Exploring the Application of Dynamic Programming in Blockchain Technology

Kenneth Ezekiel Suprantoni - 13521089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521089@mahasiswa.itb.ac.id

***Abstract*—Blockchain technology has emerged as a revolutionary solution for decentralized and secure data storage and transaction verification. In recent years, various algorithms from computer science have been investigated for their applicability in enhancing the performance and scalability of blockchain systems. This paper focuses on the application of Dynamic Programming, a widely used algorithmic technique, in blockchain technology. We delve into the fundamental concepts of Dynamic Programming and explore how it can address critical challenges in blockchain, such as consensus protocols, scalability, and optimization problems. Through comprehensive analysis and case studies, we demonstrate the potential benefits and limitations of utilizing Dynamic Programming in blockchain applications. The findings of this research shed light on the integration of algorithmic techniques within blockchain technology, paving the way for further advancements and innovation.**

***Keywords*—Dynamic Programming, Consensus Protocols, Scalability, Resource Allocation, Resource Optimization, Blockchain**

I. Introduction

Data, as defined by the Merriam-Webster dictionary, is a factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation, while data security, as defined by IBM, is the practice of protecting digital information from unauthorized access, corruption, or theft throughout its entire lifecycle. The main focus of this paper is the protection of data from corruption or manipulation, which introduces the concept of blockchain. Blockchain, most common in the field of economic-technology, is a way to store data that focuses primarily on the protection of the data from corruption or modification, its largest use-case is the cryptocurrency blockchain, where it is used to store the transactional data of the currency to keep track of the current flow of the currency, and the balance of users. In short, its implementation is done by storing a fixed amount of data in a block, hashing the data of the block, and storing the hash on the next block, which will include the previous block's hash as a data that will be hashed with its content.

Blockchain technology has emerged as a revolutionary solution for decentralized and secure data storage and transaction verification. In recent years, the applicability and scalability of blockchain has been massively studied. As blockchain systems continue to evolve, there is a growing need to enhance their performance, scalability, and resource utilization. One approach for addressing these challenges is the application of Dynamic programming.

Dynamic programming is a powerful algorithmic technique widely used in computer science and mathematics to solve complex problems by breaking them down into overlapping subproblems and solving each subproblem only once. It is based on the principle of optimal substructure, which means that and optimal solution to a larger problem can be constructed from optimal solutions to its smaller subproblems.

This paper aims to explore and showcase the potential benefits of integrating Dynamic programming techniques into blockchain technology. By delving into its applications in consensus protocols, scalability enhancement, and resource allocation, this paper provides insight into how Dynamic programming can revolutionize blockchain systems.

II. Theoretical Basis

1. Optimal Substructure

Optimal substructure is a property exhibited by problems where an optimal solution to a larger problem can be constructed from optimal solutions to its smaller subproblems. In other words, the optimal solution to a problem can be expressed in terms of the optimal solutions to its subproblems. This property enables the decomposition of a complex problem into smaller, more manageable solutions.

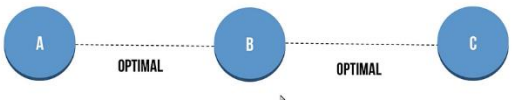

Fig. 1.1 Optimal Substructure
Source: https://tarunjain07.medium.com/dynamic-programming-notes-531de44a60f4

2. Overlapping Subproblems

Overlapping subproblems occur when the same subproblems are solved multiple times during the process of solving a larger problem. This repetition of subproblems leads to redundant computations and inefficient solutions. However, by using Dynamic Programming, solutions to subproblems can be stored and reused, eliminating the need to recompute them, and improving the overall efficiency of the algorithm.

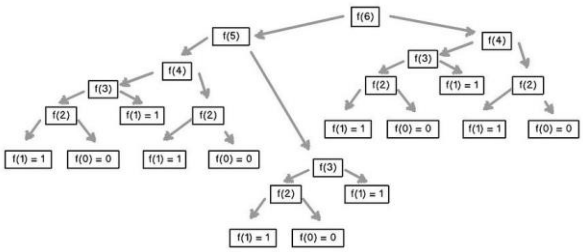

Fig. 2.1 Overlapping Subproblems
Source: https://tarunjain07.medium.com/dynamic-programming-notes-531de44a60f4

3. Dynamic Programming

1. Definition and Overview

Dynamic Programming is a problem-solving technique in computer science and mathematics that involves solving complex problems by breaking them down into smaller overlapping subproblems and solving each subproblems only once. The main idea behind it is to store solutions to subproblems in a table or memoization array, so that they can be looked up and reused when needed. This technique eliminates redundant calculations and improves the efficiency of the algorithm. By solving each subproblem only once and storing the solutions, Dynamic Programming avoids unnecessary recomputation and significantly speeds up the overall problem-solving process.

