

**Tugas Besar 1 IF2211 Strategi Algoritma
Semester II tahun 2024/2025**

**Pemanfaatan Algoritma *Greedy* dalam pembuatan *bot* permainan
*Robocode Tank Royale***



Oleh:

Buege Mahara Putra	13523037
Abrar Abhirama Widyadhana	13523038
Jethro Jens Norbert Simatupang	13523081

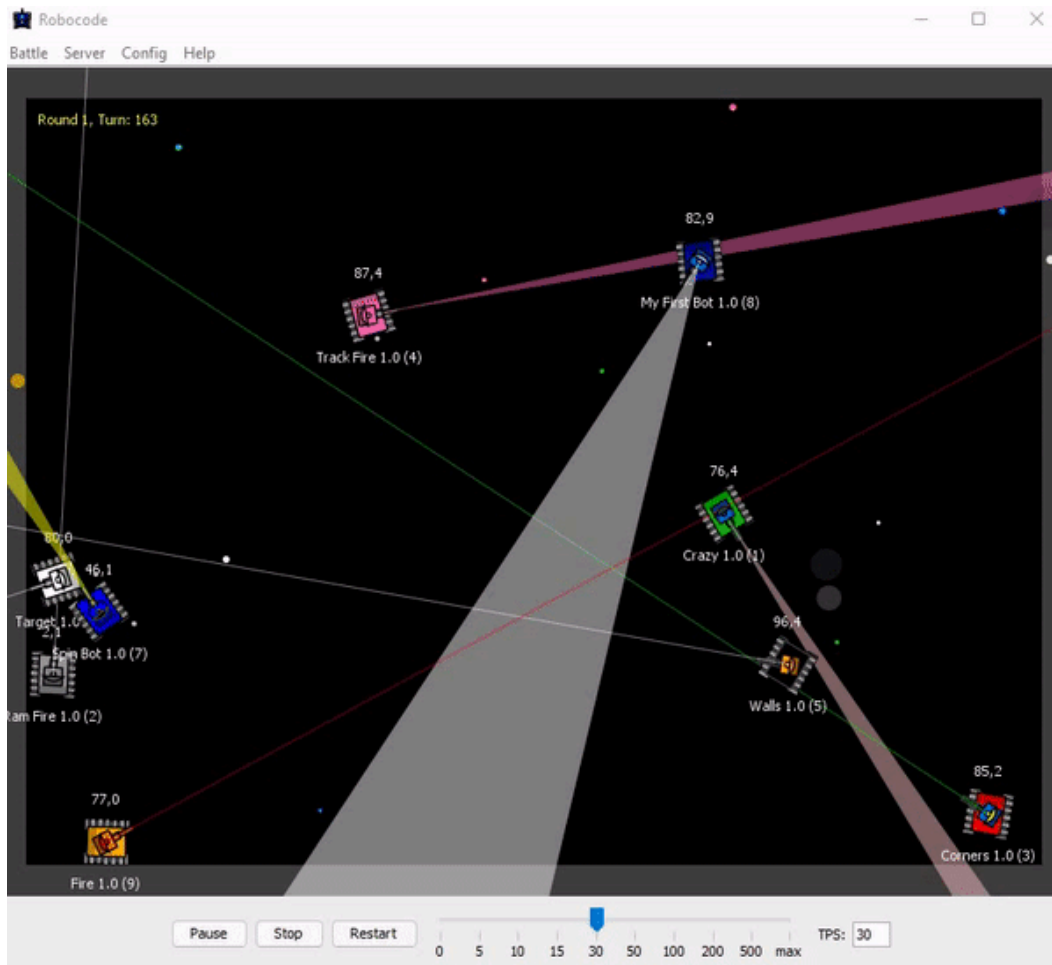
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

DAFTAR ISI

DAFTAR ISI	2
BAB 1	3
BAB 2	9
2.1 Algoritma Greedy	9
2.2 Cara Kerja Program	10
BAB 3	11
3.1 Proses Mapping Persoalan <i>Robocode Tank Royale</i>	11
3.2 Eksplorasi Alternatif Solusi Greedy	12
3.3 Analisis Efisiensi dan Efektivitas	15
3.4 Strategi Greedy yang Dipilih	16
BAB 4	19
4.1 Pseudocode	19
4.2 Penjelasan Struktur Data, Fungsi, dan Prosedur	25
4.3 Pengujian	29
4.4 Analisis Hasil	30
BAB 5	32
5.1 Kesimpulan	32
5.2 Saran	32
LAMPIRAN	33
DAFTAR PUSTAKA	34

BAB 1

DESKRIPSI TUGAS



Gambar 1 *Robocode Tank Royale*

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. Nama *Robocode* adalah singkatan dari "Robot code," yang berasal dari [versi asli/pertama permainan ini](#). *Robocode Tank Royale* adalah evolusi/versi berikutnya dari permainan ini, di mana bot dapat berpartisipasi melalui Internet/jaringan. 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, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan **strategi greedy** dalam membuat bot ini.

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

Pertempuran dapat terdiri dari beberapa rounds. Secara default, satu pertempuran berisi 10 rounds, di mana setiap rounds akan memiliki pemenang dan yang kalah.

Setiap round dibagi menjadi beberapa turns, yang merupakan unit waktu terkecil. Satu turn adalah satu ketukan waktu dan satu putaran permainan. Jumlah turn dalam satu round tergantung pada berapa lama waktu yang dibutuhkan hingga hanya tersisa bot terakhir yang bertahan.

Pada setiap turn, sebuah bot dapat:

- Menggerakkan bot, memindai musuh, dan menembakkan senjata.
- Bereaksi terhadap peristiwa seperti saat bot terkena peluru atau bertabrakan dengan bot lain atau dinding.
- Perintah untuk bergerak, berputar, memindai, menembak, dan sebagainya dikirim ke server untuk setiap turn.

Perlu diperhatikan bahwa [API \(Application Programming Interface\)](#) bot resmi secara otomatis mengirimkan niat bot ke server di balik layar, sehingga Anda tidak perlu mengkhawatirkannya, kecuali jika Anda membuat API Bot sendiri.

Pada setiap turn, bot akan secara otomatis menerima informasi terbaru tentang posisinya dan orientasinya di medan perang. Bot juga akan mendapatkan informasi tentang bot musuh ketika mereka terdeteksi oleh pemindai.

Perlu diketahui bahwa game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round & Turns.

2. Batas Waktu Giliran

Penting untuk dicatat bahwa setiap bot memiliki batas waktu untuk setiap turn yang disebut turn timeout, biasanya antara 30-50 ms (dapat diatur sebagai aturan pertempuran). Ini berarti bahwa bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini.

Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan melewatkan turn tersebut. Jika bot melewatkan turn, ia tidak akan bisa menyesuaikan gerakannya atau menembakkan senjatanya karena server tidak menerima perintah tepat waktu sebelum turn berikutnya dimulai.

3. Energi

Semua bot memulai permainan dengan jumlah energi awal sebanyak 100 poin energi.

- Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot musuh.
- Bot juga akan kehilangan energi jika menembakkan meriamnya.
- Bot akan mendapatkan energi jika peluru dari meriamnya mengenai musuh. Energi yang didapat akan lebih banyak 3 kali lipat dari energi yang digunakan untuk menembakkan peluru.
- Bot dengan energi nol akan dinonaktifkan dan tidak bisa bergerak. Jika bot terkena serangan dalam keadaan ini, bot akan hancur.

4. Peluru

Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

Seperti disebutkan sebelumnya, peluru yang lebih berat akan bergerak lebih lambat. Ini berarti akan membutuhkan waktu lebih lama untuk mencapai target, meningkatkan risiko peluru tidak mengenai sasaran. Sebaliknya, peluru yang lebih ringan bergerak lebih cepat, sehingga lebih mudah mengenai target, tetapi peluru ringan tidak memberikan banyak poin energi saat mengenai bot musuh.

5. Panas Meriam (Gun Heat)

Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan perlu waktu untuk mendingin sebelum bisa digunakan untuk pertama kalinya.

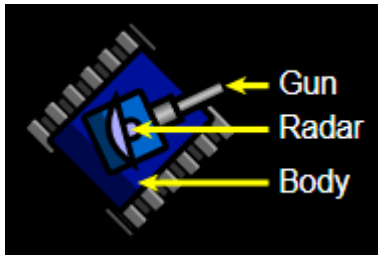
6. Tabrakan

Perlu diperhatikan bahwa bot akan menerima kerusakan jika menabrak dinding (batas arena), yang disebut wall damage. Hal yang sama juga terjadi jika bot bertabrakan dengan bot lain.

