Implementation of the Greedy Algorithm in the Bridge Card Game

Raynard Fausta - 13524052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: raynardfaustasmlone@gmail.com , 13524052@std.stei.itb.ac.id

***Abstract*—Bridge is a classic trick-taking card game requiring profound strategy. Despite its digital popularity, current applications suffer from built-in bots that exhibit naive decision-making and lack adaptive foresight. To address this, this study implements a Pure Greedy Algorithm to govern bot behavior, maximizing immediate tactical advantages. The algorithm focuses on two areas: deterministically calculating bids based on hand strength, and heuristically selecting the most advantageous card during gameplay. Results demonstrate that the greedy approach effectively automates short-term tactical maneuvers during localized phases. However, critical vulnerabilities emerge due to its lack of look-ahead capabilities and historical tracking. The bot is tactically exposed when acting as the third player. Furthermore, because it operates with a static hand evaluation without a memory state of played cards, it frequently falls into mid-game ruffing traps. In conclusion, while successful in short-term tactics, future developments must integrate basic memory structures to achieve robust strategic viability**

Keywords—Bridge Card Game, Greedy Algorithm, Game Bot, Heuristics, Trick-taking

I. Introduction

Bridge is a classic, trick-taking card game played by four individuals divided into two competing partnerships, where partners sit opposite each other. Bridge is highly regarded as a game of profound strategy, communication, and probability. In the modern digital era, Bridge has successfully transitioned into the online space. Bridge has also successfully maintained its popularity in the digital market. For instance, the mobile application Bridge - Card Game developed by Topy Games has surpassed 500,000+ downloads on the Google Play Store, maintaining an impressive 4.8-star rating from over 2,110 reviewers.

Despite this high demand, a significant weakness persists in current digital Bridge applications which is the sub-optimal performance of non-player characters (NPCs) or bots. From author's experiences and observations, the built-in bots often exhibit naive decision-making patterns. They frequently fail to play optimally, lack adaptive foresight, and struggle to execute sound strategies. In a game like Bridge, where victory heavily relies on precise cooperation and calculating risks, having an incompetent bot partner or opponent severely disrupts the immersion, leading to frustration for seasoned players who crave a genuine challenge.

To bridge this gap, the author is interested in developing an optimized decision-making model for digital Bridge game. The author proposes the implementation of a Pure Greedy Algorithm to govern the bot's behavior. This bot will be engineered to maximize immediate tactical advantages in two critical areas:

1. The Bidding Phase: Deterministically calculating the optimal number of tricks to bid based on hand strength
2. The Gameplay Phase: Selecting and playing the most advantageous card in each sequential round to secure maximum points

By implementing this approach, the bot is expected to deliver more competitive, logical, and engaging gameplay, ultimately elevating the standard of digital Bridge game.

II. Theoretical Background

2.1. Bids

Before the actual gameplay begins, players must participate in a bidding phase to determine the game's "contract." The minimum bid is a commitment for a team to win at least 7 out of the 13 possible tricks.

To overcall or challenge a previous bid, a player must declare either a higher number of tricks or the same number of tricks but in a higher-ranking suit. The official suit rankings (lowest to highest) are Clubs (♣), Diamonds (♦), Hearts (♥), Spades (♠), and No Trump (NT).

A player wins the bids when all subsequent players pass. The highest bid then becomes the official Contract. The player from the winning team who first mentioned the contract's suit becomes the Declarer. Their partner becomes the Dummy, laying their cards face-up on the table. The Declarer must now play both their own hand and the Dummy's hand to fulfill the contract, while the opposing team defends.

Figure 1. Bid Table

[Source: https://westviewbridgeclub.online/uploads/funbridge-bidding-removebg-preview.png]

2.2. Trump

The suit that secures the highest bid during the auction phase is designated as the trump suit for the game. The trump suit introduces a superseding hierarchical power over the standard suits. If a player possesses no cards matching the suit currently being played, they are permitted to play a trump card. Because the trump suit inherently outranks all other non-trump suits, playing it allows the player to capture the trick, if no subsequent opponent counters it with a higher-ranking trump card.

2.3. Card Rank

Within any given suit, the cards are ranked (lowest to highest) as follows: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack (J), Queen (Q), King (K), and Ace (A).

