TUGAS BESAR 1
STRATEGI ALGORITA IF2211
PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN ROBOCODE TANK ROYALE

Kelompok 34
Undang-undang 34

Richard Christian	13523024
David Bakti Lodianto	13523083
Lutfi Hakim Yusra	13523084

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132

2024

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pertempuran Robocode berlangsung hingga bot-bot bertarung hanya tersisa satu seperti permainan Battle Royale, karena itulah permainan ini dinamakan Tank Royale. Dalam permainan ini, pemain berperan sebagai programmer bot dan tidak memiliki kendali langsung atas permainan. Pemain hanya bertugas untuk membuat program yang menentukan logika atau "otak" bot. Program yang dibuat akan berisi instruksi tentang cara bot bergerak, mendeteksi bot lawan, menembakkan senjatanya, serta bagaimana bot bereaksi terhadap berbagai kejadian selama pertempuran.

Pada Tugas Besar pertama Strategi Algoritma ini, kami diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain dengan implementasi strategi *Greedy*.

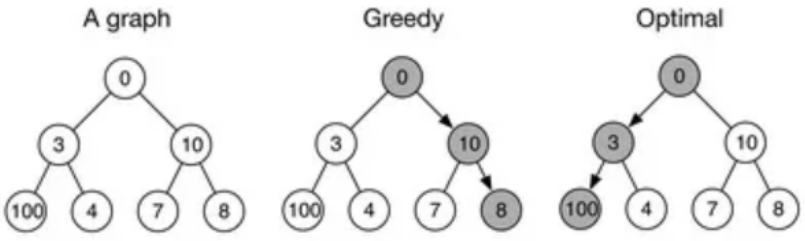
BAB II
LANDASAN TEORI

A. Algoritma Greedy

Terdapat banyak pendekatan terhadap penyelesaian permasalahan optimasi, salah satunya adalah pendekatan menggunakan algoritma *greedy*. Algoritma *Greedy* merupakan kategori algoritma yang cukup populer digunakan, karena dapat terpikirkan cepat, seringkali merupakan optimasi heuristik dari penyelesaian *brute-force*. Kategori algoritma ini dilakukan dengan memecahkan permasalahan langkah demi langkah dan mengambil pilihan terbaik tiap langkahnya yang dapat diperoleh pada saat itu (*local optimum*), dan "berharap" bahwa pilihan-pilihan tersebut dapat berakhir pada pilihan terbaik secara global (*global optimum*). Berikut contoh persoalan-persoalan populer yang dapat diselesaikan dengan algoritma *greedy*:

- Algoritma Kruskal dan Prim untuk mencari MST (*Minimum Spanning Tree*)
- Algoritma Dijkstra untuk mencari rute minimum pada graf berbobot
- Algoritma Coding Huffman untuk pemampatan data.

Dikatakan kategori algoritma *greedy* karena alasan sebelumnya, dan bahwa algoritma penyelesaian ini belum tentu membawakan hasil paling optimal, kecuali dibuktikan secara matematis. Selain itu, algoritma ini seringkali dibuat untuk tidak perlu memutar balik ke langkah sebelumnya, sehingga harus mengambil pilihan terbaik pada langkah itu. Namun, pilihan tersebut belum tentu merupakan pilihan yang terbaik secara keseluruhan. Contohnya, pada persoalan penukaran koin dengan koin yang bukan kelipatan satu sama lain.



Akan tetapi, kategori algoritma ini dapat menghasilkan kompleksitas waktu yang tergolong sederhana dibandingkan dengan algoritma *brute-force*, karena *greedy* hanya mencari satu solusi, sedangkan *brute-force* mencari setiap solusi dan mencari yang paling optimal darinya. Oleh karena itu, untuk persoalan-persoalan yang cenderung menghabiskan waktu dan/atau memori yang besar untuk mencari solusi paling optimal, lebih baik menggunakan algoritma *greedy* yang dapat menghasilkan solusi yang paling mendekati optimal.

Pada algoritma *greedy*, terdapat beberapa elemen-elemen yang dijadikan acuan untuk proses algoritmanya, yaitu:

1. Himpunan kandidat, C : Kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi, S : Berisi kandidat yang sudah dipilih
3. Fungsi solusi : Menentukan apakah kandidat memberikan solusi
4. Fungsi seleksi : Memilih kandidat berdasarkan strategi *greedy* (heuristik)
5. Fungsi kelayakan : Memeriksa apakah kandidat yang dipilih dimasukkan ke S
6. Fungsi objektif : Memaksimumkan atau meminimumkan

Dari elemen-elemen di atas, algoritma *greedy* melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, pemilihan S ditentukan oleh pemenuhan fungsi solusi, dengan menentukan langkah terbaik melalui fungsi seleksi, dibatasi fungsi kelayakan, menuju fungsi objektif.

B. Robocode Tank Royale

Robocode Tank Royale merupakan sebuah permainan yang terdiri dari tank-tank yang saling bertarung untuk mendapatkan poin tertinggi. Bot yang bermain tidak dapat dikendalikan oleh pemain secara tradisional, tetapi setiap gerakannya diatur oleh kode yang disusun sebelumnya. Untuk mencapai poin yang tinggi, bot harus dapat menentukan aksi yang paling sesuai di setiap momennya dengan memanfaatkan algoritma yang efektif dan efisien.

