Penggunaan *Exhaustive Search* dengan Heuristik untuk Pencacahan *Path* Lemparan *Nine-dart finish*

Devin Lukianto - 13514040
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514040@std.stei.itb.ac.id

***Abstract* — Darts 501 adalah salah satu jenis dari pearmainan Darts, dimana untuk memenangkannya pemain harus dapat mengumpulkan poin tepat sejumlah 501 lebih cepat dari lawan bermainnya. Tentu dalam permainan ini, terdapat berbagai macam kemungkinan dan kombinasi angka perolehan poin untuk menyelesaikan permainan. Namun, permainan yang dikatakan sempurna dapat dilakukan hanya dengan melempar sembilan buah panah Darts, yang disebut dengan *nine-dart finish*. Dengan menggunakan algoritma *exhaustive search* yang diberi heuristik, maka dapat diperlihatkan kombinasi dari lemparan sembilan panah tersebut untuk mencapai *nine-dart finish*.**

***Keywords* — Darts 501, *exhaustive search*, heuristik, kombinasi, *nine-dart finish*.**

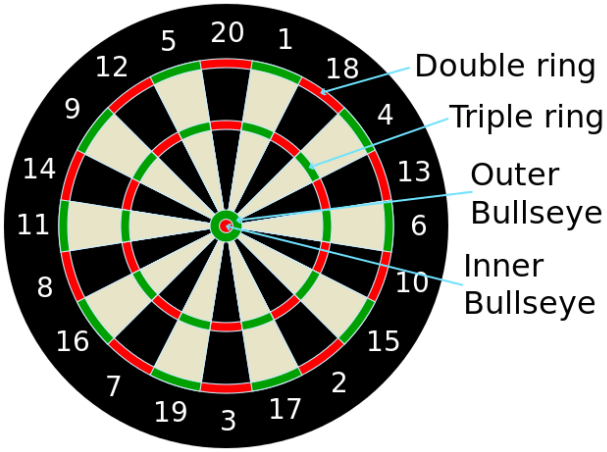
I. PENDAHULUAN

Darts merupakan salah satu jenis permainan membidik target menggunakan tangan dimana pemain harus melemparkan panah kecil ke arah papan target yang berbentuk lingkaran, yang tergantung melekat pada dinding dengan jarak tertentu. Awal mulanya, Darts hanya menjadi sebuah permainan tradisional yang sering dimainkan oleh orang-orang bangsa Eropa. Permainan ini juga cukup marak dimainkan pada bar ataupun kafe di negara-negara barat sana. Namun, seiring berjalannya waktu, Darts kini telah berkembang menjadi sebuah cabang kompetisi permainan profesional yang mendunia.

Terdapat dua alat utama yang digunakan untuk memainkan permainan Darts ini yaitu panah Darts, dan Dartboard (papan Dart).

Panah Dart biasanya terbuat dari bahan besi, kuningan, perak-nikel, atau perpaduan tungsten. Bobot dari panah Dart cukup beragam, dengan standar bobot internasional yaitu 22 gram.

Sedangkan untuk papan Dart, biasanya terbuat dari serat sisal untuk standar internasional. Ukuran papan Dart standar yaitu berdiameter 451 mm, yang dibagi-bagi menjadi 20 bagian radial. Tiap-tiap bagian tersebut dipisahkan oleh kawat ataupun besi tipis. Tiap bagian tersebut ditandai dengan angka yang menunjukan *basic score* dari bagian radial yang bersangkutan. Selain itu, papan dart tersebut juga dibagi lagi menjadi tiga bagian melingkar (berbentuk cincin). Bagian tersebut berpengaruh terhadap penilaian skor permainan karena memiliki arti sebagai *multiplier*. [1]



Gambar 1. Dartboard Diagram
(Sumber:
https://en.wikipedia.org/wiki/File:Dartboard_diagram.svg
diakses pada 5 Mei 2016 pukul 18:54 WIB)

Pada permainan Darts yang sesungguhnya, poin yang didapatkan oleh pemain bergantung pada letak menancapnya panah Dart yang dilemparkan pemain tersebut.

Jika panah Dart pemain menancap pada bagian *Double Ring* (lihat gambar 1), maka poin yang didapatkan pemain tersebut akan dikali dua. Sebagai contoh jika seorang pemain melemparkan panah dan panah tersebut menancap pada *Double Ring* dengan angka 20 (yang disebut dengan D20), maka poin yang didapatkan pemain tersebut adalah 40 poin.