Figure 2. Card Rank

[Source: https://cdn.shopify.com/s/files/1/0208/0286/3168/files/Shit_Head_-_Card_Rank_Order_480x480.png?v=1645440165]

2.4. Essential Rules

1. Players are obligated to play a card matching the suit of the initial lead if they possess one in their hand
2. The initial trick of the round is commenced by the player positioned at the left of the Declarer
3. Player who captures a trick earns the right to lead the first card for the subsequent trick
4. The sequence of play progresses in a clockwise direction
5. Teams that won the bid must secure at least the number of tricks specified in their winning bid to win the round
6. The opposing team must secure a minimum number of 14 minus the Declarer's winning bid to win the round

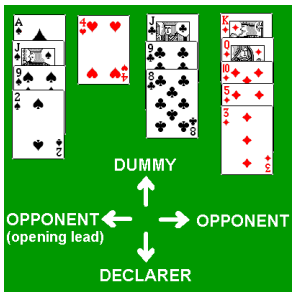

Figure 3. Initial Trick

[Source: https://www.pagat.com/images/boston/dummy.gif]

2.5. Greedy Algorithm

Algorithm that operates by selecting the most advantageous option based on the objective at every decision point (the local optimum). This algorithm operates under the assumption that those localized choices will cumulatively lead to the globally optimal solution at the end.

Greedy Algorithms consists of:

1. Candidate set contains all the potential choices or at each step
2. Solution set contains candidates that have been definitively chosen and accumulated from the previous steps
3. Solution function determines whether the current set of candidates has formed a complete solution
4. Selection function chooses the most promising candidate from the candidate heuristically
5. Feasibility function evaluates whether a selected candidate is viable and can be added to the solution set without violating any problem constraints
6. Objective function measures the value of a solution (mathematical metric) depends on what that the algorithm aims to reach (maximal or minimal)

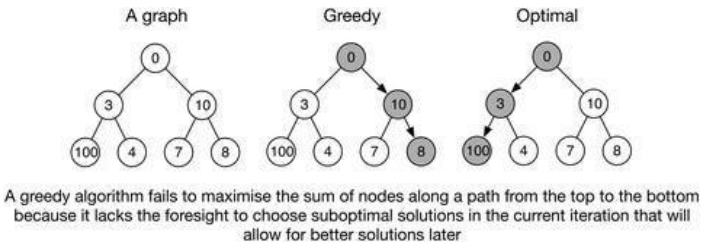

Figure 4. Greedy Algorithm

[Source: https://miro.medium.com/v2/resize:fit:960/1*1vQi9JlHWFs13IWiTjvJhQ.jpeg]

III. METHOD

For the purpose of this study, the author introduced a specific modification to the game's rules which if the user's team wins the bid, the user is automatically designated as the Declarer, and their partner will consequently default to the role of the Dummy.

3.1. User's Card Gathering

```
case INPUT_MY_HAND:
        List<String> mySel = getSelectedCardsFromGrid();
        if (mySel.size() != 13) throw new
```

```
IllegalArgumentException("Harus memilih tepat 13 kartu!");
            for (String s : mySel) {
                myHand.add(new Card(s));
                myHandSet.add(s);
            }
            log("> 13 Kartu Anda telah disimpan.");
```

At the beginning of each game, the system prompts the user to input the cards dealt to their hand. Program continuously loops and validates each entry, handling errors or invalid inputs, until exactly 13 valid cards are successfully recorded. Subsequently, the gathered hand data is evaluated by the algorithm to assist the user in formulating an optimal bid.

3.2. Hand Analysis

```
hcp = calculateHCP(myHand);
String bestSuit = suggestBestSuit(myHand);

  private int calculateHCP(List<Card> hand) {
    int p = 0;
    for (Card c : hand) {
      if (c.value == 14) p += 4;
      else if (c.value == 13) p += 3;
      else if (c.value == 12) p += 2;
      else if (c.value == 11) p += 1;
    }
    return p;
  }

  private String suggestBestSuit(List<Card> hand) {
    int s = 0, h = 0, d = 0, c = 0;

    for (Card card : hand) {
      if (card.suit == 'S') s++;
      else if (card.suit == 'H') h++;
      else if (card.suit == 'D') d++;
      else if (card.suit == 'C') c++;
    }

    int max = Math.max(Math.max(s, h), Math.max(d, c));

    if (s == max) return "Sekop (S)";
    if (h == max) return "Hati (H)";
    if (d == max) return "Wajik (D)";
    return "Keriting (C)";
  }
```