Pada permainan ini, bot dapat melakukan sebuah aksi setiap *turn*, yang berjalan setiap 30-50 ms. Aksi yang dipilih oleh bot pada umumnya meliputi gerakan maju mundur dari badan bot, rotasi dari badan, *gun*, dan *radar* bot, dan menembakkan peluru. Setiap aksi yang dilakukan dapat dilakukan dengan parameter yang ditentukan.

1. Bot

Bot terdiri dari tiga bagian, *gun*, *radar*, *body*. *Body* menggerakkan keseluruhan tank, *radar* berfungsi untuk pemindaian posisi bot lainnya, dan *gun* digunakan untuk menembakkan peluru. Setiap bot juga memulai permainan dengan awal energi sebanyak 100 poin. Jika energi berkurang hingga di bawah 0, maka bot tersebut dinonaktifkan.

2. Gerakan Bot

Body dapat bergerak maju mundur dengan percepatan maksimum sebesar 1 unit per *turn* dan pengereman dengan perlambatan maksimum 2 unit per *turn*. *Body* juga dapat berputar 10 derajat per *turn*, yang semakin lambat seiring kecepatan bot bergerak. *Gun* dapat berputar 20 derajat per *turn*, dan *radar* dapat berputar 45 derajat per *turn*.

3. **Pemindaian**

   Semua bot lain yang dilewati oleh *radar* akan dideteksi keberadaannya oleh bot. Pada awalnya, jangkauan *radar* hanya berbentuk sebuah garis. Namun, perpindahan dari garis *radar* (dari perputaran *radar* ataupun gerakan *body*) antar *turn* akan mengakibatkan jangkauan *radar* meluas, dengan luas sapuan *radar* dari garis *radar* sebelumnya ke arah *radar* saat ini.

4. **Tembak Peluru**

   Peluru akan muncul dari *gun*, dengan kekuatan yang ditentukan (0.1 - 3). Peluru yang ditembakkan akan mengurangi energi bot sebesar kekuatan peluru tersebut. Peluru yang semakin kuat akan memiliki radius peluru dan nilai *damage* yang lebih besar, tetapi kecepatan peluru lebih kecil dan jeda antar penembakan lebih besar. Ketika peluru mengenai bot musuh, bot akan mendapatkan energi sebesar 3 kali lipat kekuatan peluru tersebut, dan bot musuh akan berkurang energinya tergantung kekuatan peluru.

   $$bulletSpeed = 20 - (3 \times bulletPower)$$

   $$bulletDamage = 4 \times bulletPower$$

   $$bulletDamage = 4 \times bulletPower + 2 \times (bulletPower - 1)$$

   (bulletPower > 1)

5. **Tabrakan**

   Ketika bot menabrak dinding, akan berkurang energi tergantung kecepatannya. Ketika mengenai bot lain, akan memberikan *damage* 0.6 kepada sesama. Namun, jika salah satu dari bot berjalan maju ke depan, bot yang berjalan maju ke depan akan mendapatkan bonus ketika merusakkan bot.

   $$wallDamage = \frac{|v|}{2} - 1$$

6. **Skor**

   Setiap bot akan mengumpulkan skor selama 10 ronde. Pemenang ditentukan murni oleh pemegangan skor. Bot menghasilkan skor untuk setiap nilai bulletDamage yang dihasilkan. Ketika musuh bot dijatuhkan oleh peluru kita, kita mendapatkan bonus sebesar 20% dari *damage* yang diakibatkan. Setiap ada bot yang dinonaktifkan, bot lainnya akan mendapatkan skor sebesar 50, dan yang *last survivor* akan memberikan skor sebesar 10 untuk tiap bot yang terjatuh. Untuk setiap nilai *damage* dari tabrakan, akan mendapatkan 2 poin. Ketika bot menjatuhkan bot musuh, akan mendapatkan 30% dari damage yang diakibatkan

Berdasarkan enam parameter yang telah disebutkan, bot harus menentukan langkah terbaik dari setiap turn. Langkah tersebut ditentukan dengan algoritma *Greedy*, mengolah data yang saat itu dimiliki oleh bot melalui pemikiran heuristic yang dipilih.

BAB III
APLIKASI STRATEGI GREEDY

A. Elemen-elemen Algoritma Greedy pada Pembuatan Bot

Dengan mengacu pada elemen-elemen algoritma *greedy*, kita dapat melakukan mapping permasalahan menentukan algoritma bot pada permainan Robocode Tank Royale menjadi sebuah permasalahan yang dapat dicari solusinya melalui pendekatan *greedy*.

1. Himpunan Kandidat

Setiap bot yang ada pada arena memiliki kemampuan untuk melakukan sebuah aksi yang sama di setiap *turn*-nya. Kemampuannya telah dijelaskan lebih lanjut pada bab sebelumnya, dan setiap kemungkinan aksi tersebut pada setiap *turn* masuk ke himpunan kandidat.