The key steps involved in applying Dynamic Programming to a problem are:

- Characterizing the structure of the problem and identifying optimal substructure.
- Defining the recursive relationship that expresses the solution to the larger problem in terms of solutions to its smaller subproblems.
- Creating a memoization table or array to store the solutions to subproblems.
- Filling the table iteratively or recursively by solving subproblems and storing their solutions.
- Utilizing the table to construct the final solution to the original problem.

Dynamic Programming is commonly used to solve optimization problems, such as finding the shortest path, maximizing, or minimizing a value, or allocating resources optimally. It has applications in various domains, including algorithm design, artificial intelligence, operations research, and computational biology, among others.

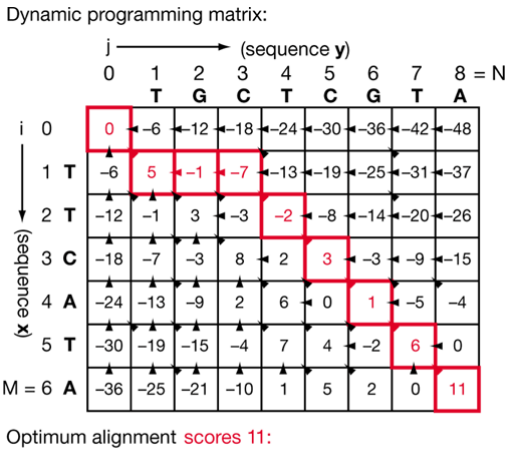

Fig. 3.1.1 Dynamic Programming Example
Source: https://www.nature.com/articles/nbt0704-909

2. Principle of Optimality

The Principle of Optimality, formulated by Richard Bellman, is the core principle underlying Dynamic Programming. It states that an optimal solution to a problem contains optimal solutions to its subproblems. By exploiting this principle, Dynamic Programming algorithms break down a problem into smaller subproblems and solve each subproblem only once, storing the solutions for future use.

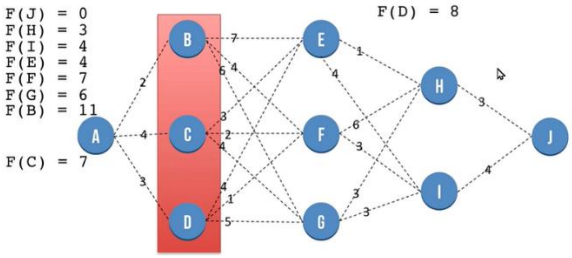

Fig. 3.2.1 Principle of Optimality
Source: https://tarunjain07.medium.com/dynamic-programming-notes-531de44a60f4

3. Recursive Structure

Dynamic Programming problems are often characterized by recursive structures. The solution to a problem can be defined in terms of solutions to smaller subproblems of the same nature. This recursive structure allows for the formulation of a recursive relationship or recurrence relation, which expresses the solution to a larger problem in terms of the solutions to its subproblems.

4. Memoization

Memoization, or tabulation, is a technique used in Dynamic Programming to store the solutions to subproblems in a table or memoization array. By storing the computed solutions, redundant calculations are avoided, and the efficiency of the algorithm is improved. Memoization ensures that each subproblem is solved only once, and subsequent occurrences of the same subproblem can be quickly retrieved from the

table.

5. **Bottom-up and Top-down approaches**

   Dynamic Programming algorithms can be implemented using two approaches: bottom-up and top-down. In the bottom-up approach, also known as tabulation, solutions to smaller subproblems are computed first and then used to compute solutions to larger subproblems until the final solution is obtained. In the top-down approach, also known as memoization, the problem is recursively divided into smaller subproblems, and the solutions to these subproblems are memoized and reused during the process.

6. **Complexity Analysis**

   Dynamic Programming algorithms often exhibit improved time and space complexity compared to naive approaches due to the elimination of redundant computations. By avoiding recomputation of overlapping subproblems, the overall time complexity can be significantly reduced. However, the space complexity may increase due to the storage of solutions in the memoization table.