Once the cards are successfully gathered, the algorithm evaluates the hand's offensive potential through two metrics: High Card Points (HCP) and suit length. The calculateHCP method implements a standard point-count evaluation, assigning specific weights to high-ranking cards: 4 points for an Ace, 3 for a King, 2 for a Queen, and 1 for a Jack. Concurrently, the suggestBestSuit method calculates the frequency distribution of each suit within the user's hand. The algorithm identifies the longest suit and recommends it as the optimal trump suit for the upcoming bidding phase.

3.3. Bidding Suggestion

```
        case BIDDING:
            if (inputText.isEmpty()) throw new
IllegalArgumentException("Teks tidak boleh kosong.");
            log("> Teman Bid: " + inputText);

            String recommendedSuit = suggestBestSuit(myHand);

            if (inputText.equals("PASS")) {
                if (hcp >= 13) {
                    log("-> Saran Bet Bot: OPEN BID 1 " + recommendedSuit +
" (Poin Anda " + hcp + ").");
                }
                else {
                    log("-> Saran Bet Bot: PASS.");
                }
            }
            else {
                int estimatedTotalHCP = hcp + 13;
                log("-> Estimasi Total Poin Tim: ~" + estimatedTotalHCP);

                if (hcp < 6) {
                    log("-> Saran Bet Bot: PASS. Poin Anda terlalu kecil.");
                }
                else if (hcp >= 6 && hcp <= 9) {
                    log("-> Saran Bet Bot: RESPONS MINIMUM. (Dukung suit
teman di level 2, atau bid 1NT).");
                }
                else if (hcp >= 10 && hcp <= 12) {
                    log("-> Saran Bet Bot: INVITE GAME. (Dukung suit teman
di level 3, atau bid " + recommendedSuit + " di level 2).");
                }
                else {
                    log("-> Saran Bet Bot: GAME FORCING! (Misal 3NT, atau
4 " + recommendedSuit + ").");
                }
            }

            log("\n[FASE 3] KONTRAK FINAL");
            log("Ketik '1' jika Tim Kita Menyerang (Declarer), atau '2' jika
Bertahan.");
            currentPhase = GamePhase.CONTRACT_ROLE;
            inputField.setText("");
            break;
```

The algorithm acts as an advisory system that evaluates the partner's input and the user's hand strength to recommend an optimal bid. The decision mechanism is divided into two scenarios based on the partner's action:

1. Opening Bid Evaluation: If the partner passes, the system checks the user's High Card Points (HCP). If the user possesses an opening hand (HCP >= 13), the bot advises an "Open Bid" using the longest suit previously calculated. Otherwise, it advises to pass.

2. Response Evaluation: If the partner initiates a bid, the bot assumes the partner holds at least 13 HCP and estimates the team's total point potential. The algorithm then categorizes the user's hand into four distinct heuristic thresholds:
   - Pass (HCP < 6): The hand is too weak to support the partner
   - Minimum Response (6-9 HCP): Advises supporting the partner at level 2 or bidding 1 No Trump (1NT)
   - Invite Game (10-12 HCP): Advises pushing the contract to level 3 or introducing a new suit at level 2
   - Game Forcing (HCP >= 13): Advises aggressively securing a game contract (e.g., 3NT or level 4 in a major suit)