Hal ini berlaku juga pada bagian *Triple Ring* (misal T20, berarti 20 × 3 = 60 poin), dimana poin yang didapat pada panah tersebut akan dikali tiga.

Jika panah Dart menancap pada bagian *Outer Bullseye*, pemain akan mendapatkan poin tunggal *bullseye* sebanyak 25 poin. Sedangkan pemain yang panah Dartnya

menancap pada bagian *Inner Bullseye* akan mendapatkan poin ganda untuk *bullseye*, yaitu 50 poin.

Selain bagian-bagian di atas, poin yang didapatkan pemain adalah poin normal sesuai angka poin yang tertera pada bagian radialnya. Namun, jika panah menancap pada bagian berwarna hitam diluar *Double Ring* ataupun tidak menancap pada papan sama sekali, maka pemain tidak mendapat nilai apapun untuk panah tersebut. [2]

Pada permainan normal Dart 501, pemain harus membuat poin yang dimiliki menjadi nol poin. Poin awal pemain pada permainan ini yaitu 501. Pemain yang bisa menghabiskan 501 poin tersebut lebih cepat dari pemain lawannya akan memenangkan ronde permainan. Pemain mendapatkan poin dari total skor yang didapatkan saat melempar tiga panah Dart tiap gilirannya. Lalu, poin pemain tersebut akan dikurangi poin yang didapatkan saat gilirannya tadi.

Biasanya, lemparan bernilai ganda (*Double*) harus dilakukan pemain untuk dapat mengakhiri permainan. Sebagai contoh, jika poin pemain hanya tersisa sebanyak 40, maka salah satu cara untuk memenangkan permainan yaitu pemain tersebut harus mendapatkan D20 dengan satu panah, yang berarti *Double* 20 (20 × 2 = 40).

Jika total poin yang didapat melebihi poin tersisa (sehingga poin pemain menjadi negatif), atau poin yang tersisa hanyalah sejumlah 1, maka pemain tersebut akan dikenakan *bust* sehingga poin pemain tersebut akan dikembalikan ke poin terakhir semula. [3]

Jika kita melihat semua kemungkinan poin yang dapat diperoleh oleh pemain, dalam permainan normal Dart 501, pemain dapat memenangkan permainan dengan kombinasi poin hanya dari sembilan buah panah Dart yang dilemparkan. Kemenangan ini biasa disebut dengan istilah *nine-dart finish*. Pada makalah ini, penulis akan membahas bagaimana kita dapat mencacah kombinasi *nine-dart finish* tersebut, menggunakan *exhaustive search* yang merupakan brute force, dengan penambahan heuristik di dalamnya.

II. TEORI DASAR

Strategi Algoritma merupakan sebuah pendekatan umum untuk memecahkan persoalan secara algoritmik yang dapat diterapkan pada bermacam-macam persoalan dari berbagai bidang komputasi [Levitin, 2003]. Jika kita dapat menggunakan algoritma yang tepat dan benar, maka kita dapat memecahkan suatu persoalan algoritmik secara mangkus dan cepat. Faktor penting yang harus dilakukan dalam memecahkan persoalan algoritmik yaitu dengan menganalisisnya terlebih dahulu sehingga dapat diperkirakan kira-kira algoritma mana yang paling tepat untuk digunakan untuk mendapatkan solusi dari persoalan tersebut.

Terdapat berbagai macam teknik-teknik algoritma yang dapat digunakan untuk memecahkan persoalan, seperti *brute-force, greedy, Depth First Search, Breadth First Search, backtracking, Divide and Conquer, Decrease and Conquer*, dsb. Pada penulisan makalah kali ini, akan dibahas mengenai penggunaan algoritma brute-force dalam bentuk *exhaustive search* yang mengimplementasikan heuristik di dalamnya.

A. *Algoritma Brute-Force*

Algoritma brute-force merupakan sebuah algoritma pemecahan masalah yang paling sederhana dan mudah. Algoritma ini pada umumnya tidak cerdas dan tidak mangkus dikarenakan jumlah langkahnya yang besar untuk menyelesaikan suatu masalah. Algoritma ini akan mencoba segala kemungkinan solusi yang ada dari suatu permasalahan dan membandingkan tiap solusi tersebut untuk mendapatkan jawaban solusi dari masalah tersebut.