2. Himpunan Solusi

Rangkaian aksi yang dipilih bot selama permainan menuju kemenangan melalui strategi bot masing-masing merupakan himpunan solusi dari bot tersebut.

3. Fungsi Solusi

Setiap strategi memiliki acuan strategi masing-masing, dan fungsi solusi menentukan apakah urutan aksi yang telah dipilih oleh bot sudah memenuhi syarat dari strategi tersebut.

4. Fungsi Seleksi

Bot menentukan aksi yang terbaik himpunan kandidat yang akan dimasukkan ke dalam himpunan solusi melalui fungsi seleksi. Lebih tepatnya, fungsi seleksi menentukan langkah terbaik yang akan diambil oleh bot pada saat itu.

5. Fungsi Kelayakan

Fungsi kelayakan pada pembangunan algoritma pada bot berperan sebagai pengecekan apakah mungkin atau tidaknya sebuah aksi.

6. Fungsi Objektif

Algoritma yang diikuti oleh bot diukur keberhasilannya dalam pemenuhan strategi bot yang telah dipilih melalui fungsi objektif.

Berikut adalah persoalan utama yang terdapat pada elemen di permainan robocode:

No	Elemen	Kegunaan
1	Movement	- Menentukan pergerakan robot - Robot dapat maju, mundur, dan berbelok sesuai perintah
2	Ramming	- Mekanisme menyerang lawan dengan menyeruduk tank musuh.
3	Firing	- Mekanisme menyerang lawan dengan menembakkan proyektil yang mengkonsumsi energi - Kecepatan proyektil tergantung dengan energi yang digunakan
4	Scanning	- Proses pemindaian oleh robot untuk mendeteksi musuh dan posisi
5	Energy	- Sistem yang digunakan sebagai nyawa dan kapasitas bot untuk menyerang.
6	Events	- Event yang terdeteksi oleh bot - Dapat digunakan untuk menciptakan *trigger* pada robot dan merespon
7	Map	- Daerah pertempuran robot, dengan batasan ukuran tertentu dan dinding di setiap sisi.

B. Eksplorasi Alternatif Solusi

Terdapat berbagai alternatif solusi pendekatan *Greedy* pada masalah RoboCode yang dianalisis oleh kelompok kami, diantaranya:

1. Greedy by Distance

Greedy by Distance dilakukan dengan bot berusaha untuk mendekatkan diri ke target yang dipilih. Pendekatan ini berusaha meminimalkan kemungkinan peluru untuk tidak mengenai target, dan juga memaksimalkan kerusakan melalui tabrakan. Bot dengan strategi ini sangat kuat untuk menyerang bot dengan mobilitas yang minim, tetapi lemah jika bot lawan sudah siaga dan menjauh, atau ramai.

2. Greedy by Movement

Greedy by Movement dilakukan dengan bot bergerak sedemikian mungkin sehingga bot susah diprediksi pergerakannya, sehingga jarang menerima

serangan dari peluru-peluru musuh. Selain itu, perlu diperhatikan juga pergerakan bot agar energi tidak terbuang sia-sia dengan menabrak dinding.

3. **Greedy by Position**

   Greedy by Position dilakukan dengan memposisikan bot pada lokasi yang menguntungkan bagi bot. Pendekatan ini fokus kepada survivability agar bot dapat lebih terhindar dari keramaian dan lebih sulit untuk diserang. Bot dengan pendekatan ini lebih diuntungkan pada pertarungan dengan banyak bot

4. **Greedy by Firepower**

   Greedy by Firepower dilakukan dengan menyesuaikan energi dalam penembakan sesuai dengan jarak musuh. Pendekatan ini berusaha memaksimalkan energi yang didapatkan sambil menjaga presisi penembakan bot yang dipengaruhi oleh kecepatan peluru.

5. **Greedy by Targeting**

   Greedy by Targeting dilakukan dengan memilih target yang akan diserang oleh bot. Pendekatan ini bekerja dengan memfokuskan serangan pada salah satu target yang sudah dipilih. Dengan pendekatan ini, bot berupaya mengurangi waktu mencari target sekaligus meningkatkan efektivitas dari serangan.

C. Analisis Solusi Greedy

a. Global (Greedy By Firepower)

Greedy by firepower diimplementasikan pada semua alternatif bot dengan alternatif weighting yang berbeda-beda. Kami memilih untuk mengimplementasikan firepower secara global, karena proses implementasinya yang cukup sederhana, tidak mengganggu pendekatan *greedy* lain, dan cukup efektif dalam pemanfaatan bot. Firepower secara umum diimplementasikan dengan meningkatkan kekuatan serangan semakin dekat musuh pada bot.

b. Conquest (Greedy By Distance)

Bot akan menggunakan implementasi dari *greedy by firepower*, dan *greedy by distance*. Bot ini akan memindai arena, lalu akan mengejar dengan kecepatan penuh. Setiap *turn*, bot akan mendekati bot lainnya, agar penembakan peluru dapat dimaksimalkan. Selain itu, bot ini tidak memperhatikan hal yang lain, menjaminkan kecepatan tertinggi untuk mencapai tujuan.

c. Custom (Greedy By Position)

