungkinan langkah yang ada. Lubang 7-7 tidak dapat dijalankan karena tidak ada biji didalamnya sehingga diberikan tanda "-" pada penambahan biji jika lubang itu dijalankan.

Jika kita mencoba mencarinya secara manual menggunakan tenaga manusia, tentu saja hal ini kurang memungkinkan karena jumlahnya yang bisa mencapai ribuan solusi. Oleh karena itu, kita akan berikan tugas mencari solusi terbaik ini kepada program komputer saja.

B. Algoritma Evaluasi Langkah

Algoritma ini memiliki tujuan untuk mencari nilai terbesar yang bisa dimasukan ke dalam lumbung jika langkah dimulai dari lubang ke-n. Nilai dari n berkisar dari 1 sampai 7 yang merepresentasikan jumlah lubang kecil yang ada di sisi pemain. Keluaran atau hasil yang diharapkan adalah sebuah integer yang lebih besar atau sama dengan 0.

Berikut adalah algoritmanya:

```
procedure Eval ( input n : integer, congklak: integer[15], output akhir: integer, jalur: string )
{menghitung jumlah akhir lubang induk giliran itu jika langkah dimulai dari lubang sisi ke-n
Masukan: n (salah satu indeks lubang sisi , 1-7)
Keluaran: isi lubang induk akhir
}

Deklarasi
       stop, i , idx, m, temp: integer
       List2 : integer[15]
       S1, S2, S3, S4, S5, S6, S7: string
       a, b, c, d, e, f, g: string
Algoritma
       stop ← 0
       idx ← 1
       m ← n
       List ← congklak
       Jalur←""
       while (stop =0 and n><0)do
              List[n] ← 0
              for i←0 to List[n]-1 do
                     temp ← (n+1+i) mod 15
                     List[temp]++
                     idx ← temp
              endfor

              if (List[idx]=1 and idx>7 ) then
{berhenti di kosong dan di sisi lawan}
                     Stop←1
              else if (List[idx]=1 and 0<idx<8 ) then
{berhenti di kosong dan di sisi sendiri}
                     List[0] ←List[0]+List[15-idx]
                     List[15-idx] ←0;
              else if (List[idx]><1 ) then
{berhenti di tidak kosong}
                     n=idx
              else {berhenti di lubang induk}
                     List2←List
                     Eval(1, List2, a, S1)
                     List2←List
                     Eval(7, List2, b, S2)
                     List2←List
                     Eval(6, List2, c, S3)
                     List2←List
                     Eval(5, List2, d, S4)
                     List2←List
                     Eval(4, List2, e, S5)
                     List2←List
                     Eval(3, List2, f, S6)
                     List2←List
                     Eval(2, List2, g, S7)

                     Max(a,b,c,d,e,f,g)
  {prosedur max akan mencari jalur mana yang akan mencari nilai terbesar antara a-g. jika yang terbesar c maka List[0] akan ditambah c, dan jalur akan ditambah dengan S3}
                     endif endwhile
                     akhir←List[0]
```

C. Algoritma Brute force

Algoritma ini akan memanggil algoritma evaluasi yang sebelumnya untuk melihat semua solusi yang ada dari lubang ke-1 sampai lubang ke-7. Jika hasil yang dikembalikan lebih besar dari yang sebelumnya, maka kandidat solusi (*best move*) dan jumlah terbanyak (*max*) akan diperbaharui atau update ke kandidat saat ini.

Berikut adalah algoritmanya.