Adapun contoh sederhana pemakaian algoritma brute force dalam menyelesaikan masalah perpangkatan (power). Jika terdapat bilangan a yang dipangkatkan sebanyak n kali, maka algoritma ini akan memrosesnya dengan perhitungan:

$$a^n = a \times a \times ... \times a \text{ (sebanyak n kali), jika n > 0}$$
$$a^n = 1 \text{ , jika n = 0}$$

Sebagai contoh lain dalam penggunaan algoritma brute force yaitu ketika kita ingin mencari elemen terbesar (atau terkecil) dari suatu senarai yang beranggotakan n buah bilangan bulat. Algoritma brute force akan mencacah dan membandingkan satu persatu elemen senarai tersebut untuk menemukan elemen terbesarnya.

Adapun kekuatan dari algoritma brute force ini yaitu bahwa algoritma ini dapat memecahkan hampir sebagian besar dari seluruh permasalahan yang ada. Algoritma ini juga merupakan dasar untuk menghasilkan algoritma standar yang digunakan untuk menyelesaikan tugas-tugas komputasi. Selain itu, metodenya yang mudah dan sederhana membuat algoritma ini menjadi salah satu algoritma yang mudah pula untuk dimengerti.

Namun algoritma ini juga memiliki beberapa *drawback*, diantaranya algoritma ini tidak mangkus, berjalan sangat lambat untuk beberapa kasus, dan tidak sekreatif dan sekonstruktif pemecahan masalah dengan menggunakan algoritma lainnya. [4]

B. *Exhaustive Search*

Exhaustive Search merupakan salah satu cara teknik pencarian solusi yang mengimplentasikan brute force untuk masalah yang melibatkan pencarian elemen dengan sifat khusus, seperti kombinatorik maupun himpunan bagian. *Exhaustive search* merupakan brute force, tetapi tidak setiap brute force adalah *exhaustive search*. [4]

Adapun metode *exhaustive search* memiliki langkah-langkah utama sebagai berikut:

1. Enumerasi setiap solusi dengan sistematis.
2. Evaluasi tiap solusi satu per satu.
3. Umumkan solusi terbaik.

C. *Heuristik*

Untuk memperbaiki kinerja algoritma exhaustive search yang seringkali berorde eksponensial, maka heuristik dapat digunakan sehingga tidak perlu dilakukan

pencarian untuk semua kemungkinan solusi, melainkan hanya yang solusi yang masih memungkinkan saja.

Heuristik yang baik dapat mengurangi waktu yang dibutuhkan untuk memecahkan masalah secara drastis, sehingga akan mempercepat kinerja algoritma untuk menemukan solusi yang diinginkan. [4]

D. Teori Pendukung Lainnya

1. Kombinatorial

Kombinatorial merupakan cabang matematika yang mempelajari pengaturan objek-objek untuk menghitung jumlah dari penyusunan objek-objek tersebut dalam himpunannya.[5] Terdapat beberapa contoh permasalahan yang dapat dipecahkan dengan teori kombinatorial diantaranya:

- Banyak kombinasi dari kata sandi (*password*) yang dapat dibuat jika kata sandi tersebut terdiri dari enam buah karakter alfanumerik.
- Menghitung peluang munculnya nomor lotere dari sekian nomor lotere yang tersedia.

Cara termudah yang dapat dilakukan untuk memecahkan masalah-masalah di atas yaitu dengan menghitung satu-persatu kemungkinan yang ada, serta menguji semua kemungkinan susunannya (brute force murni).

Jika jumlah objek dan kemungkinan solusi dari permasalahan tersebut hanya sedikit, cara enumerasi masih dapat mungkin dilakukan. Namun, jika jumlah objek amat banyak, maka teori kombinatorial merupakan cara yang lebih efektif untuk menghitung banyaknya kemungkinan dari solusi yang ada. Pada hal ini, teori kombinatorial dapat digunakan untuk menentukan heuristik pada pencarian.

2. Permutasi

Permutasi merupakan jumlah urutan berbeda dari suatu pengaturan objek-objek dimana urutan pengurutan elemen-elemennya diperhatikan. Jika di ketahui terdapat sejumlah n elemen, maka permutasi r dari n elemen tersebut (dilambangkan dengan $P(n,r)$ dapat dihitung dengan cara:

$$P(n,r) = n(n-1)(n-2)\dots\big(n-(r-1)\big)$$

$$P(n,r) = \frac{n!}{(n-r)!}$$

3. Kombinasi

Kombinasi adalah bentuk khusus dari permutasi dimana urutan kemunculan elemen-elemennya tidak diperhatikan. Pada kombinasi, urutan acb, bca, dan abc dianggap sama dan hanya dihitung sekali.