4. Blockchain

1. Blockchain definition

Blockchain can be defined as a type of Distributed Ledger Technology that consists of continuously growing blocks of data that is linked to one another by a CHF. Each block of data contains the hash of the previous block, a timestamp, the data stored, and when the data stored reaches the maximum size, a hash of the block (including the hash function of the previous block) will be generated, effectively generating a link from a block into another block, which in turn results in the safety of the previous block, as changing some value of the previous block, will also change its hash, which creeps into the next block, and so on. For security, the blockchain is typically managed by a peer-to-peer computer network, where nodes follow a consensus algorithm protocol to add a new block. This distribution principle makes it hard for one side to manipulate the data on the previous blocks, not like centralized data storage, thus the only way to manipulate data onto the block is to add it into the new block.

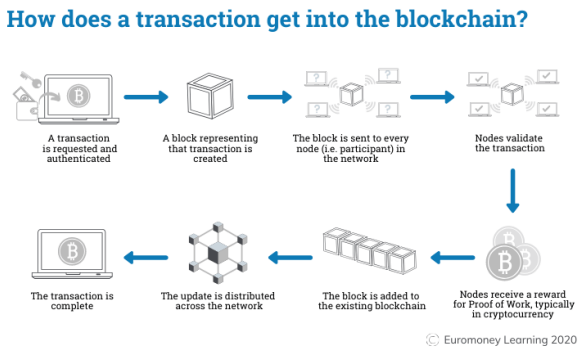

Fig. 4.1 How transactions/data gets into the blockchain
Source: https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain

2. Consensus Protocols

Consensus protocols are fundamental mechanisms in blockchain technology that enable participants in a decentralized network to agree on the state of the blockchain and validate transactions. They play a crucial role in ensuring the integrity, security, and trustworthiness of the distributed ledger. Consensus protocols enable consensus among network participants who may not trust each other or have conflicting interests, allowing them to reach agreement on the validity and order of transactions.

The choice of consensus protocol in a blockchain network depends on various factors such as the desired level of decentralization, security requirements, scalability, and energy efficiency. Different consensus protocols employ different algorithms and mechanisms to achieve consensus. Some commonly used consensus protocols:

- Proof of Work (PoW):

  Proof of Work is the most well-known and widely used consensus protocol, famously associated with Bitcoin. In PoW, participants, known as miners, compete to solve complex mathematical puzzles. The miner who solves the puzzle first gets the right to add a new block to the blockchain and receives a reward. PoW is resource-intensive, requiring significant computational power and energy consumption. It ensures network security by making it difficult to tamper with the blockchain but can suffer from scalability limitations.

- Proof of Stake (PoS):

  Proof of Stake is an alternative consensus protocol that aims to address the energy consumption and scalability issues of PoW. In PoS, instead of miners competing based on computational power, validators are chosen to create blocks based on the stake (ownership) they hold in the network. Validators are selected probabilistically, considering factors such as the number of coins held and their age. PoS requires validators to have a financial stake in the network, reducing the need for resource-intensive computations.

- Delegated Proof of Stake (DPoS):

  Delegated Proof of Stake is a variant of PoS where token holders delegate their voting rights to a limited number of trusted entities called "delegates" or "witnesses." These delegates are responsible for validating transactions and producing blocks on behalf of the network. DPoS aims to increase transaction throughput and scalability by allowing faster block creation and reducing the number of participants involved in the consensus process.

- Practical Byzantine Fault Tolerance (PBFT):

  PBFT is a consensus protocol designed for

permissioned blockchain networks, where participants are known and trusted. PBFT ensures Byzantine fault tolerance, meaning it can tolerate malicious actors or faulty nodes in the network. In PBFT, a leader is chosen to propose a block, and a certain number of replicas (nodes) perform a consensus algorithm to agree on the proposed block's validity. PBFT provides fast transaction finality and high throughput but requires a predefined set of trusted nodes.

- Proof of Authority (PoA):

  Proof of Authority is a consensus protocol commonly used in private or consortium blockchains. In PoA, a set of known and trusted authorities, often referred to as validators or nodes, take turns producing blocks and validating transactions. Validators are identified by their identities rather than by computational power or stake. PoA offers high transaction throughput and low energy consumption but sacrifices decentralization.

III. Challenges in Blockchain Technology

1. Overview

Blockchain technology faces several challenges that hinder its widespread adoption. These challenges include scalability, security and privacy concerns, interoperability, energy efficiency, governance and regulation, user experience, and the development of scalable and efficient smart contracts.