Implementasi *Greedy By Position* diimplementasikan dengan memposisikan bot di sebelah tembok. Bot akan berbelok menghadap tembok, kemudian terus maju hingga mencapai tembok tersebut. Setelah mencapai tembok, bot akan mengambil posisi penyerangan, dan berosilasi pada tembok yang sama untuk menghindari peluru. Bot akan terus bergerak dekat tembok hingga game selesai. Kelebihan approach ini adalah mengurangi sudut penyerangan terhadap bot, dimana bila bot tidak berada di sebelah tembok, maka bot dapat diserang dari segala arah, sedangkan jika berada dekat tembok bot hanya dapat diserang dari 180 derajat bot yang tidak menghadap pada tembok. Bot ini juga menghindari pergerakan yang dapat menyebabkan tabrakan tembok saat osilasi.

d. Perpendicular (Greedy By Targeting, Greedy By Movement)

Bot Perpendicular menggunakan dua pendekatan greedy sekaligus, yakni targeting dan movement. Pertama, greedy by targeting diimplementasikan dengan mengambil bot terdekat sebagai target dan mengunci radar untuk tetap mengikuti bot tersebut. Hal ini menyimpan waktu untuk mencari musuh dan mendekati musuh. Selain itu, bot ini juga bergerak dengan pendekatan greedy by movement, lebih spesifiknya dengan bergerak mengikuti musuh dan pada jarak tertentu, ia akan bergerak tegak lurus terhadap musuh, secara efektif me-mimik pergerakan musuh jika bergerak dengan cepat, dan mengitari musuh jika musuh lambat, supaya lebih susah untuk ditembak. Selain itu, bot ini juga memperhatikan apakah terdapat musuh yang menabraknya, sehingga ia bisa bergerak menjauh darinya (mengganti target ke bot yang menabraknya). Bot ini juga akan menembak sambil bergerak.

e. Box (Greedy By Positioning dan Greedy By Movement)

Bot Box memaksimalkan positioning, yakni bergerak ke pojokan terdekat. Di pojokan, bot hanya akan menerima potensi serangan lebih kecil. Jika pojokan sekarang terlalu ramai dengan musuh, bot ini akan segera memindai arena dan bergerak ke pojokan dengan jumlah musuh paling sedikit. Selain itu, bot ini akan bergerak dengan pola persegi panjang, di pojokan tersebut agar tidak diam saja dan menjadi sasaran mudah untuk ditembak. Persegi panjang ini memiliki panjang dan lebar yang cukup panjang sehingga dapat mendapatkan kecepatan dan menghindari peluru yang membidik secara langsung, namun juga cukup pendek agar dapat menghindari peluru yang membidik dengan menebak kecepatan dan direksi.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

A. Pseudocode Bot

a. Perpendicular

Greedy by Targeting

```
Class Perpendicular

Algoritma

state = 0 // keadaan dari bot, 0: mencari musuh; 1: mengejar
count = 0 // jumlah musuh yang sudah di-scan
closestBotDistance = -1
closestBotId = -1
// e adalah bot yang dipindai

if (closestBotDistance < 0 OR
    DistanceTo(e.X, e.Y) < closestBotDistance) then
    closestBotDistance = DistanceTo(e.X, e.Y)
    closestBotId = e.BotId

count = count + 1

if (count >= EnemyCount)
    state = 1
```

Greedy by Movement

```
Class Perpendicular

Algoritma ChaseEnemy

gunTurn = GunBearingTo(e.X, e.Y)

turn = BearingTo(e.X, e.Y)

enemyDistance = DistanceTo(e.X, e.Y)

enemyLateralSpeed = e.Speed *
    Sin(e.Direction - (turn + Direction))

overcompensate = enemyLateralSpeed/Speed

// ganti nilai kecepatan maksimum secara random supaya lebih susah
diprediksi
if (random(0, 1) > 0.9) { then // nilai random di antara 0 - 1
    MaxSpeed = random(0, 1) * 10 + 4
}

// sesuaikan kecepatan dan pergerakan
if (enemyDistance > 200) then
    SetTurnLeft(turn + overcompensate + Max((90 -
        (Abs(enemyDistance - 100) * 2)), 0))
```

```
    if (enemyDistance > 200) then
        dir = 1; // agar bergerak menuju musuh
    SetTurnGunLeft(gunTurn +
    enemyLateralSpeed*(800/enemyDistance))
    SetForward(100 * dir)
else if (enemyDistance >= 50) then
    perpendicular = NormalizeRelativeAngle(
        BearingTo(e.X, e.Y) + 90)
    SetTurnLeft(perpendicular + overcompensate)
    SetForward(100 * dir)
    SetTurnGunLeft(gunTurn +
        enemyLateralSpeed * (200/enemyDistance))
else
    SetTurnLeft(turn)
    SetTurnGunLeft(gunTurn)
    SetBack(100)
```