Kombinasi r dari n elemen memiliki arti yaitu jumlah kemungkinan pemilihan r elemen tidak terurut, yang di ambil dari n buah elemen. Adapun kombinasi ini dapat dirumuskan sebagai:

$$C(n,r) = \frac{n!}{r!\,(n-r)!}$$

III. PENCACAHAN *PATH NINE-DART FINISH*

Dalam permainan Dart 501, pemain melempar sebanyak tiga buah panah Darts tiap gilirannya. Tiap panah Darts tersebut akan menentukan poin yang didapatkan oleh pemain. Maka dari itu, pertama-tama kita harus mengetahui berapa kemungkinan angka poin yang dapat diperoleh untuk satu panah Dart yang dilempar.

Terdapat 20 bagian radial pada Dartboard yang tiap bagiannya dilambangkan oleh satu angka. Jika kita melihat pada satu bagian radial Dartboard (berbentuk juring), maka terdapat empat bagian petak dimana satu petak terletak di *Double Ring*, satu petak di *Triple Ring*, dan dua petak lain yang merupakan petak normal (berwarna hitam atau putih). Dari petak-petak tersebut maka dapat diketahui bahwa untuk tiap satu bagian juring radial, terdapat tiga kemungkinan skor yang dapat diperoleh oleh pemain. Selain itu, ada petak *Outer Bullseye* yang memiliki nilai 25, dan *Inner Bullseye* yang memiliki nilai 50.

Maka dapat disimpulkan bahwa untuk satu panah Dart yang dilempar, pemain bisa mendapatkan 44 jenis angka poin. Hal ini dapat kita buktikan dengan prinsip inklusi-eksklusi pada teori kombinatorial:

$|N| = 20$ (normal *point*)
$|D| = 20$ (*Double point*)
$|T| = 20$ (*Triple point / Treble*)
$|B| = 2$ (*bullseye point*)
$|F| = 1$ (lemparan meleset, bernilai 0 poin)

$|N \cap D| = 10$ (warna hijau pada Tabel 1)
$|N \cup D| = |N| + |D| - |N \cap D|$
$= 20 + 20 - 10 = 30$

$|(N \cap D) \cap T| = 9$ (warna biru pada Tabel 1)
$|(N \cup D) \cup T| = |N \cup D| + |T| - |(N \cap D) \cap T|$
$= 30 + 20 - 9 = 41$
(warna kuning pada Tabel 1)

$|SUM| = |(N \cup D) \cup T| + |B| + |F|$
$= 41 + 2 + 1 = 44$

NORMAL	DOUBLE	TREBLE
1	2	3
2	4	6
3	6	9
4	8	12
5	10	15
6	12	18
7	14	21
8	16	24
9	18	27
10	20	30
11	22	33
12	24	36
13	26	39
14	28	42
15	30	45
16	32	48
17	34	51
18	36	54
19	38	57
20	40	60
25	50	BULLSEYE

Tabel 1. Tabel Kemungkinan Poin Permainan Darts untuk Pelemparan Satu Panah Dart

Untuk dapat memenangkan permainan dengan *finish* tercepat, maka pemain membutuhkan minimal sembilan buah panah dart dalam tiga giliran bermain untuk menghabiskan 501 poin. *Finish* tercepat ini disebut juga dengan *nine-dart finish*, yang berarti *finish* sempurna hanya dengan menggunakan sembilan kali lemparan panah Dart tanpa ada lemparan meleset dan perolehan skor yang tepat.

Jika kita melakukan pencacahan brute force untuk semua kombinasi poin yang dapat dicapai pada permainan dengan sembilan buah panah dart ini, maka akan terjadi pencacahan sebanyak:

$$44^9 = 6.1812184 \times E^{14}$$

Algoritma brute-force pencacahan diatas memiliki kompleksitas algoritma sebesar O(n^9) dan ini sangatlah besar dan kompleks tidak dapat dipecahkan dalam waktu singkat, juga tidak semua alternatif solusi yang ditawarkan memenuhi peraturan untuk menghabiskan 501 poin.