3.4. Trcik Execution

```
List<String> playSel = getSelectedCardsFromGrid();
            if (playSel.size() != 1) throw new IllegalArgumentException("Pilih tepat 1 kartu di papan!");

            String playedStr = playSel.get(0);
            Card playedCard = new Card(playedStr);
            playedCardsSet.add(playedStr);

            int currentPlayer = (leader + currentTurn) % 4;

            if (currentPlayer == KITA) myHand.removeIf(c -> c.toString().equals(playedStr));
            if (currentPlayer == TEMAN && role == 1) partnerHand.removeIf(c -> c.toString().equals(playedStr));

            log("> " + getNamaPemain(currentPlayer) + " mengeluarkan: " + playedCard);

            if (currentTurn == 0) {
               ledSuit = playedCard.suit;
               highestValue = playedCard.value;
               winningCard = playedCard;
               winningPlayer = currentPlayer;
            }
            else {
               boolean beats = false;
               if (playedCard.suit == trump && winningCard.suit != trump) beats = true;
               else if (playedCard.suit == winningCard.suit && playedCard.value > winningCard.value) beats = true;

               if (beats) {
                  winningCard = playedCard;
                  highestValue = playedCard.value;
                  winningPlayer = currentPlayer;
               }
            }

            currentTurn++;

            if (currentTurn == 4) {
               log("\n=> Pemenang Trick " + currentTrick + ": " + getNamaPemain(winningPlayer) + " (" + winningCard + ")");
               if (winningPlayer == KITA || winningPlayer == TEMAN) trickKitaMenang++;
               else trickLawanMenang++;

               leader = winningPlayer;
               currentTrick++;

               if (currentTrick > 13) {

log("\n================================================");
                  log("PERMAINAN SELESAI!");
                  log("Skor Akhir -> Kita: " + trickKitaMenang + " | Lawan: " + trickLawanMenang);

log("================================================");
                  currentPhase = GamePhase.GAME_OVER;
                  updateGridForPhase();
               }
               else {
                  startNewTrickUI();
                  promptNextPlayer();
               }
            }
            else {
               promptNextPlayer();
            }
            break;
```

This part governs the core gameplay loop, systematically managing card selection, turn progression, and trick evaluation. During each turn, the algorithm determines the active player utilizing a modulo arithmetic operation, (leader + currentTurn) % 4. Once a valid card is selected from the graphical grid, it is systematically removed from the respective player's or Dummy's hand.

The algorithm evaluates the winning card of the current trick through a deterministic conditional structure:

1. Initial Lead: The first card played (currentTurn == 0) automatically establishes the led suit and sets the baseline for the highest value and current winning card
2. Contesting the Lead: For subsequent plays, the system evaluates if the newly played card defeats the current winning card. A card successfully takes the lead if it introduces the designated trump suit (provided the current winning card is a non-trump) or if it matches the current winning card's suit but holds a higher numerical value

3.5. Greedy Suggestion

```
private void promptNextPlayer() {
      int currentPlayer = (leader + currentTurn) % 4;
      String namaPemain = getNamaPemain(currentPlayer);
      boolean isPartnerWinning = (winningPlayer == (currentPlayer + 2) % 4);

      updateGridForPhase();

      if (currentPlayer == KITA) {
         Card suggested = suggestCard(myHand, ledSuit, trump, highestValue, isPartnerWinning);
         log("\n[Giliran KITA] Bot Menyarankan: " + suggested);
         log("Silakan klik 1 kartu KITA di papan dan Konfirmasi.");
         highlightSuggestedCard(suggested);
      }
      else if (currentPlayer == TEMAN && role == 1) {
         Card suggested = suggestCard(partnerHand, ledSuit, trump, highestValue, isPartnerWinning);
         log("\n[Giliran DUMMY] Bot Menyarankan: " + suggested);
         log("Silakan klik 1 kartu DUMMY di papan dan Konfirmasi.");
         highlightSuggestedCard(suggested);
      }
      else {
         log("\n[Giliran " + namaPemain + "]");
         log("Silakan klik 1 kartu LAWAN di papan dan Konfirmasi.");
      }
   }

private Card suggestCard(List<Card> hand, char lSuit, char tSuit, int highestVal, boolean partnerWinning) {
      if (hand.isEmpty()) return null;

      if (lSuit == ' ') {
         hand.sort((a, b) -> b.value - a.value);
         return hand.get(0);
      }

      List<Card> followSuitCards = new ArrayList<>();
      for (Card c : hand) if (c.suit == lSuit) followSuitCards.add(c);

      if (!followSuitCards.isEmpty()) {
         followSuitCards.sort((a, b) -> a.value - b.value);
         if (partnerWinning) return followSuitCards.get(0);
         for (Card c : followSuitCards) {
```

```
            if (c.value > highestVal) return c;
        }
        return followSuitCards.get(0);
    }

    List<Card> trumpCards = new ArrayList<>();
    for (Card c : hand) if (c.suit == tSuit) trumpCards.add(c);

    if (!trumpCards.isEmpty() && !partnerWinning) {
        trumpCards.sort((a, b) -> a.value - b.value);
        return trumpCards.get(0);
    }

    hand.sort((a, b) -> a.value - b.value);
    return hand.get(0);
}
```