Greedy by Firepower

```
Class Perpendicular

Algoritma ChaseEnemy

// sesuaikan kecepatan dan pergerakan
if (enemyDistance > 200) then
    SetFire(0.7)
else if (enemyDistance >= 50) then
    SetFire(1.5)
else
    SetFire(3)
```

b. Conquest

Greedy by Distance

```
Class ConquestBot

Algoritma

TargetSpeed = 8 // maksimalkan kecepatan

if (no turn request)then // belum ada bot yang ke scan
   setTurnRadarLeft(infinite)

if (ada bot yang ke-scan) then
   e = bot tersebut
   radarTurn = radarBearingTo(e.X, e.Y)//arah ke koordinat bot
   direction_tujuan = bodyBearingTo(e.X, e.Y)

   // putarkan radar dan badan bot menuju arah bot
   setRadarTurnLeft(radarTurn)
   setTurnLeft(direction)
```

Greedy by Firepower

```
Class ConquestBot

Algoritma

if (ada bot yang ke-scan) then
   e = bot tersebut
   arahAwal = gunBearingTo(e.X, e.Y) // arah ke koordinat bot

   relative = e.arahBot - arahBot

   // menebak arah bot akan bergerak relatif terhadap kita.
Gerakkan tembakan ke arah tersebut. Semakin cepat, semakin besar
correction dari arahAwal

   if (90 <= relative && relative <= 270) // kuadran 2 atau 3
       correction = e.Speed + relative/distance
   else
       correction = -(e.Speed + relative/distance)

   SetTurnGunLeft(arahAwal + relative) //bidik…

   // kekuatan tembakan sesuai dengan jarak terhadap bot, dan
kecepatan

   if (distance > 300) then
       bulletPower = 0.5
   else if (distance >= 200) then
       bulletPower = 1
   else if (distance >= 100) then
       bulletPower = 2
   else if (distance < 100 || e.Speed < 1)
       bulletPower = 3
   else bulletPower = 0

   SetFire(bulletPower) //tembak!
```

c. Custom

Greedy by Position

```
Class Custom

Algoritma

TurnRight(Direction % 90) // belok melihat tembok (bebas)
Forward(arenaWidth) // maju hingga kena tembok

while (bot berjalan)
{ while (belum di tembok){
  Forward(250) // maju
  onwall = OnWall() // cek posisi dekat tembok

  if (onwall) then
     TurnGunRight(180)
     TurnRIght(90) // penyesuaian tank
```

```
     whichwall = WhichWall //bool identifikasi tembok

  Movement() //fungsi pergerakan bot
  TurnGunLeft(180)
  TurnGunRIght(180) //putar gun

Function OnWall() -> bool
  if (bot dalam margin dekat tembok) then
     -> true
  else
     -> false

Function WhichWall() -> int
  if (bot dekat tembok kiri)
     -> 1
  else if (bot dekat tembok kanan)
     -> 2
  else if (bot dekat tembok bawah)
     -> 3
  else if (bot dekat tembok atas)
     -> 4
   else
    -> 0

Procedure Movement(int whichwall)
   if (DistanceRemaining == 0) // gerakan selesai
      moveCounter ++;
      moveDistance = 60 + (moveCounter % 5) * 60 //jarak gerak
   if (moveCounter ganjil)
      if (akan menabrak tembok)
         moveDistance = -moveDistance // ganti arah
      SetForward(moveDistance) //maju sebanyak moveDistance
   else // moveCounter genap
      if (akan menabrak tembok)
         moveDistance = -moveDistance // ganti arah
      SetBack(moveDistance) //mundur sebanyak moveDistance
```

Greedy by Firepower

```
Class Custom

Algoritma

if (ada bot yang ke-scan) then
   distance = DistanceTo(e.X, e.Y) // jarak ke bot
   firepower = Max(0,5, Min(3 , 500/distance))
   // sesuaikan firepower sesuai jarak lawan, dari 0.5 hingga 3

   Fire(firepower) //tembak sesuai bobot firepower!

if (Bot menabrak bot lain) then
   if (onwall = false) then //belum sampai tembok
      Fire(2) //tembak bot yang menghalangi
```

d. Box

Greedy by Positioning

```
Class Box

Algoritma

Rectangle box
padding = 20
boxWidth = 250
boxHeight = 200
// X dan Y adalah posisi bot saat ini

if (X < ArenaWidth / 2) then
     if (Y < ArenaHeight / 2) then
         box = new Rectangle(padding, padding,
              boxWidth, boxHeight)
         corner = 3
         dir = 1
     else
         box = new Rectangle(padding,
              ArenaHeight - padding - boxHeight, boxWidth,
              boxHeight)
         corner = 2
         dir = 1
else
     if (Y < ArenaHeight / 2) then
          box = new Rectangle(ArenaWidth - padding - boxWidth,
               padding, boxWidth, boxHeight)
          corner = 4
          dir = -1
     else
          box = new Rectangle(ArenaWidth - padding - boxWidth,
               ArenaHeight - padding - boxHeight, boxWidth,
               boxHeight)
          corner = 1
          dir = -1
GoCorner()

Procedure GoCorner()
    double angle, movement;
    if (corner == 1) then
        angle = arctan(ArenaHeight - padding - Y,
            ArenaWidth - padding - X)
    else if (corner == 2) then
        angle = 180 - arctan(ArenaHeight - Y - padding,
            X - padding)
    else if (corner == 3) {
        angle = 180 + arctan(Y - padding, X - padding)
    else
        angle = 360 - arctan(Y - padding,
            ArenaWidth - padding - X)
    TurnLeft(CalcBearing(angle))
    Forward(1000)
```