Maka dari itu, kita akan mengubah pencacahan ini menjadi sebuah *exhaustive search*, di mana batasan utamanya ialah bahwa jumlah dari semua poin sembilan anak panah haruslah sama dengan 501. Patut diingat kembali bahwa pada permainan Dart 501, lemparan terakhir harus merupakan lemparan Double, yang berarti lemparan yang sah adalah jika panah lemparan tersebut jatuh pada bagian *Double Ring*, atau *Inner Bullseye*.[6]

Karena lemparan terakhir (*Double-out*) harus merupakan lemparan *Double Ring*, atau *Inner Bullseye*, maka skor yang dapat pemain peroleh pada panah ke-sembilan adalah seluruh skor yang terdapat pada kolom *Double* di Tabel 1. Delapan panah awal dapat berupa kombinasi skor yang dapat diubah urutannya. Namun, perlu diperhatikan bahwa tidak semua skor dalam *Double-out* dapat menghasilkan *nine-dart finish*. Adapun Tabel *Double-out* yang lebih jelas dapat dilihat di Tabel 2.

Jika kita mengasumsikan semua panah Dart suatu pemain mendapatkan skor tertinggi yaitu T20 (60 poin), maka untuk sembilan buah panah, total skor yang didapatkan adalah 540, melebihi batas poin permainan 501, sehingga pemain pun akan terkena *bust* dan tidak dapat menyelesaikan permainan dengan *nine-dart finish*. Maka dari itu, kita harus memperkecil perolehan skor T20 menjadi delapan panah saja.

Ternyata, jika delapan panah pertama suatu pemain mendapatkan T20, maka sisa skor yang dimiliki pemain yaitu 501 – (8 × 60) = 21. Kombinasi kali ini jelas mustahil untuk dapat memperoleh *nine-dart finish*, sebab untuk panah ke-sembilan, pemain harus bisa mendapatkan skor 21 dengan kondisi *Double*, yang berarti pemain harus mendapat D11.5 dimana angka tersebut tidak ada dalam permainan (lihat Tabel 2).

1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18
10	20
11	22
12	24
13	26
14	28
15	30
16	32
17	34
18	36
19	38
20	40
BULLSEYE	50

Tabel 2. Tabel Lemparan Panah Dart ke-9
(poin sah diwarnai kuning, kolom pertama menunjukan petak *Double-out*, kolom kedua menunjukan poin yang didapat)

Pengecekan delapan panah T20 di atas sekaligus membuktikan bahwa semua skor *Double* di bawah D11.5 tidak akan bisa menyelesaikan permainan hanya dengan sembilan buah panah. Sehingga dapat kita nyatakan bahwa D1 hingga D11 tidak akan bisa menjadi solusi panah ke-sembilan dari *Nine-dart finish*. Hal ini dapat kita jadikan **salah satu kriteria heuristik** pada pencarian yang akan kita lakukan nantinya.

Untuk mempermudah tahap pengecekan selanjutnya, kita dapat membuat model permasalahan:

Jika 8 panah awal bernilai 60, maka berapa poin yang harus didapatkan panah terakhir untuk mencapai nilai 501?

Jika 7 panah awal bernilai 60, maka berapa poin yang didapatkan 2 panah terakhir untuk mencapai nilai 501? dst.

Kita akan melakukan pengujian ini terus-menerus hingga kita mencapai keadaan bahwa hanya 1 panah yang memiliki skor 60 poin. Dengan melakukan percobaan ini, kita akan sekaligus membuat semua kemungkinan dari poin-poin yang dapat diperoleh untuk tiap anak panahnya, dengan tidak melupakan syarat bahwa anak panah terakhir harus merupakan lemparan *Double-out* yang sah. Secara matematis dapat dtuliskan sebagai:

$$a + b + c + d + e + f + g + h + i = 501$$

a merupakan skor panah pertama
b merupakan skor panah kedua
c merupakan skor panah ketiga, dst.

dimana

$$21 \leq a \leq 60$$
$$81 - a \leq b \leq 60$$
$$141 - a - b \leq c \leq 60$$
$$201 - a - b - c \leq d \leq 60$$
$$261 - a - b - c - d \leq e \leq 60$$
$$321 - a - b - c - d - e \leq f \leq 60$$
$$381 - a - b - c - d - e - f \leq g \leq 60$$
$$441 - a - b - c - d - e - f - g \leq h \leq 60$$
$$501 - a - b - c - d - e - f - g - h \leq i \leq 60$$

dan nilai i merupakan nilai yang harus dipenuhi untuk *Double-out*.