2. Consensus Protocols' Impact

Consensus protocols play a crucial role in blockchain systems, ensuring agreement and consistency among network participants. However, the choice of consensus protocol can significantly impact the performance and scalability of blockchain networks. The scalability challenge arises due to the computational complexity involved in consensus protocols. Exploring the application of algorithms like Dynamic Programming can optimize the consensus process, leading to more efficient and faster consensus algorithms.

3. Optimization problems and Resource Allocation

Resource allocation and optimization are essential factors for maximizing the performance of blockchain systems. Dynamic Programming can be leveraged to optimize resource allocation and utilization within the blockchain network. By analyzing data characteristics and transaction requirements, Dynamic Programming algorithms can identify efficient storage allocation strategies, minimize storage overhead, and maximize capacity utilization. Additionally, Dynamic Programming techniques can optimize transaction pools and block creation and propagation, leading to reduced network congestion, latency, and improved scalability.

IV. Dynamic Programming Techniques in Blockchain

1. Application of Dynamic Programming in Consensus Protocols

Consensus protocols play a vital role in blockchain networks, ensuring agreement among participants on the validity and ordering of transactions. Dynamic Programming can be employed to optimize consensus protocols by improving their efficiency and performance. By analyzing the structure and characteristics of consensus algorithms, Dynamic Programming techniques can be used to optimize the selection of validators, block creation, and validation processes, leading to faster consensus and reduced energy consumption.

An example of this is the optimization of block validation in the Proof of Work (PoW) consensus, where miners compete to solve a complex mathematical puzzle to validate transactions and create new blocks. As the computational power required to solve the puzzle can be resource-intensive and time-consuming, dynamic programming can be utilized to optimize the process. By employing dynamic programming techniques, miners can analyze and break down the puzzle-solving process into smaller subproblems. The solution to these problems can be stored in a memoization table to avoid redundant computation. As miners encounter similar puzzle components, they can reference the precomputed solutions, reducing the computational burden. The technique optimizes the utilization of computational resources by efficiently reusing previously computed solutions, thereby enhancing the overall efficiency and performance of the consensus protocol.

By applying Dynamic Programming in PoW-based consensus, miners can achieve faster block validation, reduced energy consumption, and improved scalability. The utilization of this technique optimizes the consensus process, making it more efficient and capable of handling increased transaction volumes, ultimately enhancing the overall performance of the blockchain network.

Another example is the application in Proof of Stake (PoS) consensus. In PoS, block validators are selected based on the stake (ownership) they hold in the network. The probability of being chosen as a validator is proportional to the amount of stake an individual holds. However, determining the optimal set of validators for each block can be a complex task, especially in large-scale PoS networks with numerous participants. Dynamic Programming can be employed to optimize the validator selection process by considering various factors, such as the stake distribution, reputation, and previous performance of validators.

The state represented in PoS consensus is a set of validators and their corresponding stakes. Each validator has associated attributes like reputation and historical performance. The subproblems to be divided is to determine the optimal set of validators for a

specific block. The solution to this subproblem will depend on the previously selected validators and their performance. The recursive relationship among the states and subproblems involves computing the optimal set of validators for the current block based on the optimal set of validators for the previous block. It can consider factors such as stake distribution, validator reputation, and past performance. By dynamically programming the selection algorithm, the process will be optimized, and the overall network security and efficiency will increase. To avoid redundant computations, the algorithm will also utilize memoization to store the previously computed selection, so it can be efficiently accessed and reused during block validation.

2. Improving Scalability through Dynamic Programming-based Approaches

Scalability is a significant challenge in blockchain technology as the number of participants and transactions increases. Dynamic Programming techniques can be applied to enhance scalability by optimizing various aspects of the blockchain network. For example, Dynamic Programming algorithms can optimize the selection of transactions for inclusion in blocks, prioritize transactions based on factors like fees or urgency, and optimize block propagation strategies. By efficiently utilizing network resources and prioritizing important transactions, Dynamic Programming-based approaches can improve the overall scalability of blockchain networks. Parallelization of computations can also be achieved with dynamic programming by breaking down complex computations into smaller subproblems, which optimizes the utilization of computational resources, accelerates transaction processing, and overall scalability of the blockchain network.