Greedy by Movement

```
Class Box

Algoritma
```

```
procedure AdjustMovement()
   if (currentHeading == 90) then // facing right wall
       movement = boxWidth;
   else if (currentHeading == 180) then // facing top wall
       movement = boxHeight;
   else if (currentHeading == 270) then // facing left wall
       movement = boxWidth;
   else // facing bottom wall
       movement = boxHeight;

SetTurnRadarLeft(Positive_Infinity)
AdjustMovement()

if (DistanceRemaining == 0) then
    TurnLeft(CalcBearing(currentHeading + 90))
    Forward(movement)

if (TurnRemaining == 0) then
    currentHeading = (currentHeading + 90) % 360
```

Greedy by Firepower

```
Class Custom

Algoritma

if (ada bot yang ke-scan) then
   distance = DistanceTo(e.X, e.Y) // jarak ke bot
   firepower = Max(0,5, Min(3 , 500/distance))
   // sesuaikan firepower sesuai jarak lawan, dari 0.5 hingga 3

   Fire(firepower) //tembak sesuai bobot firepower!

if (Bot menabrak bot lain) then
   if (onwall = false) then //belum sampai tembok
      Fire(2) //tembak bot yang menghalangi
```

B. Implementasi Kode Bot

a. Perpendicular

- Atribut

Atribut	Deskripsi
int state	Menyimpan aktivitas bot saat ini 0 = Scanning 1 = Chasing
int count	Jumlah bot musuh yang sudah discan
int dir	Arah pergerakan bot 1 = maju

	-1 = mundur
int closestBotId	Menyimpan ID bot musuh terdekat
int lastBotId	menyimpan ID bot musuh yang terakhir diserang
double closestBotDistance	menyimpan jarak terhadap bot terdekat
double radarTurn	Sudut pemutaran radar agar lock pada bot musuh
double extraTurn	Overshoot pada pemutaran radar agar tidak kehilangan musuh akibat pergerakan musuh
double gunTurn	Sudut pemutaran senjata agar lock juga pada bot musuh
double Turn	sudut antara bot dengan bot musuh, digunakan untuk kalkulasi pergerakan
double enemyLateralSpeed	Menghitung kecepatan lateral musuh terhadap bot, dan digunakan untuk menyesuaikan sudut tembakan
double Overcompensate	Kompensasi tambahan untuk menyesuaikan belokan dengan gerakan musuh
double MaxSpeed	Kecepatan maksimum pergerakan bot

- Metode

Metode	Deskripsi
void SetTurnRadarRight(double angle)	Gerakkan *radar* ke kanan menuju derajat yang ditentukan
private void	Mengejar musuh berdasarkan

ChaseEnemy(ScannedBotEvent e)	hasil pemindaian ketika bot terdekat terpindai.
void SetTurnLeft(double angle)	Gerakkan badan ke kiri menuju derajat yang ditentukan
void SetTurnGunLeft(double angle)	Gerakkan *gun* ke kiri menuju derajat yang ditentukan
void SetForward(double distance)	Memajukkan bot dengan jarak yang ditentukan
void SetBack(double distance)	Memundurkan bot dengan jarak yang ditentukan
void SetFire(double firepower)	Menembakkan peluru dengan kekuatan yang ditentukan
double DistanceTo(double X, double Y)	Menentukan jarak antara titik (X, Y) dan posisi bot
double GunBearingTo(double X, double Y)	Menentukan sudut relatif *gun* terhadap titik (X, Y)
double BearingTo(double X, double Y)	Menentukan sudut relatif bot terhadap titik (X, Y)
double Math.Sin(double angle)	Menentukan perbandingan sinus dari sudut yang ditentukan
double Math.Atan(double ratio)	Menentukan sudut arctan dari rasio yang diberikan
double Math.Min(double x, double y)	Mengembalikan nilai terkecil dari dua double
double Math.Max(double x, double y)	Mengembalikan nilai terbesar dari dua double

b. ConquestBot

- Atribut

Atribut	Deskripsi
int enemyID	menentukan target yang akan dikejar.

	Default = -1
double enemyY	menentukan koordinat Y target yang dikejar. Default = 0
double enemyX	menentukan koordinat X target yang dikejar. Default = 0
bool AdjustGunForBodyTurn	menentukan apakah rotasi *gun* terpengaruh oleh rotasi *body*. Pada bot ini, dijadikan *True* agar terpisah. Default = False
bool AdjustRadarForBodyTurn	menentukan apakah rotasi *radar* terpengaruh oleh rotasi *body*. Pada bot ini, dijadikan *True* agar terpisah. Default = False
bool AdjustRadarForGunTurn	menentukan apakah rotasi *radar* terpengaruh oleh rotasi *gun*. Pada bot ini, dijadikan *True* agar terpisah. Default = False
int TargetSpeed	kecepatan tank. Bot ini menggunakan kecepatan 8, yaitu maksimum.