Pertidaksamaan matematis diatas dapat kita uji untuk menentukan semua kemungkinan *path* yang mungkin untuk mencapai *nine-dart finish*. Maka dari itu, pertidaksamaan ini dapat kita bentuk sebagai suatu algoritma *exhaustive search*.

Heuristik pertama yang kita gunakan dalam algoritma kita adalah, bahwa terdapat beberapa angka yang tidak akan pernah muncul sebagai sebuah skor. Angka-angka tersebut merupakan hasil komplemen dari seluruh himpunan angka yang terdapat pada Tabel 1, dengan *range* dari 1 hingga 60. Angka-angka tersebut ialah 23, 29, 31, 35, 37, 41, 43, 44, 46, 47, 49, 52, 53, 55, 56, 58, dan 59. Maka range yang mengandung angka-angka ini akan tidak akan dicari lebih lanjut pada algoritma kita. Angka-angka ini akan kita sebut sebagai daftar **`Heuristik 1`** untuk mempercepat kita menemukan solusi pemecahan masalah kombinasi lintasan ini.

Selain itu, dengan referensi nilai untuk panah ke-sembilan dari Tabel 2, maka kita dapat membuat heuristik kedua untuk pencarian panah *Double-out* yang dapat mengakibatkan *nine-dart finish* terjadi. Maka dari itu, seluruh elemen Tabel 2 kolom kedua yang bernilai diatas 24 (D12) akan dimasukkan ke dalam daftar **`Heuristik 2.`** Adapun pembahasan mengapa yang D1 hingga D11 tidak perlu untuk dimasukkan ke dalam daftar pengecekan terlah dijelaskan sebelumnya sebagai salah satu kriteria heuristik.

Selain itu untuk lebih memperkecil ruang pencarian, kita juga hanya akan mengecek untuk poin tiap panah dengan range variabel *temp* hingga 60 saja, dimana temp merupakan sisa kumulatif untuk poin yang mungkin dari total poin yang dapat diperoleh, sehingga program tidak perlu mengecek angka-angka yang akan menghasilkan poin *exceeding*. Heuristik kali ini dapat kita lihat pada bagian algoritma *assignment temp*, dan pengulangan pencarian untuk variabel poin panah ke – n yang nilainya berada di antara *temp* dan 60.

Adapun *pseudocode* permutasi algoritma *exhaustive search* kita adalah seperti ini:

```
procedure ninedartfinish()

DEKLARASI
 a,b,c,d,e,f,g,h : integer
 temp : integer
 Heuristik 1 : daftar elemen Heuristik 1
 Heu2 : daftar elemen Heuristik 2

ALGORITMA
 temp = 21
 for (temp < a <=60) do
  if a <> Heuristik 1 then
  temp <- 81-a

  for (temp < b <=60) do
   if b <> Heuristik 1 then
   temp <- 141-a-b

   for (temp < c <=60) do
    if c <> Heuristik 1 then
    temp <- 201-a-b-c

    for (temp < d <=60) do
     if d <> Heuristik 1 then
     temp <- 261-a-b-c-d

     for (temp < e <=60) do
      if e <> Heuristik 1 then
      temp <- 321-a-b-c-d-e

      for (temp < f <=60) do
       if f <> Heuristik 1 then
       temp <- 381-a-b-c-d-e-f
```

```
        for (temp < g <=60) do
         if g <> Heuristik 1 then
         temp <- 441-a-b-c-d-e-f-g

         for (temp < h <=60) do
          if h <> Heuristik 1 then

          for (cacah Heu2) do
           if a+b+c+d+e+f+g+h+Heu2 = 501
            print(a,b,c,d,e,f,g,h,Heu2)
          endfor
         endfor
        endfor
       endfor
      endfor
     endfor
    endfor
   endfor
  endfor
endfor
```

Setelah program berjalan dan dilakukan pencacahan, maka terlihat bahwa hanya ada enam jenis skor yang sah yang dapat diperoleh pemain untuk menghasilkan *nine-dart finish*. Keenam jenis skor tersebut adalah D12 (24 poin), D15 (30 poin), D17 (34 poin), D18 (36 poin), D20 (40 poin), dan *Inner Bullseye* (50 poin). Hasil tersebut dapat dilihat dengan lebih jelas pada Tabel 2.