The core of the bot's decision-making process during the gameplay phase is governed by the suggestCard method, which functions as a Pure Greedy Algorithm. This function evaluates the current trick's state, specifically the led suit, trump suit, highest value played, and the partner's current winning status, to recommend the locally optimal move. The greedy heuristics are prioritized hierarchically as follows:

1. Leading a Trick: If the player initiates the trick (the led suit is empty), the algorithm greedily attempts to secure immediate dominance by selecting the absolute highest-ranking card in the hand
2. Cost Minimization (Partner Winning): If the player must follow suit and their partner is already winning the trick, the algorithm prioritizes resource preservation by selecting the lowest valid card
3. Minimal Sufficient Force (Opponent Winning): If the opponent is winning, the algorithm searches for the lowest possible card that can beat the current highest value. If no such card exists, it discards the lowest card to minimize losses
4. Opportunistic Ruffing: If the player is void of the led suit and the partner is not winning, the algorithm deploys the lowest available trump card to capture the trick efficiently
5. Asset Preservation (Discarding): If neither following suit nor ruffing is viable, the algorithm discards the absolute lowest card in the hand, preserving high-value cards for future tricks

IV. Results and Discussion

4.1. Results

4.1.1. Bid Suggestion

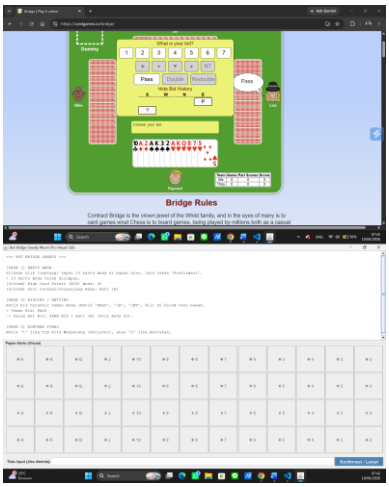

Figure 5. Bot Suggesting Heart
[Source: Author]

4.1.2. Trick Suggestion

Figure 6. Suggestion to Play King Spade
[Source: Author]

4.1.3. Suggestion to Initiate Trick

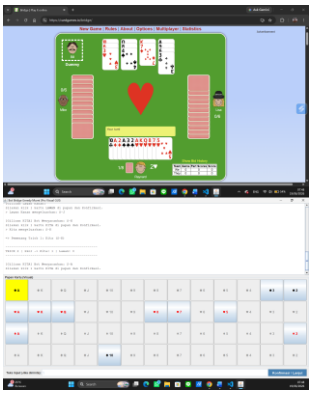

Figure 7. Suggestion to Start Trick with Ace Spade
[Source: Author]

4.1.4. Suggestion when Team is Winning

Figure 8. Suggestion to Play Low Cards When Winning [Source: Author]

4.1.5. Suggestion When Winning As Second Last Player

Figure 9. Suggestion to Play 5 Heart When Team is Winning [Source: Author]

4.2. Discussion

4.2.1. Bid Suggestion

Based on the combined evaluation of the High Card Points (HCP) and the longest suit metric, the bot recommends the user to open the auction with a 1 Heart (1♥) bid. Furthermore, the algorithm advises the user to prioritize the Heart suit in subsequent bidding escalations.

This bid recommendation is highly optimal. Because the user possesses the three highest-ranking cards in the Heart suit (Ace, King, and Queen) which are mathematically guaranteed to secure at least three consecutive tricks, as the opponents lack the higher-value cards necessary to contest them.

4.2.2. Trick Suggestion

When acting as the final player in a trick and currently trailing, the bot optimally suggests deploying the King of Spades to definitively secure the trick.

This heuristic decision is strategically sound. By acting as the last player, the bot operates with perfect information regarding the opponents' plays. Deploying a high-ranking card in this specific position guarantees an immediate trick win without the risk of being overcalled. Accumulating these guaranteed trick victories is a crucial greedy strategy for the Declarer to progressively fulfill the contract established during the bidding phase.

4.2.3. Suggestion to Initiate Trick

When initiating a trick, the algorithm optimally suggests leading with the Ace of Spades. This heuristic decision is highly effective, as the Ace holds the highest absolute rank in its suit, leading it guarantees capturing the current trick.