- Metode

Metode	Deskripsi
double Distance(double x1, double y1, double x2, double y2)	Memberikan jarak antara dua titik berdasarkan koordinat
double RadarBearingTo(double x, double y)	Memberikan perbedaan sudut dari *radar* menuju koordinat tersebut
double GunBearingTo(double x,	Memberikan perbedaan sudut

double y)	dari *gun* menuju koordinat tersebut
double BearingTo(double x, double y)	Memberikan perbedaan sudut dari *body* menuju koordinat tersebut
double Math.Min(double x, double y)	Mengembalikan nilai terkecil dari dua double
double Math.Atan(double d)	Mengembalikan arctan dari rasio d
void SetTurnGunLeft(double angle)	Gerakkan *gun* ke kiri menuju derajat yang ditentukan
void SetTurnRadarLeft(double angle)	Gerakkan *radar* ke kiri menuju derajat yang ditentukan
void SetTurnLeft(double angle)	Gerakkan badan ke kiri menuju derajat yang ditentukan
void SetFire(double BulletPower)	Menembak peluru sesuai dengan kekuatan yang ditentukan

c. Custom

- Atribut

Atribut	Deskripsi
bool onwall	menentukan apakah bot sudah di tembok atau belum
double moveAmount	jarak pergerakan bot
int whichwall	sisi tembok mana bot berada 1 = kiri 2 = kanan 3 = bawah 4 = atas
int moveCounter	berapa kali gerakan sudah dilakukan, digunakan untuk memvariasikan pergerakan
double margin	margin antara bot dan tembok

	yang dianggap sudah di tembok
double botX/botY	posisi bot
double arenaWidth/arenaHeight	lebar/panjang arena
double distance	Menyimpan jarak yang diperlukan
double firepower	Digunakan untuk menyimpan kalkulasi kekuatan tembakan

- Metode

Metode	Deskripsi
void Forward(double x)	maju sejauh double
void TurnGunLeft(double angle)	Gerakkan *gun* ke kiri sebanyak sudut yang ditentukan
void TurnGunRight(double angle)	Gerakkan *gun* ke kanan sebanyak sudut yang ditentukan
void Movement(int whichwall)	Menggerakan bot maju mundur, dengan wall avoidance sesuai whichwall
void TurnRight(double angle)	Gerakkan bot ke kanan sebanyak sudut yang ditentukan
bool OnWall()	Mengirimkan true jika bot berada dekat wall, false jika tidak.
void SetForward(double dist)	Gerakkan bot maju sebanyak dist yang ditentukan
void SetBack(double dist)	Gerakkan bot mundur sebanyak dist yang ditentukan
int WhichWall()	Menentukan sisi tembok lokasi bot sekarang 1 = kiri 2 = kanan 3 = bawah

	4 = atas

d. Box

- Atribut

Atribut	Deskripsi
const double deg_to_rad/rad_to_deg	Konstanta untuk konversi antara derajat dan radian
const int boxWidth/boxHeight	Konstanta untuk area pergerakan bot
const int GUN_FACTOR	Konstanta yang menyesuaikan pembidikan pada bot
int corner	Menyimpan sudut lokasi bot pada saat ini
Dictionary<int,int> enemyLocations	Menyimpan dictionary dari estimasi lokasi setiap bot musuh dengan estimasi posisi sudut bot. 0 = area tengah 1 = sudut kanan atas 2 = sudut kiri atas 3 = sudut kiri bawah 4 = sudut kanan bawah
int closestBotId	Menyimpan ID bot terdekat yang terdeteksi
double closestBotDistance	Jarak antara bot dengan bot musuh terdekat
const int padding	Margin dari sudut arena, digunakan untuk memposisikan bot
double currentHeading	menyimpan sudut Heading bot saat ini
double movement	menyimpan jarak pergerakan yang dibutuhkan untuk pergerakan bot

int dir	arah pergerakan, sesuai dengan sudut yang dipilih.
int state	Menyimpan aktivitas bot pada saat ini (menembak, melakukan scan)
int count	Jumlah bot musuh yang sudah discan
int cornerChange	Menghitung berapa kali bot sudah mengubah posisi sudut
Rectangle box	Menyesuaikan daerah pergerakan bot pada sudut yang ditempati

- Metode

Metode	Deskripsi
void AdjustMovement()	Menyesuaikan jarak pergerakan berdasarkan posisi dan heading bot pada box
void GoCorner()	Memindahkan bot pada sudut sesuai dengan nilai pada corner
PointDouble(double x, double y)	Konstruktor PointDouble
PointDouble(PointDouble p)	Copy Konstruktor PointDouble
double distance(PointDouble p)	Menghitung jarak dari bot dan target.