Jika bot menabrak bot musuh dengan bergerak maju, ini disebut ramming (menabrak dengan sengaja), yang akan memberikan sedikit skor tambahan bagi bot yang menyerang.

7. Bagian Tubuh Tank

Tubuh tank terdiri dari 3 bagian:



Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.

Gun digunakan untuk menembakkan peluru dan dapat berputar bersama *body* atau independen dari *body*.

Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama *body* atau independen dari *body*.

8. Pergerakan

Bot dapat bergerak maju dan mundur hingga kecepatan maksimum. Dibutuhkan beberapa giliran untuk mencapai kecepatan maksimum. Bot dapat mengalami percepatan maksimum sebesar 1 unit per giliran dan pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

Seperti yang disebutkan sebelumnya, bagian tubuh, turret (meriam), dan radar dapat berputar secara independen satu sama lain. Jika turret atau radar tidak diputar, maka keduanya akan mengarah ke arah yang sama dengan tubuh bot.

Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran.

Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

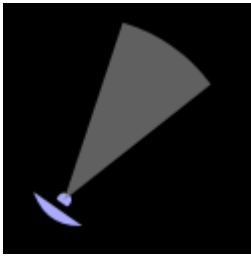
Perlu diperhatikan bahwa tidak ada energi yang dikonsumsi saat bot bergerak atau berbelok.

10. Pemindaian

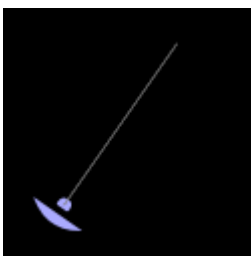
Aspek penting dalam *Robocode* adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh radar.

Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini

merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran.



Jika radar tidak bergerak dalam suatu giliran, artinya radar tetap mengarah ke arah yang sama seperti pada giliran sebelumnya, maka sudut pemindaian akan menjadi nol derajat, dan bot tidak akan dapat mendeteksi musuh.



Oleh karena itu, sangat disarankan untuk selalu mengubah arah radar agar tetap dapat memindai musuh.

11. Skor

Pada akhir pertempuran, setiap bot akan diranking berdasarkan total skor yang diperoleh masing-masing bot selama keseluruhan pertempuran. Tentunya, tujuan utama pada tugas besar ini adalah membuat bot yang memberikan skor setinggi mungkin. Berikut adalah rincian komponen skor pada pertempuran:

- **Bullet Damage:** Bot mendapatkan **poin sebesar *damage*** yang dibuat kepada bot musuh menggunakan peluru.
- **Bullet Damage Bonus:** Apabila peluru berhasil membunuh bot musuh, bot mendapatkan **poin sebesar 20% dari *damage*** yang dibuat kepada musuh yang terbunuh.
- **Survival Score:** Setiap ada bot yang mati, bot lainnya yang masih bertahan pada ronde tersebut mendapatkan **50 poin**.
- **Last Survival Bonus:** Bot terakhir yang bertahan pada suatu ronde akan mendapatkan **10 poin** dikali dengan banyaknya musuh.
- **Ram Damage:** Bot mendapatkan **poin sebesar 2 kalinya *damage*** yang dibuat kepada bot musuh dengan cara menabrak.
- **Ram Damage Bonus:** Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan **poin sebesar 30% dari *damage*** yang dibuat kepada musuh yang terbunuh.

Skor akhir bot adalah akumulasi dari 6 komponen diatas. Perlu diperhatikan bahwa game akan menampilkan berapa kali suatu bot meraih peringkat 1, 2, atau 3 pada setiap ronde. Namun, hal ini tidak dihitung sebagai komponen skor maupun untuk perankingan akhir. Bot yang dianggap menang pertempuran adalah bot dengan akumulasi skor tertinggi.

Untuk informasi lebih lengkap, silahkan buka dokumentasi Tank Royale pada link [berikut](#).

Starter Pack

Untuk tugas besar ini, game engine yang akan digunakan sudah dimodifikasi oleh asisten. Berikut adalah beberapa perubahan yang dibuat oleh asisten dari game engine Tank Royale:

- **Theme GUI:** diubah menjadi light theme
- **Skor & Energi:** masing-masing bot ditampilkan di samping arena permainan agar lebih mudah diamati saat pertarungan
- **Turn Limit:** Durasi pertarungan tidak akan bergantung pada banyak ronde. Pada game engine ini, pertarungan akan berakhir apabila banyaknya turn sudah mencapai batas tertentu. Apabila batasan ini tercapai, ronde otomatis langsung berakhir dan pemenang pertarungan akan ditampilkan. Turn Limit dapat diatur pada menu “setup rules”.

Source Code untuk game engine dan template bot telah disediakan pada tautan berikut.

[tubes1-if2211-starter-pack](#)

Adapun panduan mengenai cara menjalankan game engine, membuat bot, dan melihat referensi API dapat dilihat melalui tautan berikut.

 [Get Started With Robocode](#)

BAB 2

LANDASAN TEORI

2.1 Algoritma *Greedy*

Algoritma *Greedy* merupakan algoritma yang paling umum dan mudah dalam memecahkan persoalan optimasi. Algoritma ini memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga pada setiap langkah, algoritma mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan pengoptimalan keseluruhan solusi yang akan terbentuk (prinsip “*take what you can get now!*”) dan “berharap” bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Algoritma *Greedy* memiliki karakteristik, yaitu bersifat heuristik, memilih solusi optimal secara lokal di setiap tahapnya, tidak menyimpan hasil sebelumnya atau melihat ke depan untuk langkah-langkah selanjutnya, dan mengembalikan nilai minimum atau maksimum pada setiap langkahnya. Dalam mengembangkan algoritma *Greedy*, terdapat 6 elemen penting yang harus dimengerti:

- Himpunan kandidat, C: kandidat yang akan dipilih pada setiap langkah
- Himpunan solusi, S: kandidat yang sudah dipilih
- Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
- Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi *greedy* tertentu
- Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih layak untuk dimasukkan ke dalam himpunan solusi
- Fungsi objektif: memaksimalkan atau meminimumkan

2.2 Cara Kerja Program

Secara umum, setiap bot dikendalikan oleh kode yang menentukan pergerakan, penargetan musuh, dan penembakan menggunakan prinsip *Greedy*.

Aksi utama bot:

1. Scanning (*Radar Movement*):
 - Setiap bot memiliki strateginya sendiri dalam memutar radarnya.
 - Jika menemukan musuh, bot menyimpan koordinatnya untuk digunakan dalam pengambilan keputusan aksi berikutnya.
2. Penargetan (*Gun Movement*):
 - Bot menyesuaikan sudut meriam ke arah musuh berdasarkan posisi terbaru.
 - Jika musuh berada dalam jarak tembak yang efektif, bot akan menembak dengan daya tembak yang ditentukan sesuai strategi yang ditetapkan.
3. Pergerakan (*Body Movement*):

- Bot bergerak dengan pola yang berbeda-beda dengan tujuan menghindari peluru musuh sambil disesuaikan dengan strategi serangan setiap bot.
4. Tanggapan Terhadap Peristiwa (*Events*):
- Jika bot menabrak tembok, bot akan mengambil tindakan sesuai strateginya untuk menghindari tembok.
 - Jika bot terkena tembakan, bot akan mencoba untuk tidak terkena peluru berikutnya.
 - Jika bot menabrak musuh, ia akan menyerang atau mundur tergantung strategi bot.

Algoritma *Greedy* diterapkan dalam setiap aksi ini untuk menentukan keputusan yang terbaik. Keputusan tersebut ditentukan berdasarkan kondisi bot dan sekitarnya pada saat itu (sesuai dengan prinsip algoritma *Greedy*).