Jika kita menambahkan sebuah variabel *count* dimana jumlah dari variabel ini akan bertambah satu untuk tiap solusi yang sah, maka akan terlihat bahwa terdapat sejumlah 3944 jalur permutasi solusi yang sah untuk mendapatkan *nine-dart finish*.

Panah *Double-out*	Jumlah Permutasi Jalur
D12 (24)	8
D15 (30)	120
D17 (34)	56
D18 (36)	792
D20 (40)	621
Inner Bullseye (50)	2296

Tabel 3. Tabel Permutasi Kemungkinan Jalur Lemparan Panah Dart ke-9 (Double-out)

Untuk mendapatkan berapa jumlah kombinasi dari semua kemungkinan jalur lemparan yang dapat mencapai *nine-dart finish*, penulis mengimplementasikan aturan yang akan menjadi heuristik tambahan bahwa penulis hanya menghitung jumlah kemungkinan yang urutannya terurut membesar dari semua kemungkinan permutasi solusi dari *nine-dart finish*. Hal ini dapat dilakukan, sebab dalam permutasi urutan dari elemen-elemen diperhatikan. Jika terdapat urutan solusi yang urutan elemennya sama, maka kemungkinan tersebut akan disederhanakan menjadi satu buah saja oleh program yang digunakan penulis sehingga kita bisa mendapatkan hasil kombinasinya. Syarat inilah yang akan kita jadikan heuristik tambahan untuk menemukan jumlah kombinasi jalur.

Heuristik tambahan tersebut dapat kita rumuskan dalam *pseudocode* seperti berikut:

```
//penambahan pada tiap pengulangan, untuk
variabel a sampai variabel g pada bagian
sesudah assign temp
…
temp = …
if (temp < a)
   temp = a  …
```

Panah *Double-out*	Jumlah Kombinasi Jalur
D12 (24)	1
D15 (30)	3
D17 (34)	1
D18 (36)	7
D20 (40)	3
Inner Bullseye (50)	7

Tabel 4. Tabel Kombinasi Kemungkinan Jalur Lemparan Panah Dart ke-9 (Double-out)

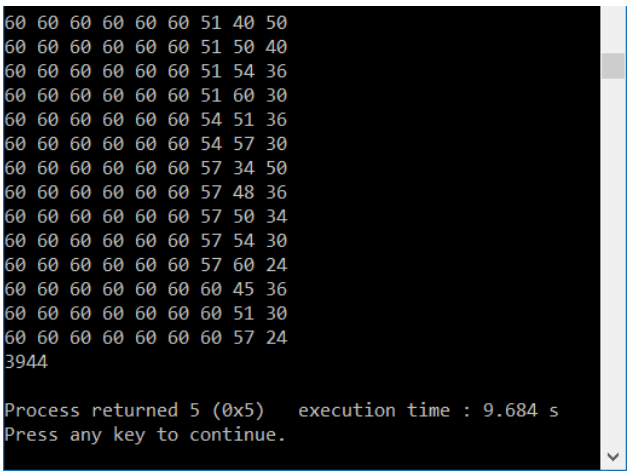
Possible paths for a nine-dart finish

Double-Out	The other 8 darts								Paths
	T20	T19	T18	T17	T16	T15	Bull	D20	D17	
D12	7	1								8
D15	7			1						8
	6	1	1							56
	5	3								56
D17	6	1					1			56
D18	7					1				8
	6	1			1					56
	6		1	1						56
	5	2		1						168
	5	1	2							168
	4	3	1							280
	3	5								56
D20	6			1			1			56
	5	1	1				1			336
	4	3					1			280
Bull	5			1			2			168
	4	1	1				2			840
	3	3					2			560
	6			1				1		56
	5	1	1					1		336
	4	3						1		280
	6	1							1	56
									Sum	3944