C. Pengujian Bot

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Conquest 1.0	1689	350	0	772	69	449	48	3	2	1
2	Undang-Undang34 1.0	1633	750	90	716	65	11	0	2	4	1
3	Custom 1.0	1294	700	90	433	38	32	0	2	1	3
4	Box 1.0	387	250	0	115	0	22	0	0	0	2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Conquest 1.0	1776	300	0	788	81	542	64	3	3	1
2	Undang-Undang34 1.0	1773	800	120	755	79	18	0	3	3	2
3	Custom 1.0	975	450	90	359	18	56	0	3	0	1
4	Box 1.0	718	550	0	141	4	23	0	0	1	5

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Undang-Undang34 1.0	1640	650	60	828	83	18	0	2	4	0
2	Custom 1.0	1465	600	60	594	48	163	0	2	1	2
3	Conquest 1.0	1142	150	0	486	22	460	24	1	2	2
4	Box 1.0	1112	650	60	335	37	30	0	2	0	3

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Undang-Undang34 1.0	1251	500	60	618	54	18	0	3	1	0
2	Conquest 1.0	1162	250	0	482	31	342	57	1	2	0
3	Custom 1.0	957	350	30	424	51	101	0	1	2	2
4	Box 1.0	867	650	60	145	0	11	1	1	1	4

D. Pembahasan Hasil Pengujian

Bot Perpendicular (Main-Bot)

Berdasarkan hasil pengujian, bot Perpendicular efektif melawan berbagai tipe bot, karena memanfaatkan pergerakan bot yang diincar, sehingga untuk bot yang mengejar, ia dapat menjaga jarak, sedangkan untuk bot yang relatif diam pada satu area tertentu, ia dapat mengelilinginya. Selain itu, dengan pembidikan prediktif, ia dapat memanfaatkan arah dan kecepatan bot untuk mengira-kira arah penembakan yang paling efektif. Hal ini terbukti dengan skor Bullet Damage dan Bullet Bonusnya yang tinggi.

Bot Conquest

Bot Conquest sangat kuat melawan bot yang memiliki mobilitas terbatas, mampu memaksimalkan skor dengan damage yang sangat tinggi. Namun, ketika melawan bot dengan gerakan yang berusaha menjauh, disertai gerakan melingkar, bot Conquest biasanya tinggal memiliki energi yang sedikit ketika mencapai tujuannya. Selain itu, ketika melakukan *penabrakan*, bot ini stasioner, sehingga sangat rentan terhadap serangan oleh bot lainnya. Menurut percobaan juga, dapat dilihat bahwa poin yang didapatkan bot Conquest tidak konsisten, dan sangat mengandalkan RNG penempatan di awal ronde.

Bot Custom

Bot Custom karena memiliki perintah yang lebih statis dibandingkan bot lain, cenderung kurang fleksibel pada saat pertarungan dengan bot jumlah kecil. Namun, bot custom masih mempunyai *survivability* dan pencapaian poin yang masih bisa bersaing bila pertandingan dilakukan dengan jumlah bot yang banyak. Posisi *spawn*

dari Bot Custom juga sangat berpengaruh terhadap hasil akhir, sehingga hasil setiap ronde cenderung fluktuatif.

Bot Box

Terakhir, bot Box terbukti kurang efektif dari hasil pengujian ini karena kurang dapat melawan bot-bot yang mengejar dan jarak dekat. Hal yang menarik adalah bahwa bot Box seringkali berakhir pada posisi 1st atau terakhir (3rd, 4th) tiap rondenya. Hal ini dikarenakan pada kasus posisi terakhir, bot Box diincar oleh bot-bot jarak dekat terlebih dahulu, sehingga tidak bisa kabur ke pojokan atau pindah ke pojokan yang lebih aman. Namun, ketika bot Box dapat kabur terlebih dahulu, ia bisa bertahan dan karena bot lawan yang tersisa sudah cukup rendah energinya, ia dapat menghabisinya karena selisih energi yang tinggi.

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Algoritma *greedy* dapat diaplikasikan untuk berbagai masalah, salah satunya pada pembuatan bot pada robocode. Aplikasi greedy pada robocode berfokus pada variabel-variabel yang tersedia secara langsung dan mengambil tindakan sesuai dengan masukan yang diterima. Dalam pengerjaan, pendekatan greedy dari berbagai sudut juga dicoba, seperti *Greedy by firepower, Greedy by position, Greedy by movement, Greedy by Distance, dan Greedy by targeting.* Solusi optimal lokal yang dicari menggunakan algoritma greedy sudah cukup baik dan berhasil membuat bot-bot yang dapat beradaptasi terhadap situasi, walaupun tidak mencapai hasil maksimum dengan pendekatan solusi global. Projek robocode juga membantu kami dalam memahami tentang penerapan dan aplikasi algoritma greedy secara langsung.

B. Saran

Beberapa saran dari kelompok kami untuk pengerjaan selanjutnya adalah:

- Mengalokasikan lebih banyak waktu pada testing perilaku robot, untuk menghindari *unintended behaviour.*
- Melakukan pengujian bersama lebih sering, agar dapat menguji kelemahan bot dengan bot yang lebih *advanced.*
- Mengatur waktu dengan lebih baik, khususnya dengan membuat timeline yang lebih konkrit dan tidak sporadis.

LAMPIRAN

https://youtu.be/eZjxJzH-9Tc - Video Penjelasan Program
https://github.com/koinen/Tubes1_UndangUndang34 - Github

DAFTAR REFERENSI

https://ineedmoreplates.medium.com/greedy-algorithms-5364fd15a04f
https://robowiki.net/wiki/Main_Page
https://robocode-dev.github.io/tank-royale/