3. Resource Allocation and Optimization using Dynamic Programming

Resource allocation is a critical aspect of blockchain networks, involving the efficient allocation of storage, computing power, and network bandwidth. Dynamic Programming techniques can be utilized to optimize resource allocation and utilization. For instance, Dynamic Programming algorithms can analyze transaction patterns, data characteristics, and storage availability to optimize storage allocation strategies. They can minimize storage overhead, ensure efficient utilization of available storage space, and optimize data compression techniques. Additionally, Dynamic Programming can optimize resource allocation for computing power and network bandwidth, ensuring efficient utilization and minimizing bottlenecks and compressing data and improving storage strategies within the blockchain network.

V. CASE STUDIES

1. Consensus Algorithm

In this case study, we explore the application of Dynamic Programming in developing a consensus algorithm for a blockchain network. The goal is to improve the efficiency and performance of the consensus process.

Background:

The blockchain network in question faces scalability challenges due to the complexity of its consensus algorithm. The existing algorithm requires significant computational resources and can be time-consuming. The network seeks to optimize the consensus process using Dynamic Programming techniques.

Implementation:

- Subproblem Identification: The consensus algorithm is decomposed into smaller subproblems, such as block validation, leader selection, and agreement on block ordering.
- Recursive Relationship: Dynamic Programming is used to establish the recursive relationship between subproblems. For example, the result of validating a block depends on the previous block's validation status and the agreement reached by other nodes in the network.
- Memoization: Solutions to subproblems are memoized, allowing for efficient reuse and avoiding redundant computations. The memoization table stores intermediate results, such as the validation status of previous blocks and the agreed-upon ordering.
- Optimal Solution Determination: The Dynamic Programming algorithm utilizes the memoized solutions to determine the optimal path for consensus. This can involve considering factors like energy consumption, computational efficiency, and fault tolerance.

Results:

By leveraging Dynamic Programming techniques, the consensus algorithm achieves improved efficiency and performance. The optimized algorithm reduces the computational effort required for block validation, accelerates the consensus process, and enhances the overall scalability of the blockchain network. The use of Dynamic Programming allows for faster agreement on block validity and ordering, making the network more robust and scalable.

2. Scalable Blockchain Data Structures

In this case study, we examine how Dynamic Programming can be employed to optimize data structures in a blockchain network, leading to improved scalability.

Background:

The blockchain network is experiencing scalability issues due to inefficient data structures that do not effectively handle increasing transaction volumes. The network aims to enhance scalability by utilizing Dynamic Programming to optimize its data structures.

Implementation:

- Data Structure Analysis: The existing data structures, such as the transaction pool, block storage, and state storage, are analyzed to identify areas for improvement.
- Subproblem Decomposition: Dynamic Programming is applied to decompose the data

structure optimization problem into smaller subproblems. For example, optimizing block storage may involve subproblems like transaction compression, efficient indexing, and retrieval.

- Recursive Relationship: The subproblems are interconnected through a recursive relationship. The solution to each subproblem depends on the solutions of its subproblems, and the relationship is established to ensure optimal results.
- Memoization and Reuse: Dynamic Programming employs memoization to store and reuse solutions to subproblems. For instance, previously compressed transactions or indexed blocks can be efficiently accessed and reused to avoid redundant computations.

Results:

By utilizing Dynamic Programming-based approaches, the blockchain network achieves improved scalability through optimized data structures. The optimized transaction pool, block storage, and state storage facilitate faster transaction processing, reduced storage overhead, and improved utilization of available resources. The application of Dynamic Programming enables the blockchain network to efficiently handle increased transaction volumes and promotes scalability in its data management.

3. Blockchain Transactions Optimization

In this case study, we explore the use of Dynamic Programming techniques for optimizing transactions in a blockchain network, improving efficiency and scalability.

Background:

The blockchain network experiences congestion and delays due to suboptimal transaction processing and prioritization. The network aims to address these issues by leveraging Dynamic Programming to optimize transaction handling.

Implementation:

- Transaction Selection and Prioritization: Dynamic Programming is utilized to analyze transaction characteristics, such as fees, urgency, and size. The algorithm prioritizes transactions based on these factors to ensure optimal selection for inclusion in blocks.
- Transaction Validation Optimization: Dynamic Programming techniques are applied to optimize the validation process. The algorithm breaks down the validation process into smaller subproblems and memoizes the results to avoid redundant computations.
- Parallelization of Transaction Processing: Dynamic Programming allows for the parallelization of transaction processing. By decomposing the process into subproblems, the algorithm can process transactions concurrently, optimizing resource utilization and reducing processing time.