BAB 3

APLIKASI STRATEGI *GREEDY*

3.1 Proses Mapping Persoalan *Robocode Tank Royale*

Dalam mengimplementasikan algoritma *Greedy* pada bot *Robocode Tank Royale*, kita dapat menganalisis keenam elemen algoritma *Greedy* terlebih dahulu:

- Himpunan kandidat, C
Himpunan kandidat terdiri dari semua kemungkinan aksi yang bisa diambil bot dalam suatu kondisi tertentu. Aksi ini mencakup perintah seperti maju atau mundur dengan jarak tertentu, berputar arah dengan sudut tertentu, mengatur radar, menembak dengan kekuatan tertentu, dan lain-lain.
- Himpunan solusi, S:
Himpunan solusi adalah sekumpulan aksi yang dipilih dari setiap kemungkinan aksi yang ada (subset dari himpunan kandidat). Himpunan solusi setiap bot berbeda-beda sesuai dengan heuristik dan strategi masing-masing bot.
- Fungsi solusi:
Fungsi solusi menentukan apakah aksi-aksi dalam himpunan kandidat dapat dipilih untuk mencapai tujuan yang diinginkan. Pada perancangan bot *Robocode Tank Royale*, kriteria fungsi solusi ini bisa berupa keberhasilan bot menghindari tembok dan serangan musuh, keberhasilan radar dalam memindai musuh, keberhasilan peluru untuk mengenai musuh yang ditarget, keberhasilan bot dalam manajemen energi, dan keberhasilan bot dalam bertahan hidup.
- Fungsi seleksi (selection function):
Fungsi seleksi dalam perancangan bot *Robocode Tank Royale* adalah pemilihan aksi terbaik berdasarkan strategi *Greedy* tertentu, misalnya:
 - Dalam pergerakan bot, algoritma *Greedy* dapat berupa pergerakan yang menghindari tembok dan serangan musuh semaksimal mungkin.
 - Dalam pemindaian, algoritma *Greedy* dapat berupa radar yang berputar secara terus-menerus agar bot terus mendeteksi musuh sebanyak mungkin atau radar yang bertahan pada suatu bot agar terus mendeteksi bot yang sama secara berulang.
 - Dalam penembakan peluru, algoritma *Greedy* dapat berupa penembakan semua musuh yang terdeteksi secara membabi buta agar mendapatkan poin sebanyak mungkin, penembakan pada musuh terdekat untuk memastikan peluru terkena target, ataupun penembakan pada musuh yang sama secara terus-menerus agar mendapatkan bonus eliminasi.
- Fungsi kelayakan (feasible):
Fungsi kelayakan memeriksa apakah aksi yang diambil bot dapat tereksekusi sesuai rencana, misalnya memeriksa apakah bot tidak menabrak tembok terus-menerus, memeriksa apakah pergerakan bot mudah diprediksi musuh, memeriksa apakah peluru bot efektif mengenai musuh, dan lain-lain.

- Fungsi objektif:
Fungsi objektif menentukan tujuan utama yang ingin dicapai bot melalui maksimasi dan minimasi. Misalnya, minimasi damage yang diterima, maksimasi efisiensi energi, dan maksimasi poin yang diterima.

3.2 Eksplorasi Alternatif Solusi *Greedy*

- Alternatif Solusi 1
 - Strategi Pergerakan (Run Method)
 - Bot bergerak berdasarkan jarak ke musuh. Jika terlalu dekat (kurang dari 150 unit), bot akan menjauh untuk menghindari serangan langsung. Jika terlalu jauh (lebih dari 400 unit), bot akan mendekat agar dapat menyerang dengan lebih akurat. Jika dalam jarak menengah, bot akan bergerak mengorbit musuh untuk menjaga keseimbangan antara menghindari serangan dan menjaga posisi menyerang.
 - Heuristik: Pergerakan ini membuat bot dapat menjaga jarak optimal meningkatkan efisiensi serangan sekaligus mengurangi risiko terkena tembakan. Pergerakan melingkar membuat bot lebih sulit menjadi target.
 - Memindai Musuh (handleRadar Method & OnScannedBot Method)
 - Bot memindai keseluruhan arena dan mencari musuh dengan jarak terdekat. Setelah mendeteksi musuh terdekat, ID musuh akan dijadikan target dan serangan difokuskan pada musuh terdekat tersebut.
 - Heuristik: Dengan memfokuskan serangan pada musuh terdekat, serangan akan lebih mudah mengenai target secara terus-menerus dan lebih berpeluang untuk mengeliminasi target.
 - Menyerang Musuh (OnScannedBot Method & handleGun Method)
 - Bot menyesuaikan daya tembak berdasarkan jarak ke musuh, yaitu daya tembak 3 untuk jarak < 200 unit, daya tembak 2 untuk 200 - 400 unit, dan daya tembak 1 untuk > 400 unit.
 - Heuristik: Bot menggunakan daya tembak lebih besar pada jarak dekat untuk memastikan mengeliminasi musuh secepatnya dan memaksimalkan perolehan poin. Sebaliknya, bot menggunakan daya tembak kecil pada jarak jauh untuk mengenai peluru sebanyak-banyaknya.
 - Menanggapi Tabrakan dengan Musuh (OnHitBot Method)
 - Jika menabrak musuh, bot akan menilai energi musuh, yaitu menabrak musuh dan menembak dengan daya maksimum jika musuh memiliki energi rendah dan bot memiliki energi cukup.
 - Heuristik: Menyerang musuh yang memiliki energi rendah saat energi bot cukup bisa memaksimalkan peluang bot mendapatkan poin eliminasi sambil memastikan bot aman.
 - Menghindari dinding (OnHitWall Method)
 - Jika menabrak dinding, bot akan berbelok 60° dan mengubah arah pergerakan.

- Heuristik: Mengutamakan langkah tercepat untuk keluar dari posisi macet. Pergantian arah membantu bot kembali ke arena pertempuran tanpa kehilangan terlalu banyak waktu.
- Alternatif Solusi 2
 - Strategi Pergerakan (Run Method)
 - Bot berputar 360° sambil bergerak maju 100 unit dengan kecepatan maksimum 7. Pola ini dilakukan dalam loop dengan tiga iterasi, memberikan fleksibilitas dalam mendeteksi musuh dan menghindari peluru musuh.
 - Heuristik: Pola ini memaksimalkan area pengamatan radar dan mempertahankan pergerakan agar tidak menjadi target diam.
 - Memindai Musuh (Run Method & *TurnToFaceTarget* Method)
 - Bot melakukan putaran radar dan mengunci target pertama yang terdeteksi. Penguncian dilakukan dengan menghitung sudut yang diperlukan agar bot bisa menghadapkan badan dan meriam ke target.
 - Heuristik: Dengan penguncian target, bot meminimasi kemungkinan musuh untuk kabur dan meningkatkan kesempatan untuk mengeliminasi target.
 - Menyerang Musuh (OnScannedBot Method)
 - Bot mengunci radar dan meriam ke arah target yang terdeteksi. Bot akan menembak dengan daya 3 jika jarak < 200, menembak dengan daya 2 jika jarak 200-400, dan menembak dengan daya 1 jika jarak > 400.
 - Heuristik: Serangan berbasis jarak ini memaksimalkan akurasi dan efektivitas tembakan, dengan daya tinggi pada target yang lebih dekat.
 - Menanggapi Tabrakan dengan Musuh (OnHitBot Method)
 - Jika bot menabrak musuh (IsRammed), bot menyesuaikan arah meriam agar tetap menghadap musuh. Jika musuh memiliki energi rendah (≤ 30 , ≤ 20 , ≤ 10), dan energi bot mencukupi, bot langsung menembak dengan daya maksimum. Setelah menembak, bot maju 40 unit untuk menabrak musuh.
 - Heuristik: Strategi ini memastikan bahwa bot memanfaatkan peluang untuk menyerang musuh yang lemah, sambil tetap bergerak untuk menghindari serangan balik.
 - Menghindari Dinding (OnHitWall Method)
 - Jika menabrak dinding, bot mundur 100 unit dan mencoba berbelok dengan sudut acak (-50° ke kiri dan -50° ke kanan).
 - Heuristik: Dengan mundur dan berbelok, bot menghindari terjebak di dinding dan memastikan bahwa ia tidak kembali ke arah yang sama.
- Alternatif Solusi 3
 - Strategi Pergerakan (Run Method)
 - Pergerakan seperti gelombang slinki, yaitu bot mengatur pola gerakan melingkar-lingkar dengan pola berbelok 45° ke kiri, maju 100 unit, berbelok 90° ke kanan, dan maju 100 unit.
 - Heuristik: Pergerakan melingkar dapat menghindari peluru secara efektif. Pergerakan melingkar dapat mengurangi kemungkinan bot terkena tembakan lurus yang menargetkannya karena bot terus melakukan belokan.

Selama bergerak, bot juga terus memutar radarnya ke seluruh arena sehingga pemindaian dilakukan secara menyeluruh tanpa mengunci pada satu target tertentu. Hal ini bertujuan untuk memperoleh poin sebanyak-banyaknya dari menembak bot musuh.