Furthermore, executing this move during the early phases of the game is a sound greedy strategy. At this phase, the probability of opponents being void in Spades is statistically low so they are forced to follow suit and are unable to contest the trick using a trump card.

4.2.4. Suggestion when Team is Winning

When acting as the final player in a trick where team has already secured the winning position, the algorithm optimally suggests playing the lowest-ranking card available in the required suit. This is a highly efficient resource management strategy inherent to the greedy approach.

By contributing a low-value card to an already won trick, the bot deliberately preserves its high-ranking cards. These conserved assets can then be strategically deployed in subsequent rounds to guarantee additional trick victories, ultimately maximizing the likelihood of fulfilling the contract.

4.2.5. Suggestion When Winning As Second Last Player

A notable vulnerability in the current algorithm emerges when the bot acts as the third player in a trick where team is currently winning, but the trick is not definitively secured (the winning card is not an Ace or an unassailable high card).

In this specific scenario, the greedy logic conservatively opts to play the lowest available card to save resources. However, this reveals a significant tactical flaw: it leaves the trick exposed to the final opposing player, who can easily capture the trick by playing a higher-ranking card. This limitation underscores the inherent weakness of a Pure Greedy Algorithm, as its focus on immediate resource conservation prevents it from anticipating and mitigating the opponent's subsequent moves.

V. CONCLUSION

This study proves that the implementation of a Pure Greedy Algorithm provides highly effective and computationally efficient strategies during specific, localized phases of Bridge game. The bot exhibits optimal decision-making during the bidding phase by accurately leveraging High Card Points (HCP) and suit length to establish a statistically sound contract. During the card play phase, the algorithm excels in scenarios where immediate tactical advantage aligns with perfect information. This includes successfully securing tricks from the final position, efficiently conserving resources when the partnership is already winning, and capitalizing on

absolute high card during the early rounds before opponents become void in specific suits.

However, the evaluation also highlights critical vulnerabilities inherent to the pure greedy approach, primarily stemming from its lack of look-ahead capabilities and historical tracking. A notable tactical flaw occurs when the bot acts as the third player. Its greedy instinct to conservatively save resources leaves the trick exposed and easily captured by the final opposing player.

Furthermore, extended analysis reveals a severe limitation as the game progresses into the middle phase. Because the current pure greedy architecture does not maintain a memory state of previously played cards, it operates with a static evaluation of its hand. Consequently, the bot may aggressively deploy a high-ranking card in a standard suit, failing to recognize that the opponents have already exhausted that specific suit. This lack of historical data management allows the opponents to easily ruff the bot's high card with a low-value trump card. Ultimately, this results in a devastating loss of critical offensive assets.

In conclusion, while a Pure Greedy Algorithm successfully automates fundamental Bridge mechanics and executes sound short-term tactical maneuvers, its inability to anticipate future moves or track historical game states limits its overall strategic viability. To achieve a more robust performance, future developments must integrate basic memory structures to track depleted suits, thereby preventing the algorithm from falling into predictable late-game traps.

VI. Appendix

Video Link: https://youtu.be/IhV_AK5Aa-w
Duration: 20 minutes 1 second

Programs used: https://github.com/RayapSunggal/Makalah-Strategi-Algoritma.git

VII. Acknowledgment

The author sincerely thanks God Almighty for providing the strength and opportunity to complete this article successfully. The author also extends his deep appreciation and gratitude to Ir. Rila Mandala, M.Eng., Ph.D. of K02 IF2211 Strategi Algoritma, whose semester-long support enabled the smooth completion of this article. The author would also like to extend his personal appreciation to creators of 出山，芒种，阳光彩虹小白马，浪子回头，夜空中最亮的星，刚好遇见你，and Harta, Tahta, Boru ini Raja songs, whose music has accompanied and uplifted the author during the preparation of this article

References

[1] Rauðás Games. 2024. *BRIDGE*. Retrieved June 18, 2026 https://cardgames.io/bridge/

[2] World Bridge Federation. 2017. *The Laws of Duplicate Bridge*.

[3] Rinaldi. 2026. *Algoritma Greedy (Bagian 1)*. Informatika. Retrieved June 18, 2026 https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/04-Algoritma-Greedy-(2026)-Bag1.pdf

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.

Medan, 19 Juni 2026

Raynard Fausta
13524052