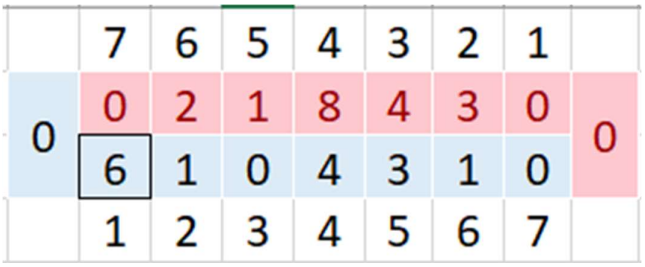
```
procedure ES( )
{menghasilkan jalur yang menghasilkan lubang induk dengan isi terbanyak
List telah diketahui.
}

Deklarasi
       i, akhir, max: integer
       s,l: string
Algoritma
       max←0
       l←1
       for i←1 to 7 do
              s←integerToString(i)
              Eval(i,List, akhir, s)
              if(akhir>max)then
                     max←akhir
                     l←s
              endif
       endfor
       output (lubang maksimal: s)
```

IV. APLIKASI PENGGUNAAN ALGORITMA *BRUTE FORCE*

A. Contoh Penerapan Algoritma

Berikut adalah contoh dari penerapan algoritma yang kita buat pada sebuah contoh kasus.



Gambar 6 Contoh Kasus

Dari contoh diatas (lumbung diisi 0 untuk memudahkan mencari hasil setiap langkah) jika diasumsikan giliran saat ini adalah pemain biru, maka tabel yang akan kita dapat dari penggunaan algoritma *brute force* adalah sebagai berikut.

Tabel 5 Hasil pencarian solusi menggunakan algoritma *brute force*

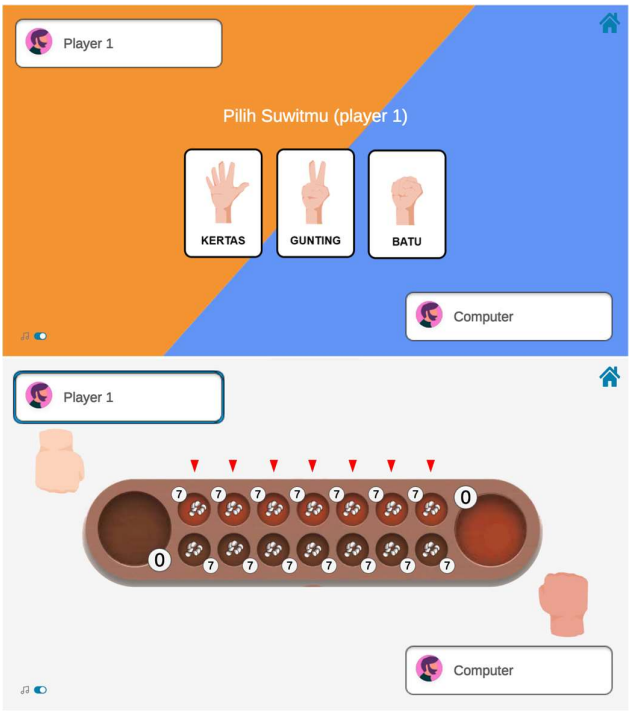
Lubang	Penambahan	Lumbung
1	1+2	3
2	1+2	3
3	-	-
4	1	1
4-1	1+3	4
4-2	1+1	2
4-2-1	2+1	3
4-2-2	-	-
4-2-3	2+3	5
4-2-4	-	-
4-2-5	2+3	5
4-2-6	2+1+2	5
4-2-7	-	-
4-3	1+1	2
4-4	-	-
4-5	1+1	2
4-6	1+1	2
4-7	-	-
5	1	1
5-1	1+1+6	8
5-2	-	-
5-3	1+3	4
5-4	1+1	2
5-5	-	-
5-6	1+5	6
5-7	-	-
6	1+6	7
7	-	-
	Best Move = 5-1	Max = 8

Dari tabel diatas yang merupakan penggambaran dari hasil algoritma *brute force*, dapat disimpulkan bahwa langkah terbaik yang dapat diambil adalah dengan memilih lubang ke-5 terlebih dahulu, yang mana akan memberikan kesempatan giliran lagi, dan memilih lubang ke-1 untuk langkah selanjutnya. Dengan cara ini, jika dibandingkan dengan langkah lainnya yang mungkin, kita dipastikan untuk mendapatkan hasil yang terbaik dan memperbesar kemungkinan memenangkan permainan.