- Memindai Musuh (Run Method & OnScannedBot Method)
 - Bot memindai keseluruhan arena untuk mencari bot tanpa terkecuali.
 - Heuristik: Bot memaksimalkan serangan ke semua musuh tanpa terkecuali untuk memaksimalkan poin yang didapat dari peluru yang terkena target.
- Menyerang Musuh (OnScannedBot Method)
 - Menentukan firepower yang disesuaikan dengan jarak.
 - Heuristik: Jika musuh lebih dekat, bot akan menggunakan daya tembak yang lebih besar untuk memperoleh poin maksimal. Sebaliknya, jika musuh jauh, bot akan menggunakan daya tembak yang lebih kecil agar dapat mengenai musuh dengan mudah.
- Menghindari Serangan (OnHitByBullet Method)
 - Berbelok acak ($90^\circ \pm 30^\circ$) dan maju acak ($150 + \text{random}(50)$).
 - Heuristik: Bot mengubah posisi secepat mungkin untuk mengurangi kemungkinan ditembak di posisi yang sama ataupun menjadi target bot musuh.
- Menanggapi Tabrakan dengan Musuh (OnHitBot Method)
 - Jika menabrak musuh, bot akan mundur 50 unit dan berputar 30° untuk menghindari kebuntuan. Selain itu, jika GunHeat = 0 dan energi mencukupi, bot akan menembak.
 - Heuristik: Bot mundur sedikit dan berputar untuk mencari posisi yang lebih aman dan menembak segera untuk memanfaatkan kesempatan mendapatkan poin.
- Menghindari Dinding (OnHitWall Method)
 - Mundur 50 unit dan berputar 90° untuk segera keluar dari posisi yang macet.
 - Heuristik: Mengutamakan langkah tercepat untuk keluar dari dinding tanpa mempertimbangkan posisi optimal berikutnya.
- Alternatif Solusi 4
 - Strategi Pergerakan (Run Method)
 - Bot bergerak dalam pola zig-zag yang tetap, dengan urutan: memutar ke kiri 200° , maju 200 unit, memutar ke kanan 200° , dan maju sejauh 200 unit.
 - Heuristik: Pola pergerakan ini memberikan gerakan yang statis, tetapi cukup variatif sehingga bot tidak menjadi target mudah bagi musuh yang mencoba menembaknya dengan prediksi gerak lurus. Selain itu, bot juga terus memutar radar untuk melakukan pemindaian ke seluruh arena secara menyeluruh.
 - Memindai Musuh (Run Method & OnScannedBot Method)
 - Bot memindai keseluruhan arena untuk mencari bot tanpa terkecuali.
 - Heuristik: Bot memaksimalkan serangan ke semua musuh tanpa terkecuali untuk memaksimalkan poin yang didapat dari peluru yang terkena target.

- Menyerang Musuh (OnScannedBot Method)
 - Saat bot mendeteksi musuh, ia akan menghitung sudut yang diperlukan untuk mengarahkan senjata ke musuh, mengubah arah turret menghadap ke musuh, menentukan daya tembak berdasarkan jarak ke musuh, yaitu daya tembak 3 jika jarak < 100 , daya tembak 2 jika jarak $100 - 200$, dan daya tembak 1 jika jarak > 200 .
 - Heuristik: Menembak dengan daya tinggi saat musuh berada dekat meningkatkan peluang mengenai target dan mengeliminasi musuh lebih cepat. Saat musuh lebih jauh, bot menggunakan daya lebih kecil untuk menghemat energi dan meningkatkan kesempatan peluru mengenai target.
- Menanggapi Tabrakan dengan Musuh (OnHitBot Method)
 - Jika bot bertabrakan dengan musuh, maka bot akan menghindari musuh dengan belok ke kiri 50° dan ke kanan 50° sambil mengarahkan turret ke arah musuh dan menembak dengan daya maksimum.
 - Heuristik: Bot menghindari musuh untuk mengurangi risiko kehilangan energi lebih lanjut dari serangan keberlanjutan musuh. Bot melakukan ini sambil menembaki musuh yang menabraknya untuk mendapat poin tambahan.
- Menghindari Dinding (OnHitWall Method)
 - Jika bot menabrak dinding, ia akan memutar ke kiri sejauh 100° , lalu memutar ke kanan sejauh 100° .
 - Heuristik: Gerakan ini memungkinkan bot segera keluar dari dinding tanpa perlu mundur, sehingga menghindari risiko terkena serangan.

3.3 Analisis Efisiensi dan Efektivitas

Untuk menganalisis efisiensi masing-masing bot, dilakukan uji 1 vs 1 melawan bot template sebanyak satu round, sedangkan untuk menganalisis efektivitas masing-masing bot, dilakukan uji 1 vs 1 vs 1 vs 1 antara bot alternatif solusi sebanyak lima pertandingan.

- Alternatif Solusi 1

Secara teori, bot ini memiliki keseimbangan yang cukup baik dalam pola bertahan dan pola menyerangnya. Bot ini memiliki pola serangan yang efektif karena ia berfokus pada satu bot sambil mengorbit bot tersebut sehingga bot yang ditarget akan mengalami kesulitan dalam menghindari tembakan bot. Di samping itu, pola bertahan bot yang mempertahankan jarak aman juga sangat efektif jika diterapkan dalam strategi mengorbit ini karena bot yang ditarget akan kesulitan menyerang balik. Strategi respons bot terhadap tabrakan dengan dinding ataupun bot musuh pun juga baik karena bot bersikap agresif sambil juga menjaga posisi dirinya agar aman.
- Alternatif Solusi 2

Secara teori, bot ini dapat menjadi bot yang mematikan karena ia mengunci target dan tidak akan menggeser radar selama bot yang ditarget masih di dalam jangkauan radar sehingga pola serangan bot ini bisa dibilang efektif dan dapat meningkatkan kesempatan bot untuk mendapatkan poin eliminasi. Bot ini juga memiliki pola

pergerakan yang cukup baik, walaupun mungkin terkesan statis. Di samping itu, respons bot terhadap dinding terlihat efisien dan dapat menjauhkan bot dari dinding secepat mungkin.

- Alternatif Solusi 3

Secara teori, pola gerakan bot ini dapat dibilang sangat efektif dalam menghindari serangan musuh. Gerakan yang berputar-putar dapat secara efektif menghindari peluru yang sifatnya lurus dan gerakan tidak statis mengurangi risiko bot untuk menjadi target bot musuh. Di samping itu, strategi respons bot terhadap peluru musuh, tabrakan dengan musuh, dan tabrakan dengan dinding juga dapat dikatakan efektif karena masing-masing strategi tersebut mencari cara yang paling efisien untuk mencegah bot kehilangan energi lebih jauh. Namun, strategi penembakan bot yang membabi buta dapat menjadi tidak efektif apabila bot terus menyerang musuh yang sulit diincar.

- Alternatif Solusi 4

Secara teori, pola gerakan bot ini mungkin dapat terlihat mudah diprediksi karena pergerakan zig-zag sifatnya lurus. Namun, jika dianalisis, pergerakan ini dapat secara efektif mengurangi risiko bot menjadi target musuh karena bot terus bergerak secara dinamis. Bot ini memiliki strategi menghindari tabrakan musuh dan tabrakan dinding yang juga efektif dan efisien karena memprioritaskan pergerakan paling cepat dalam menghindari tabrakan berkelanjutan.

3.4 Strategi *Greedy* yang Dipilih

Dari empat alternatif solusi yang telah dibuat, alternatif solusi yang dipilih adalah alternatif solusi 1. Hal ini didasari oleh beberapa alasan:

- Optimasi Jarak untuk Menyerang dan Menjauhi Target

Strategi ini sangat baik diterapkan dalam pertempuran karena bot selalu memastikan dirinya berada dalam jarak yang optimum. Bot dapat menjaga keseimbangan antara menyerang dan bertahan, berbeda dengan alternatif solusi lain yang hanya berfokus pada pola gerakan yang sulit diprediksi tanpa memperhitungkan jarak yang optimum untuk menyerang musuh.

- Pemilihan Target yang Efektif

Dengan memprioritaskan musuh terdekat, bot dapat meningkatkan kemungkinan eliminasi musuh secara cepat. Serangan yang mengorbit dan berfokus pada target yang paling dekat juga meningkatkan akurasi dan efisiensi peluru, serta menyulitkan musuh untuk menargetnya. Hal ini lebih unggul dari bot lainnya yang mengunci target tanpa pertimbangan jarak atau bahkan menyerang secara membabi buta.

- Respons terhadap Tabrakan dengan Musuh

Jika bot menabrak musuh, ia akan mengevaluasi energi musuh. Jika musuh memiliki energi rendah dan bot memiliki cukup energi, bot akan memanfaatkan kesempatan untuk menyerang dengan daya maksimum. Strategi ini unggul dibanding bot lain yang hanya berusaha untuk kabur tanpa mempertimbangkan serangan agresif jika energi musuh sedikit.

- Efektivitas dalam Menghindari Dinding

Dengan berbelok 60° saat menabrak dinding, bot dapat dengan cepat kembali ke pertempuran tanpa kehilangan momentum. Strategi ini lebih efektif dibandingkan dengan mundur atau belokan acak yang berpotensi membuat bot kehilangan posisi strategis.