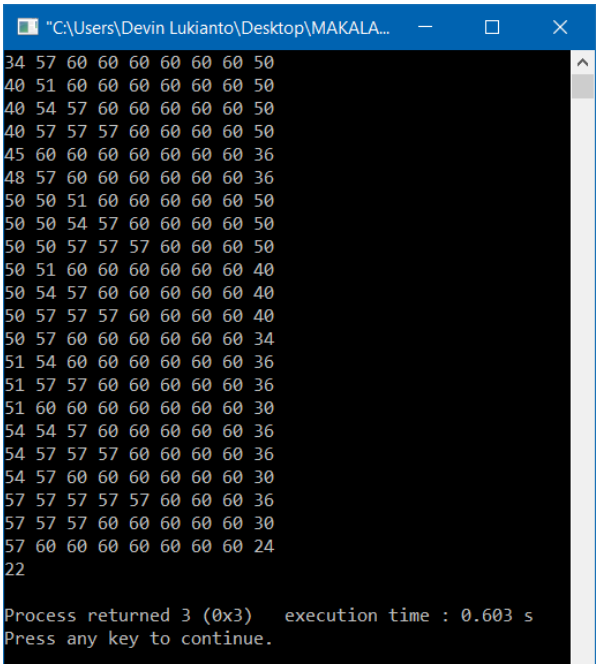
Gambar 2. *Possible Paths for a Nine-dart Finish*
(Sumber: https://en.wikipedia.org/wiki/Nine-dart_finish#/media/File:Possible_paths_for_a_nine-dart_finish.png diakses pada 6 Mei pukul 23:03 WIB)

Jika kita membuat kombinasi dari 3944 jalur tersebut, dimana urutan delapan panah awal tidak diperhatikan, maka akan terdapat 22 jenis langkah yang sah.

Berikut penulis lampirkan dua hasil screenshot program yang mengacu pada *pseudocode* diatas, dengan satu screenshot menunjukan hasil permutasi, dan yang satu menunjukan hasil kombinasi dari *path nine-dart finish*:



Gambar 3. Hasil Permutasi Jumlah Lintasan *Nine-dart finish*



Gambar 4. Hasil Kombinasi Jumlah Lintasan *Nine-dart finish*

IV. KESIMPULAN

Penggunaan algoritma *exhaustive search* dengan beberapa heuristik dapat memecahkan permasalahan kombinatorial *nine-dart finish* ini dengan cukup cepat. Adapun untuk permutasi *path*, dibutuhkan waktu 9.684 sekon untuk mencacah semua kemungkinan. Sedangkan untuk kombinasi *path*, dibutuhkan waktu 0.603 sekon.

Algoritma ini juga dapat menunjukan kepada kita semua kombinasi yang mungkin untuk melakukan nine-dart finish yaitu sebanyak 3944 jalur permutasi dan 22 jalur kombinasi skor sembilan buah panah.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan terima kasih dan segala puji syukur kepada Tuhan Yang Maha Esa atas segala rahmat dan berkat yang diberikan, sehingga penulis dapat menyelesaikan makalah ini tepat waktu.

Penulis juga mengucapkan terima kasih kepada orang tua penulis, yang senantiasa mendukung penulis dari rumah sana sehingga sekarang penulis dapat menuntut ilmu di Institut Teknologi Bandung dan mengerjakan makalah ini.

Penulis juga mengucapkan terima kasih kepada dosen mata kuliah IF2211 Strategi Algoritma yang penulis hormati, yaitu Bapak Dr. Ir. Rinaldi Munir dan Ibu Nur Ulfa Maulidevi, atas segala ilmu dan pelajaran yang telah mereka berikan kepada penulis selama kurang lebih satu semester ini.

Tidak lupa penulis berterima kasih kepada teman-teman yang telah mendukung, membantu, dan memberikan semangat selama penulis membuat makalah ini.

Akhir kata, semoga makalah ini dapat bermanfaat bagi para pembaca. Terima kasih.

REFERENSI

[1] http://dartsinfoworld.com/about-darts/ diakses pada 5 Mei 2016 pukul 19:00

[2] http://www.dartswdf.com/basicsofdarts/ diakses pada 5 Mei 2016 pukul 20:11

[3] http://www.mastersgames.com/rules/darts-rules.htm diakses pada 5 Mei 2016 pukul 20:23

[4] Munir, Rinaldi. "Diktat Kuliah IF2211 Strategi Algoritma". Program Studi Teknik Infomatika STEI ITB. 2009.

[5] Munir, Rinaldi. "Diktat Kuliah IF2120 Matematika Diskrit" edisi keempat. Program Studi Teknik Infomatika STEI ITB. 2006.

[6] http://dartsinfoworld.com/9-dart-finish/ diakses pada pukul diakses pada 6 Mei 2016 pukul 22:11

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, 7 Mei 2016

Devin Lukianto
13514040