Results:

By applying Dynamic Programming-based approaches to transaction optimization, the blockchain network achieves improved efficiency and scalability. The optimized transaction selection and prioritization algorithms reduce congestion, improve transaction processing speed, and enhance the overall throughput of the network. The parallelization of transaction processing further boosts scalability by efficiently utilizing computational resources. The application of Dynamic Programming in transaction optimization enables the blockchain network to handle increased transaction volumes and promotes scalability in its transaction processing capabilities.

VI. Benefits and Limitations

1. Benefits

- Optimal Solutions: Dynamic Programming enables the identification and computation of optimal solutions to complex problems in blockchain. It breaks down problems into smaller subproblems, allowing for efficient and accurate optimization.
- Scalability Improvement: Dynamic Programming-based approaches can improve the scalability of blockchain networks by optimizing various processes, such as consensus, transaction handling, and resource allocation. This optimization enhances the network's capacity to handle increased transaction volumes and improves overall performance.
- Resource Efficiency: By optimizing computations, resource allocation, and storage utilization, Dynamic Programming minimizes resource wastage and improves the efficiency of blockchain networks. It enables the efficient use of computational power, storage capacity, and network bandwidth.
- Flexibility and Adaptability: Dynamic Programming provides flexibility in adapting to changing network conditions, transaction patterns, and scalability requirements. It allows blockchain systems to adjust and optimize their operations based on real-time data and evolving network dynamics.

2. Limitations and challenges

- High Computational Complexity: Dynamic Programming algorithms can be computationally intensive, especially for large-scale blockchain networks. As the size and complexity of the problem increase, the computational resources required for the optimization process may become a limitation.
- Time Complexity: Dynamic Programming solutions often involve analyzing and computing multiple subproblems, which can result in significant time complexity. For real-time blockchain applications that require immediate transaction processing, the time

taken for Dynamic Programming optimization may impact overall system responsiveness.

- Storage Requirements: Dynamic Programming algorithms typically require storing and accessing intermediate results (memoization). This can lead to increased storage requirements, particularly when dealing with large-scale blockchain networks with extensive transaction histories.
- Algorithm Design Complexity: Designing and implementing Dynamic Programming algorithms in the blockchain context can be complex. It requires a deep understanding of both the underlying blockchain system, and the specific problem being addressed. Developing efficient algorithms that consider the unique characteristics of blockchain technology can be challenging.

V. Review

Blockchain technology has some challenges to be addressed, such as the energy efficiency, computational cost, and time-consuming algorithms. Dynamic programming can be used as a mean to reduce the time and cost needed to do the computational work in blockchain technology, such as in consensus protocols, improvement of scalability of blockchain systems, and the optimization of transactions via validation of blocks.

VI. Acknowledgment

The Author would like to thank, first, the lecturer of Class 1 of Algorithm and Strategies, Mr. Rinaldi Munir of Bandung Institute of Technology, as the materials given are presented in a way that can be fully understood deeply by The Author, and also assigning the paper project as it is a way for The Author to explore the materials given in lectures and applications.

References

[1] https://www.ibm.com/id-en/topics/data-security , accessed 20 May 2023, 12.05 P.M.
[2] https://tarunjain07.medium.com/dynamic-programming-notes-531de44a60f4 , accessed 20 May 2023, 12.10 P.M.
[3] https://www.adsbynimbus.com/tech-blog/dynamic-programming , accessed 20 May 2023, 12.30 P.M.
[4] https://blog.boot.dev/cryptography/how-sha-2-works-step-by-step-sha-256/ , accessed 20 May 2023, 13.30 P.M.
[5] https://www.investopedia.com/terms/p/proof-work.asp , accessed 20 May 2023, 14.58 P.M.
[6] https://www.investopedia.com/terms/p/proof-stake-pos.asp , accessed 20 May 2023, 15.05 P.M.
[7] https://cryptocurrencyworks.com/.res/doc/Bitcoin-SNakamoto-Oct-2008.pdf , accessed 21 May2022, 10.50 A.M.
[8] https://github.com/B-Con/crypto-algorithms , accessed 21 May 2023 11.30 A.M.

Pernyataan

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

Bandung, 21 Mei 2023

Kenneth Ezekiel Suprantoni, 13521089

Link Video:
https://www.youtube.com/watch?v=PPFKeUKlJW8