Dengan demikian, dibandingkan dengan alternatif solusi yang lain, strategi ini menawarkan kombinasi terbaik antara pertahanan dan agresivitas sehingga memberikan keseimbangan yang ideal untuk memenangkan pertempuran dengan lebih konsisten.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Pseudocode

- main_bot

```
Program main_bot

using System
using System.Drawing
using Robocode.TankRoyale.BotApi
using Robocode.TankRoyale.BotApi.Events

procedure Main(input args: array[] of string)
    main_bot().Start()
end

{Konstruktor bot}
BotInfo ← FromFile("main_bot.json")

double eX, eY, eSpeed, eFacing, eDistance
eX ← 0, eY ← 0, eSpeed ← 0, eFacing ← 0, eDistance ← double.MaxValue;
integer directionToMove, eID
directionToMove ← 1, eID ← -1;

procedure Run()
    {Mengatur warna bot}
    BodyColor ← Color.Black
    GunColor ← Color.Black
    RadarColor ← Color.Black
    TracksColor ← Color.Black
    TurretColor ← Color.Black
    ScanColor ← Color.Black
    {Memastikan radar, turret, dan badan bisa bergerak independen}
    AdjustGunForBodyTurn ← true
    AdjustRadarForGunTurn ← true
    AdjustRadarForBodyTurn ← true
    {Loop utama bot}
    while IsRunning do
        handleRadar()
        handleGun()
        handleMove()
        Go()
    end
end

procedure OnScannedBot(input e: ScannedBotEvent)
    scannedDistance → DistanceTo(e.X, e.Y)
    {Jika belum ada target atau target lebih dekat, perbarui data
musuh}
    if eID = -1 or scannedDistance < eDistance - 20 then
        eID ← e.ScannedBotId
        eX ← e.X
        eY ← e.Y
        eDistance ← scannedDistance
    else if eID = e.ScannedBotId then
```

```

        eX ← e.X
        eY ← e.Y
        eDistance ← scannedDistance
    end
end

procedure OnHitWall(input e: HitWallEvent)
    output("ouch wall")
    directionToMove ← directionToMove * -1
    SetTurnRight(60) {Berputar menjauhi dinding}
end

procedure OnHitBot(input e: HitBotEvent)
    if e.IsRammed then
        depend on (e.Energy)
            {Menembak musuh sesuai kriteria energi}
            e.Energy ≤ 10 and Energy > 30 :
                TurnToFaceTarget(e.X, e.Y)
                SetFire(3)
                Forward(40)
            e.Energy ≤ 20 and Energy > 40 :
                TurnToFaceTarget(e.X, e.Y)
                SetFire(3)
                Forward(40)
            e.Energy ≤ 30 and Energy > 65 :
                TurnToFaceTarget(e.X, e.Y)
                SetFire(3)
                Forward(40)
        end
    end
end

procedure OnBotDeath(input e: BotDeathEvent)
    if e.VictimId = eID then
        eID ← -1 {Menghapus target yang dilock saat sudah tereliminasi}
    end
end

procedure handleRadar()
    SetTurnRadarLeft(360) {Mencari target}
end

procedure handleGun()
    firepower ← 0
    {Daya tembak sesuai kriteria jarak dan energi}
    depend on (eDistance)
        eDistance < 200 : firepower ← min(3.0, Energy * 0.5)
        eDistance < 400 : firepower ← 2.0
        else : firepower ← 1.0
    gunTurn ← NormalizeRelativeAngle(DirectionTo(eX, eY) -
GunDirection)
    SetTurnGunLeft(gunTurn)
    if GunHeat = 0 and Energy > firepower then
        SetFire(firepower)
    end
end

procedure handleMove()
    bodyTurn ← DirectionTo(eX, eY) - Direction
    moveForward ← 0
    depend on (eDistance)

```

```

        eDistance < 150 : {Menjauh jika terlalu dekat}
            bodyTurn ← bodyTurn + 90
            moveForward ← -100
        eDistance > 400 : {Mendekat jika terlalu jauh}
            moveForward ← 100
        else : {Mengorbit musuh jika jarak pas}
            bodyTurn ← bodyTurn + 90
            moveForward ← 50
        SetTurnLeft(NormalizeRelativeAngle(bodyTurn))
        SetForward(moveForward * directionToMove)
    end

    procedure TurnToFaceTarget(input x: double, input y: double)
        {Mengunci target}
        angleToTarget ← DirectionTo(x, y)
        gunTurn ← NormalizeRelativeAngle(angleToTarget - GunDirection)
        bodyTurn ← NormalizeRelativeAngle(angleToTarget - Direction)
        SetTurnGunLeft(gunTurn)
        SetTurnLeft(bodyTurn)
    end
end

```

- alt_bot_1

```

Program alt_bot_1

using System
using System.Drawing
using Robocode.TankRoyale.BotApi
using Robocode.TankRoyale.BotApi.Events

integer targetBotId
targetBotId ← -1; {ID musuh yang menjadi target}
boolean radarSwingRight
radarSwingRight ← true; {Menentukan arah gerakan radar}

procedure Main(input args: array[] of string)
    alt_bot_1().Start()
end

{Konstruktor bot}
BotInfo ← FromFile("alt_bot_1.json")

procedure Run()
    {Mengatur warna bot}
    BodyColor ← Color.White
    GunColor ← Color.White
    RadarColor ← Color.White
    TracksColor ← Color.White
    TurretColor ← Color.White
    ScanColor ← Color.White
    {Menyesuaikan gerakan turret dan radar}
    AdjustRadarForGunTurn ← false;
    AdjustGunForBodyTurn ← true;
    while(IsRunning) do
        for i ← 0 to 2 do
            SetTurnGunRight(double.PositiveInfinity);
            SetTurnLeft(360)
            MaxSpeed ← 7
            {Tetap bergerak (dan berputar)}
        end
    end
end

```

```

        Forward(100)
    end
end
end

procedure OnHitBot(input e: HitBotEvent)
    if e.IsRammed then {Jika bot menabrak musuh}
        double bearing ← BearingTo(e.X, e.Y) {Menghitung sudut ke
musuh}
        SetTurnGunLeft(bearing) {Mengarahkan turret ke musuh}
        {Menyerang sesuai kriteria energi bot dan musuh}
        depend on (e.Energy, Energy)
            e.Energy ≤ 30, Energy > 65 :
                SetFire(3)
                Forward(40)
            e.Energy ≤ 20, Energy > 40 :
                SetFire(3)
                Forward(40)
            e.Energy ≤ 10, Energy > 30 :
                SetFire(3)
                Forward(40)
        end
    end
end

procedure OnScannedBot(input e: ScannedBotEvent)
    {Mengunci target}
    double gunTurn ← GunBearingTo(e.X, e.Y)
    SetTurnGunRight(2)
    if radarSwingRight then
        SetTurnGunRight(gunTurn + 100)
        SetTurnGunRight(gunTurn + double.PositiveInfinity)
    else
        SetTurnGunRight(gunTurn - 100)
        SetTurnGunRight(gunTurn - double.PositiveInfinity)
    end
    radarSwingRight ← not radarSwingRight
    double distance ← DistanceTo(e.X, e.Y)
    {Menembak sesuai energi bot dan musuh}
    depend on (distance)
        distance < 200 :
            SetFire(Math.Min(3.0, Energy * 0.5))
        distance < 400
            SetFire(2)
        else :
            SetFire(1)
    end
end

procedure OnHitWall(input e: HitWallEvent)
    SetForward(-100)
    SetTurnLeft(-50)
    SetTurnRight(-50)
end

procedure TurnToFaceTarget(input x: double, input y: double)
    double angleToTarget ← DirectionTo(x, y) {Ambil arah absolut ke
target}
    double gunTurn ← NormalizeRelativeAngle(angleToTarget -
GunDirection) {Sesuaikan arah meriam}

```

```

        double bodyTurn ← NormalizeRelativeAngle(angleToTarget - Direction)
    {Sesuaikan arah tank}
        SetTurnGunLeft(gunTurn)
        SetTurnLeft(bodyTurn) {Tank ikut menghadap target juga}
    end