B. Pengembangan Algortima Kedepannya

Seperti yang kita ketahui, masyarakat saat ini pada umumnya cenderung lebih memilih aplikasi *game* pada gadget daripada permainan atau kegiatan fisik, khususnya permainan tradisional. Memang, permainan tradisional dianggap tidak praktis dan konstan. Selain itu, banyak permainan tradisional seperti permainan congklak membutuhkan alat seperti papan yang cukup besar dan minimal 98 biji congklak untuk dimainkan. Jika dibiarkan, hal ini dapat mengikis budaya Indonesia dan membuat generasi mendatang tidak mengetahui dengan permainan congklak ini atau permainan tradisional lainnya.

Untuk mengatasi hal tersebut kita perlu dilakukan inovasi yaitu memadukan unsur tradisional dengan unsur modern. Dalam pembuatan aplikasi *game* congklak ini sebaiknya dilakukan dalam 2 mode yaitu *player* vs *player* dan juga *player* vs komputer. Algoritma *brute force* pada permainan congklak dapat diimplementasikan untuk membuat *artificial intelligence* komputer sebagai lawan dari aplikasi *game* congklak ini. Keuntungan ketika menerapkan algoritma *brute force* dalam *game* ini adalah kemampuan untuk menghitung jumlah biji congklak dan memprediksi langkah-langkah untuk semua pilihan yang dibuat saat bermain di kehidupan nyata.



Gambar 3 Contoh aplikasi permainan congklak online [6]

V. KESIMPULAN

Berdasarkan penjelasan pada makalah ini, dapat disimpulkan bahwa algoritma *brute force* merupakan algoritma yang kurang efisien namun memberikan solusi yang paling optimal untuk membantu dalam pemilihan langkah pada permainan congklak. Permainan congklak juga dapat dikembangkan sebagai dasar untuk menciptakan *artificial intelligence* dalam permainan congklak pada *gadget*. Diharapkan jika diadaptasi menjadi aplikasi *smartphone*, minat masyarakat terhadap *game* congklak bisa semakin meningkat..

VIDEO LINK AT YOUTUBE

https://www.youtube.com/watch?v=eAkE7cpf85k&ab_channel=I2OI4OO56Christian

ACKNOWLEDGMENT

Pertama-tama saya ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa yang hanya karena berkat serta hikmat dan penyertaanNyalah saya dapat menyelesaikan makalah ini. Kemudian tidak lupa saya mengucapkan terima kasih kepada dosen-dosen pengampu mata kuliah IF2211 Strategi Algoritma, yaitu Bapak Imam Eko Wicaksono, S.Si., M.Si. yang telah membimbing saya dalam belajar selama 8 minggu di semester ini dan memungkinkan saya untuk membuat makalah ini. Selanjutnya ucapan terima kasih juga ingin saya sampaikan kepada kedua orang tua saya yang telah mendukung perkuliahan saya. Terakhir, saya juga ingin mengucapkan terima kasih kepada teman-teman Teknik Informatika 2020 yang telah memberi dukungan baik secara moral maupun sosial sehingga makalah ini dapat saya selesaikan dan kumpulkan tepat pada waktunya.

REFERENCES

[1] http://www.cplusplus.com/forum/beginner/226573/ . Diakses pada tanggal 25 Maret 2022.

[2] http://*game*duniaanak.blogspot.com/2011/10/cara-bermain-mainan-tradisional.html . Diakses pada tanggal 25 Maret 2022.

[3] https://mbludus.com/main-congklak/ . Diakses pada tanggal 25 Maret 2022.

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/Makalah2017/Makalah-IF2211-2017-132.pdf . Diakses pada tanggal 25 Maret 2022.

[5] https://adoc.pub/queue/algoritma-exhaustive-search-dalam-permainan-congklak.html. Diakses pada tanggal 25 Maret 2022

[6] https://congklak.online/index.html . Diakses pada 25 Maret 2022

[7] Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Institut Teknolgi Bandung.

PERNYATAAN

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

Jakarta, 27 Maret 2022

Christian, 12014005