```

- alt_bot_2

Program alt_bot_2

```

using System
using System.Drawing
using Robocode.TankRoyale.BotApi
using Robocode.TankRoyale.BotApi.Events

procedure Main(input args: array[] of string)
    alt_bot_2().Start()
end

{Konstruktor bot}
BotInfo ← FromFile("alt_bot_2.json")

procedure Run()
    {Mengatur warna bot}
    BodyColor ← Color.Yellow
    TurretColor ← Color.Black
    RadarColor ← Color.Yellow
    ScanColor ← Color.White
    {Memastikan radar, turret, dan badan bisa bergerak independen}
    AdjustRadarForGunTurn ← true
    AdjustGunForBodyTurn ← true
    AdjustRadarForBodyTurn ← true
    {Pola gerakan utama bot}
    while IsRunning do
        SetTurnRadarLeft(360) {Memutar radar untuk mencari musuh}
        SetTurnLeft(45)
        SetForward(100)
        SetTurnRight(90)
        SetForward(100)
        Go()
    end
end

procedure OnScannedBot(input e: ScannedBotEvent)
    output("I see a bot at " + e.X + ", " + e.Y)
    distance ← DistanceTo(e.X, e.Y)
    {Menyesuaikan kekuatan tembakan berdasarkan jarak musuh}
    depend on (distance)
        distance < 200 : firepower ← Min(3.0, Energy * 0.5)
        distance < 400 : firepower ← 2.0
        else : firepower ← 1.0
    {Mengarahkan senjata ke target}
    gunTurn ← NormalizeRelativeAngle(DirectionTo(e.X, e.Y) -
GunDirection)
    SetTurnGunLeft(gunTurn)
    {Menembak jika senjata tidak panas dan energi mencukupi}
    if GunHeat = 0 and Energy > firepower then
        SetFire(firepower)
    end
end

```

```

end

procedure OnHitByBullet(input e: HitByBulletEvent)
    {Menghindar setelah terkena tembakan dengan gerakan acak}
    random ← Random()
    SetTurnRight(90 + random.Next(-30, 30))
    SetForward(150 + random.Next(50))
end

procedure OnHitBot(input e: HitBotEvent)
    output("Ouch! I hit a bot at " + e.X + ", " + e.Y)
    if e.IsRammed then
        SetBack(50) {Mundur agar tidak terjebak}
        SetTurnRight(30) {Memutar sedikit untuk menghindari stuck}
    end
    if GunHeat = 0 and Energy > 1.0 then
        SetFire(1.0)
    end
    Go()
end

procedure OnHitWall(input e: HitWallEvent)
    output("Ouch! I hit a wall, must turn back!")
    {Mundur dan berbelok setelah menabrak dinding}
    Stop()
    SetBack(50)
    SetTurnRight(90)
    Go()
end

```

- alt_bot_3

```

Program alt_bot_3

    using System
    using System.Drawing
    using Robocode.TankRoyale.BotApi
    using Robocode.TankRoyale.BotApi.Events

    integer targetBotId
    targetBotId ← -1 {Menyimpan ID bot target}

    procedure Main(input args: array[] of string)
        alt_bot_3().Start()
    end

    {Konstruktor bot}
    BotInfo ← FromFile("alt_bot_3.json")

    procedure Run()
        {Mengatur warna bot}
        BodyColor ← Color.Yellow
        GunColor ← Color.Yellow
        RadarColor ← Color.Yellow
        TracksColor ← Color.Yellow
        TurretColor ← Color.Yellow
        ScanColor ← Color.Yellow
        {Membuat turret dan radar berputar bersama}
        AdjustGunForBodyTurn ← true
        AdjustRadarForGunTurn ← true
    end

```

```

AdjustRadarForBodyTurn ← true
{Loop utama pergerakan dan pemindaian bot}
while IsRunning do
    SetTurnRadarRight(10_000)
    SetTurnLeft(200)
    Forward(200)
    SetTurnRight(200)
    Forward(200)
    Go()
end
end

procedure OnHitBot(input e: HitBotEvent)
    {Menghindari musuh yang menabrak atau tertabrak}
    SetTurnLeft(-50)
    SetTurnRight(50)
    TurnToFaceTarget(e.X, e.Y)
end

procedure OnScannedBot(input e: ScannedBotEvent)
    gunTurn ← NormalizeRelativeAngle(DirectionTo(e.X, e.Y) -
GunDirection)
    SetTurnGunLeft(gunTurn)
    distance ← DistanceTo(e.X, e.Y)
    {Menembak sesuai jarak musuh}
    depend on (distance)
        distance < 100 : SetFire(3)
        distance < 200 : SetFire(2)
        else SetFire(1)
            SetFire(1)
    end
end

procedure OnHitWall(input e: HitWallEvent)
    SetTurnLeft(-100)
    SetTurnRight(-100)
end

prosedur TurnToFaceTarget(input x: double, input y: double)
    {Mengunci target}
    angleToTarget ← DirectionTo(x, y)
    gunTurn ← NormalizeRelativeAngle(angleToTarget - GunDirection)
    bodyTurn ← NormalizeRelativeAngle(angleToTarget - Direction)
    SetTurnGunLeft(gunTurn)
    SetTurnRadarRight(0)
    SetFire(3)
    SetFire(2)
    SetTurnRadarRight(10_000)
end

```

4.2 Penjelasan Struktur Data, Fungsi, dan Prosedur

- main_bot
 - Struktur data:
 - double eX, double eY: posisi musuh terakhir yang terdeteksi
 - double eSpeed: kecepatan musuh

- double eFacing: arah musuh
- double eDistance: jarak ke musuh terdekat (diinisialisasi ke double.MaxValue)
- integer eID: ID musuh yang sedang ditargetkan..
- integer directionToMove: arah pergerakan bot (1 untuk maju, -1 untuk mundur)
- ScannedBotEvent e, HitWallEvent e, HitBotEvent e, BotDeathEvent e: event khusus dari API *Robocode* yang memicu tindakan bot berdasarkan situasi tertentu
- double x, double y: posisi musuh yang terdeteksi
- Fungsi dan prosedur:
 - Run()

Berfungsi untuk mengatur warna bot, mengatur radar dan turret agar tetap menghadap ke musuh saat bergerak, serta memanggil tiga metode utama secara berulang (handleRadar, handleGun, dan handleMove).
 - OnScannedBot(ScannedBotEvent e)

Berfungsi untuk menentukan musuh yang harus diserang berdasarkan jarak terdekat. Jika belum ada musuh yang dikunci (eID == -1) atau ada musuh lebih dekat, bot mengganti target. Jika musuh yang terdeteksi adalah musuh yang sudah dikunci, bot memperbarui data posisinya.
 - OnHitWall(HitWallEvent e)

Berfungsi untuk menghindari tabrakan dengan dinding. Bot akan membalik arah (directionToMove *= -1) dan berbelok 60° untuk keluar dari dinding.
 - OnHitBot(HitBotEvent e)

Berfungsi untuk menentukan apakah akan menyerang musuh saat terjadi tabrakan. Jika energi musuh rendah (< 30, < 20, < 10) dan energi bot cukup tinggi, bot akan mengarahkan turret ke arah musuh (TurnToFaceTarget(e.X, e.Y)), menembak dengan daya maksimum (SetFire(3)), dan bergerak maju untuk menabrak lebih lanjut.
 - OnBotDeath(BotDeathEvent e)

Berfungsi untuk menghapus target jika musuh yang sedang dikunci mati, yaitu jika ID musuh yang mati (e.VictimId) sama dengan eID, bot menghapus target (eID = -1).
 - handleRadar()

Berfungsi untuk memutar radar untuk mendeteksi musuh. Radar diputar 360° dengan SetTurnRadarLeft(360).
 - handleGun()

Berfungsi untuk menyesuaikan arah turret dan menembak musuh berdasarkan jarak. Bot akan menetapkan daya tembak 3 jika jarak < 200, 2 jika jarak 200 – 400, dan 1 jika jarak > 400. Bot kemudian akan menghitung sudut yang harus diputar agar turret mengarah ke musuh dan menembak musuh (SetFire(firepower)) jika senjata siap (GunHeat == 0) dan energi cukup.

- **handleMove()**
Berfungsi untuk mengontrol pergerakan bot berdasarkan jarak musuh. Bot akan menjauh jika jarak musuh < 150 , mendekat jika jarak musuh > 400 , dan mengorbit musuh jika jarak musuh di antaranya.
 - **TurnToFaceTarget(double x, double y)**
Bertujuan untuk mengarahkan turret dan badan bot ke musuh. Bot akan menghitung sudut ke target dan memutar turret (**SetTurnGunLeft**) dan badan (**SetTurnLeft**) agar mengarah ke musuh.
- **alt_bot_1**
 - Struktur data:
 - **boolean radarSwingRight**: status apakah radar bergerak ke kanan atau tidak
 - **HitBotEvent e, ScannedBotEvent e, HitWallEvent e**: event khusus dari API *Robocode* yang memicu tindakan bot berdasarkan situasi tertentu
 - **double x, double y**: posisi musuh yang terdeteksi
 - Fungsi dan prosedur:
 - **Run()**
Berfungsi sebagai prosedur utama bot yang mengatur warna bot, mengatur properti *gun* dan *radar*, dan mengontrol pergerakan. Bot berputar 360° sambil menggerakkan radar dan meriam. Bot juga bergerak maju 100 unit dalam siklus 3 kali. Pola ini meningkatkan kemungkinan menemukan musuh sambil menghindari tembakan.
 - **OnHitBot(HitBotEvent e) (Menabrak Musuh)**
Fungsi ini mengatur agar jika bot bertabrakan dengan musuh, bot akan menghadap ke arah musuh. Jika energi musuh rendah, bot akan menyerang dengan tembakan maksimal (3) dan bergerak maju 40 unit untuk melakukan tabrakan.
 - **OnScannedBot(ScannedBotEvent e)**
Pada fungsi ini bot menyesuaikan posisi *gun* untuk mengunci target. Bot mengubah arah radar setiap kali mendeteksi musuh dan menembak target dengan kekuatan berdasarkan jarak. Bot menembak dengan daya tembak 3 jika jarak < 200 , daya tembak 2 jika jarak $200 - 400$, dan daya tembak 1 jika jarak > 400 .
 - **OnHitWall(HitWallEvent e)**
Fungsi ini mengatur agar jika bot menabrak tembok, ia akan mundur 100 unit dan berbelok 50° ke kiri dan kanan untuk menghindari terjebak.
 - **TurnToFaceTarget(double x, double y)**
Fungsi ini menghitung sudut ke target dan menyesuaikan arah bot serta meriam, membantu dalam mengunci dan menyerang musuh dengan lebih akurat.
- **alt_bot_2**
 - Struktur data:
 - **ScannedBotEvent e, HitByBulletEvent e, HitBotEvent e, HitWallEvent e**: event khusus dari API *Robocode* yang memicu tindakan bot berdasarkan situasi tertentu

- Fungsi dan prosedur:
 - Run()

Berfungsi untuk mengatur warna bot, menyesuaikan rotasi radar, senjata, dan tubuh, serta mengatur pola pergerakan bot. Pola pergerakan menyerupai gelombang slinki, yaitu berbelok kiri 45°, maju 100 unit, berbelok kanan 90°, dan maju 100 unit. Loop akan terus berjalan sampai bot dihentikan.
 - OnScannedBot(ScannedBotEvent e)

Berfungsi untuk menentukan daya tembak (firepower) berdasarkan jarak, yaitu daya tembak 3 jika jarak < 200, daya tembak 2 jika jarak 200 – 400, dan daya tembak 1 jika jarak > 400. Prosedur ini juga menyesuaikan arah senjata ke target dan menembak target jika senjata tidak panas dan energi mencukupi.
 - OnHitByBullet(HitByBulletEvent e)

Jika terkena peluru, bot akan menghindar dengan berbelok secara acak antara 60° hingga 120° dan maju secara acak antara 150 hingga 200 unit.
 - OnHitBot(HitBotEvent e)

Jika bertabrakan dengan musuh, bot akan mundur 50 unit, berbelok 30° ke kanan untuk menghindari kebuntuan, dan menembak dengan daya 1 jika turret tidak panas.
 - OnHitWall(HitWallEvent e)

Jika bot menabrak dinding, bot akan mundur 50 unit, berbelok kanan 90° untuk segera keluar dari posisi macet, dan melanjutkan pergerakan kembali.
- alt_bot_3
 - Struktur data:
 - HitBotEvent e, ScannedBotEvent e, HitWallEvent e: event khusus dari API *Robocode* yang memicu tindakan bot berdasarkan situasi tertentu
 - double x, double y: posisi musuh yang terdeteksi
 - Fungsi dan prosedur:
 - Run()

Prosedur utama yang berjalan dalam loop selama bot masih aktif. Bot memutar radar terus-menerus ke kanan (SetTurnRadarRight(10_000)) untuk memindai musuh sambil bergerak dalam pola zig-zag (memutar kiri, maju, memutar kanan, maju).
 - OnScannedBot(ScannedBotEvent e)

Berfungsi untuk menyerang bot yang terdeteksi radar dengan menghitung arah tembakan dan menentukan daya tembakan berdasarkan jarak musuh yang terdeteksi, yaitu daya tembak 3 jika jarak < 100, daya tembak 2 jika jarak 100 – 200, dan daya tembak 1 jika jarak > 200.
 - OnHitBot(HitBotEvent e)

Berfungsi untuk menanggapi tabrakan dengan bot lain dengan menghindari musuh. Bot juga menembak musuh yang bertabrakan untuk mendapat poin.
 - OnHitWall(HitWallEvent e)

Berfungsi untuk menghindari dinding jika bot menabraknya. Bot akan

memutar 100° ke kiri dan 100° ke kanan untuk keluar dari posisi menempel di dinding.

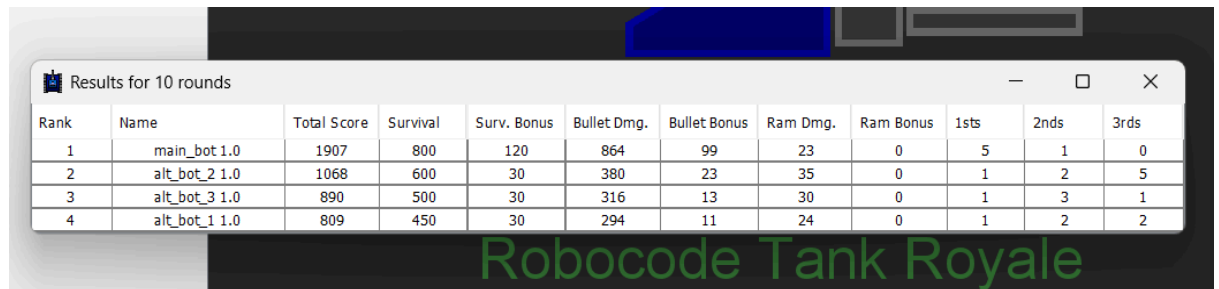
- TurnToFaceTarget(double x, double y)

Berfungsi untuk mengarahkan senjata ke target tertentu dengan menghitung sudut tembakan dan menyerang dengan dua tembakan kuat.

4.3 Pengujian

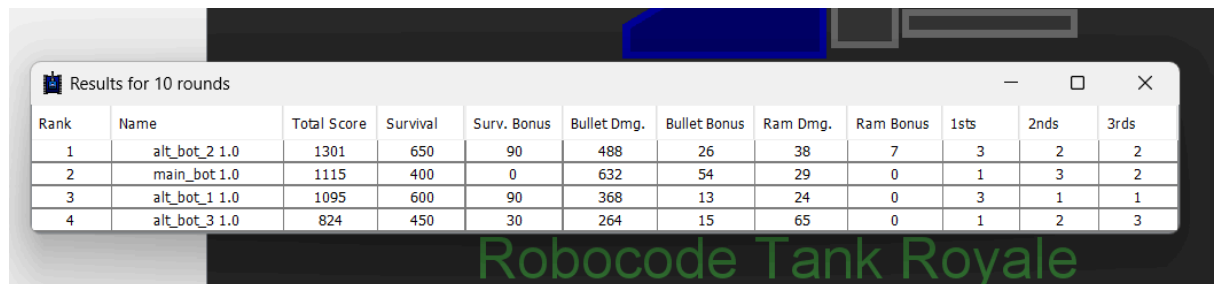
Dari 5 pengujian yang dilakukan, diperoleh hasil sebagai berikut:

- Hasil 1:



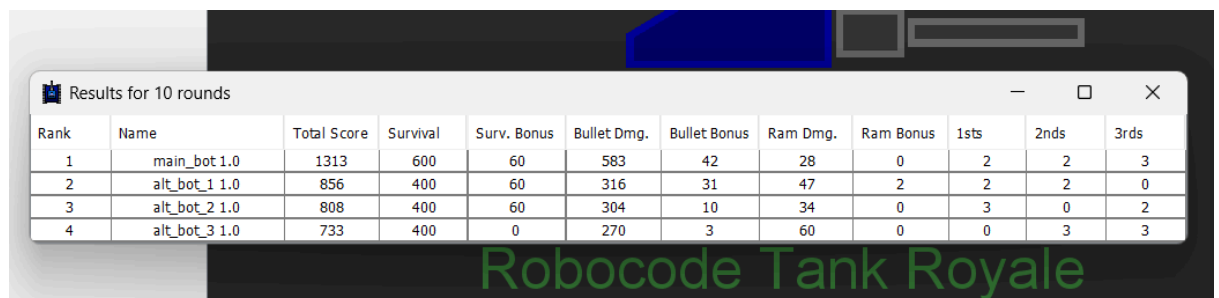
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	main_bot 1.0	1907	800	120	864	99	23	0	5	1	0
2	alt_bot_2 1.0	1068	600	30	380	23	35	0	1	2	5
3	alt_bot_3 1.0	890	500	30	316	13	30	0	1	3	1
4	alt_bot_1 1.0	809	450	30	294	11	24	0	1	2	2

- Hasil 2:



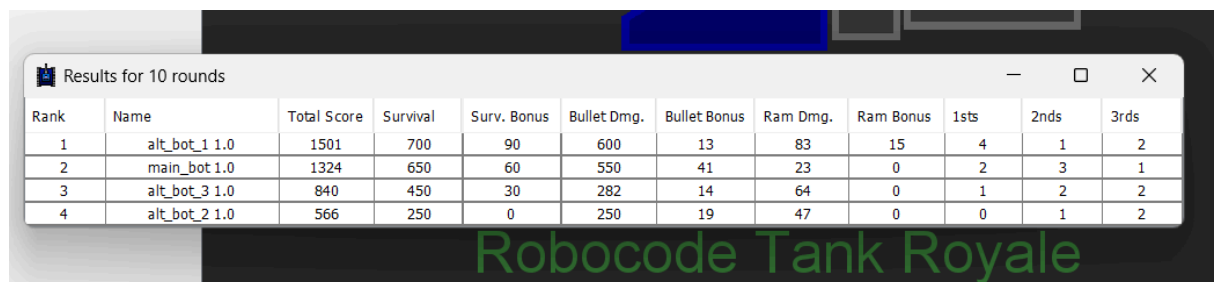
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	alt_bot_2 1.0	1301	650	90	488	26	38	7	3	2	2
2	main_bot 1.0	1115	400	0	632	54	29	0	1	3	2
3	alt_bot_1 1.0	1095	600	90	368	13	24	0	3	1	1
4	alt_bot_3 1.0	824	450	30	264	15	65	0	1	2	3

- Hasil 3:



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	main_bot 1.0	1313	600	60	583	42	28	0	2	2	3
2	alt_bot_1 1.0	856	400	60	316	31	47	2	2	2	0
3	alt_bot_2 1.0	808	400	60	304	10	34	0	3	0	2
4	alt_bot_3 1.0	733	400	0	270	3	60	0	0	3	3

- Hasil 4:



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	alt_bot_1 1.0	1501	700	90	600	13	83	15	4	1	2
2	main_bot 1.0	1324	650	60	550	41	23	0	2	3	1
3	alt_bot_3 1.0	840	450	30	282	14	64	0	1	2	2
4	alt_bot_2 1.0	566	250	0	250	19	47	0	0	1	2

- Hasil 5:

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	main_bot 1.0	1720	650	90	841	114	25	0	4	2	0
2	alt_bot_2 1.0	1303	650	90	474	41	36	12	3	1	1
3	alt_bot_1 1.0	1010	550	30	396	17	17	0	1	3	3
4	alt_bot_3 1.0	706	400	0	254	0	52	0	0	2	4

Dari hasil tersebut, dapat disimpulkan bahwa main_bot (alternatif solusi 1) mendapat kemenangan paling banyak.

4.4 Analisis Hasil

Dari hasil pengujian di atas, diperoleh hasil bahwa main_bot (alternatif solusi 1) memenangkan pertandingan sebanyak 3 kali, alt_bot_1 (alternatif solusi 2) memenangkan pertandingan sekali, dan alt_bot_2 (alternatif solusi 3) juga memenangkan pertandingan sekali. Hasil ini mungkin seperti menunjukkan ketidakkonsistenan main_bot. Namun, jika dilihat lebih lanjut, dalam setiap pertandingan yang tidak dimenangkan oleh main_bot, bot ini tetap menduduki peringkat kedua.

Saat setiap round diteliti lebih jauh, ternyata pada setiap round yang tidak dimenangkan main_bot, main_bot selalu tereliminasi pertama. Hal ini terjadi karena ketiga bot lain yang kebetulan menyerang main_bot secara bersamaan sehingga ia tereliminasi pertama. Kemungkinan ini semakin diperparah dengan perilaku main_bot yang fokus menyerang satu target dengan mengorbitnya sehingga dapat mempersulit main_bot apabila diserang secara bersamaan.

Analisis ini semakin diperkuat melalui pengujian 1 vs 1 antara main_bot dengan setiap bot lainnya, yang hasilnya dapat dilihat di bawah ini:

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	main_bot 1.0	992	300	60	554	77	0	0	7	3	0
2	alt_bot_1 1.0	344	150	30	156	8	0	0	3	7	0

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	main_bot 1.0	885	300	60	456	60	8	0	7	3	0
2	alt_bot_2 1.0	501	150	30	273	28	20	0	3	7	0

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	main_bot 1.0	850	250	50	480	70	0	0	6	3	0
2	alt_bot_3 1.0	412	150	30	212	19	1	0	3	6	0

Dalam pengujian tersebut, `main_bot` memenangkan semua pertandingan, membuktikan superioritasnya dalam pertandingan satu lawan satu. Oleh karena itu, dapat disimpulkan bahwa penyebab utama kekalahan `main_bot` dalam pertandingan sebelumnya adalah karena kebetulan ia menjadi target bersama dan langsung tereliminasi lebih awal.

Adapun bot dengan hasil terbaik kedua adalah `alt_bot_2` (alternatif solusi 3) dengan kemenangan sekali, menduduki peringkat kedua dua kali, serta menduduki peringkat ketiga dan keempat masing-masing sekali. Keunggulan `alt_bot_2` berasal dari pola pergerakannya yang sulit ditebak, menyerupai gerakan gelombang slinki, yang membuatnya sulit dikenai oleh bot lain, termasuk bot yang menggunakan strategi *targeting*.

Selanjutnya, bot dengan hasil terbaik ketiga adalah `alt_bot_1` (alternatif solusi 2) dengan kemenangan sekali, menduduki peringkat ketiga dua kali, serta menduduki peringkat kedua dan keempat masing-masing sekali. Hasil ini terjadi karena pola serangan `alt_bot_1` yang menarget satu bot spesifik sehingga meningkatkan kesempatan peluru untuk mengenai target. Namun, bot ini cukup lemah dalam menghindari peluru karena pergerakannya terbatas pada gerakan berputar sehingga saat ia menjadi target, misalnya oleh `main_bot`, `alt_bot_1` akan mengalami kesulitan dalam menghindari peluru.

Terakhir, bot dengan hasil terburuk adalah `alt_bot_3` (alternatif solusi 4) yang tidak mengalami kemenangan sekalipun. Hasil ini terjadi karena pola pergerakan `alt_bot_3` yang terbatas pada gerakan zig-zag cukup mudah diprediksi musuh.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam pengembangan bot untuk *Robocode Tank Royale*, strategi *Greedy* berperan penting dalam pengambilan keputusan setiap langkah permainan sesuai kondisi permainan pada saat itu. Bot dengan pendekatan ini selalu memilih tindakan yang memberikan keuntungan maksimal dalam kondisi saat itu, tetapi tidak mempertimbangkan dampak jangka panjang secara eksplisit. Adapun strategi *Greedy* ini dapat digunakan dalam strategi pergerakan bot, strategi pemindaian bot, strategi penyerangan, strategi menghindari musuh dan tembok, strategi manajemen energi, dan strategi lainnya yang dapat memberikan output maksimal dalam langkah tertentu.

5.2 Saran

Untuk meningkatkan efektivitas strategi *Greedy*, beberapa perbaikan dapat diterapkan:

- Adaptasi situasional: Meskipun strategi *Greedy* cukup efisien dalam kondisi statis, penyesuaian terhadap situasi dinamis perlu diterapkan agar bot tidak memiliki pola yang mudah ditebak musuh.
- Memori jangka pendek: Jika bot dibuat lebih kompleks, bot dapat dibuat menjadi bisa menyimpan data sederhana seperti lokasi terakhir musuh atau pola pergerakan sehingga bisa membantu bot memprediksi pergerakan musuh.

LAMPIRAN

1. Repository GitHub

Berikut adalah pranala ke repository GitHub yang berisi kode sumber dan dokumentasi proyek ini: https://github.com/JethroJNS/Tubes1_uyaya.git

2. Link Video YouTube: <https://youtu.be/Vey41JVb4Ug>

3. Tabel Evaluasi

No	Poin	Ya	Tidak
1	Bot dapat dijalankan pada Engine yang sudah dimodifikasi asisten.	✓	
2	Membuat 4 solusi greedy dengan heuristic yang berbeda.	✓	
3	Membuat laporan sesuai dengan spesifikasi.	✓	
4	Membuat video bonus dan diunggah pada Youtube.	✓	

DAFTAR PUSTAKA

Munir, Rinaldi. 2024. “Algoritma Greedy (Bagian 1)”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf)

Munir, Rinaldi. 2024. “Algoritma Greedy (Bagian 2)”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf)