DPOS 算法实验指导书

实验课时：3 学时
实验认识 3-4 人

一、实验名称

DPOS 共识算法的简单实现

二、实验内容

理解 DPOS 算法的工作原理和流程，了解委员会成员的选举和投票过程。

三、实验环境

Java 开发环境（JDK）：JDK17

IDE 工具：vscode

操作系统：windows11

四、算法描述

DPOS（Delegated Proof of Stake）是一种共识机制，也是区块链中一种流行的共识算法。与 PoW（Proof of Work）和 PoS（Proof of Stake）不同，DPOS 采用了委托投票的方式来选举出共识节点，由选举出的共识节点来验证和打包交易，生成新的区块。

DPOS 算法的主要思想是，让网络上的持币者通过投票选举出少数几个代表节点，这些节点被授权来验证和打包交易。持币者投票的权重与其持有的货币数量成正比。由于共识节点数量很少，交易验证速度快，处理能力强，同时也保证了网络的分布式特性和安全性。

五、实验过程

DPOS 算法的流程如下：

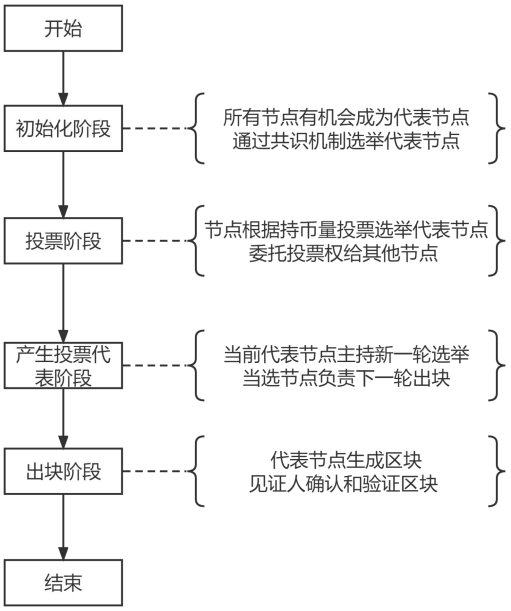


图 1：DPos 算法流程图

1. 初始化阶段：在区块链网络中，所有的节点都有机会成为“代表”（delegate），即参与出块的节点。在初始阶段，所有节点通过共识机制选举出一组“代表”节点。

区块链类：

```
public class Blockchain {
    // 区块链列表
    private static List<Blockchain> blockchainList = new ArrayList<>();
    // 区块链的难度
    private int difficulty;
    // 投票列表
    private List<Vote> voteList = new ArrayList<>();
    // 节点列表
    private List<Node> nodeList = new ArrayList<>();
    // 区块列表
    private List<Block> blockList = new ArrayList<>();
    //交易列表
    private List<Transaction> transaction = new ArrayList<>();
```

投票类：

```
public static class Vote {
    private String candidate; // 候选人
    private int voteCount; // 获得票数
```

```
    private int availableVotes; // 节点可用于投票的票数
```

节点类：

```
public static class Node {
    private String address;
    private int availableVotes; // 节点可用于投票的票数
    private int voteCount; // 获得票数
    private int tokenAmount; // 代币数量
```

区块类：

```
public static class Block {
    private int index;
    private long timestamp;
    private List<Transaction> transactionList;
    private List<Vote> voteList;
    private String previousHash;
    private String hash;
    private int nonce;
```

交易类：

```
public static class Transaction {
    private String from;
    private String to;
    private int amount;
    private String signature;
```

创建创世区块：

```
public void createGenesisBlock() {
    List<Transaction> transactions = new ArrayList<>();
    List<Vote> votes = new ArrayList<>();
    String previousHash = "";
    int nonce = 0;
    Block genesisBlock = new Block(0, System.currentTimeMillis(),
transactions, votes, previousHash, nonce);
    blockList.add(genesisBlock);
    System.out.println("genesisBlock: " + genesisBlock.hash);
}
```

2. 投票阶段：在下一轮次开始之前，所有节点都可以根据自己的持币量投票选举下一轮的“代表”节点。在此过程中，每个节点可以将自己的投票权委托给其他节点，以提高他们当选的概率。

```
// 添加节点并随机分配代币数量
List<Blockchain.Node> nodes = new ArrayList<>();
Random random = new Random();
```

```
int totalTokens = 10000; // 总代币数量
for (int i = 0; i < 100; i++) {
    int tokenAmount = 1 + random.nextInt(totalTokens / 10); // 保证每
个节点至少拥有 1 个代币
    totalTokens -= tokenAmount;
    String nodeAddress =
Blockchain.HashUtil.sha256(UUID.randomUUID().toString());
    Blockchain.Node node = new Blockchain.Node(nodeAddress,
tokenAmount);
    blockchain.addNode(node);
    nodes.add(node);
    // 在添加节点的同时，创建对应的投票并添加到投票列表
    Blockchain.Vote vote = new Blockchain.Vote(nodeAddress, tokenAmount,
tokenAmount);
    blockchain.addVote(vote);
    System.out.println("节点已添加，节点为："+ (i + 1) + ". " +
node.getAddress() + "，代币数量为：" + node.getTokenAmount());
}
// 根据分配的代币给予节点票数
for (Blockchain.Node node : nodes) {
    int numVotes = node.getTokenAmount(); // 获取节点的代币数量
    node.addVote(numVotes); // 给节点增加票数
}
// 进行随机投票模拟
Random random1 = new Random(System.currentTimeMillis());
List<Blockchain.Vote> votes = blockchain.getVoteList();
for (Blockchain.Node node : nodes) {
    int numVotes = node.getVoteCount(); // 获取节点的票数
    for (int i = 0; i < numVotes; i++) {
        int candidateIndex = random1.nextInt(votes.size()); // 随机选
择候选人索引
        Blockchain.Vote vote = votes.get(candidateIndex); // 获取对应
的候选人投票
        node.vote(vote); // 节点进行投票
    }
}
```

3. 产生投票代表阶段：在每个轮次结束时，由当前的“代表”节点主持新一轮的“代表”节点选举，当选的节点将负责下一轮的出块工作。

```
// 按票数排序节点
List<Blockchain.Node> sortedNodes =
Blockchain.sortNodesByVoteCount(nodes);

// 输出票数最高的 30 个节点
```

```
System.out.println("票数最高的 30 个节点：");
for (int i = 0; i < 30 && i < sortedNodes.size(); i++) {
    Blockchain.Node node = sortedNodes.get(i);
    System.out.println((i + 1) + ". " + node.getAddress() + " - 票数:
" + node.getVoteCount());
}
```

4. 出块阶段：在当前的轮次中，由被选中的“代表”节点负责生成区块，而其他节点将成为“见证人”（witness），对该区块的有效性进行确认和验证。

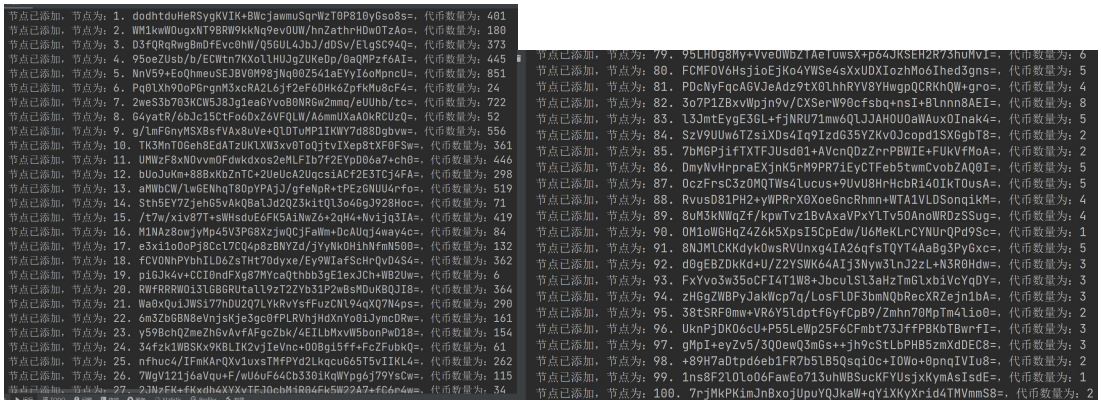
```
// 创建一个新的区块并添加到区块链
Blockchain.Block newBlock1 = new Blockchain.Block(1,
System.currentTimeMillis(), new ArrayList<>(),
blockchain.getVoteList(), blockchain.getLatestBlock().getHash(), 0);
System.out.println("等待添加区块 1：");
blockchain.addBlock(newBlock1);
System.out.println("区块 1 已添加，区块哈希为：" + newBlock1.getHash());
Blockchain.Block newBlock2 = new Blockchain.Block(2,
System.currentTimeMillis(), new ArrayList<>(),
blockchain.getVoteList(), blockchain.getLatestBlock().getHash(), 0);
System.out.println("等待添加区块 2：");
blockchain.addBlock(newBlock2);
System.out.println("区块 2 已添加，区块哈希为：" + newBlock2.getHash());
// 验证区块链的合法性
System.out.println("区块链的合法性为：" + blockchain.validate());
```

六、实验结果

生成创世区块

```
genesisBlock: zRspgzlO5gIqlEVj0uAvl6Oj33agV1ZWPfozjJojxFs=
创世区块已创建
```

生成 100 个节点，并将 10000 个代币随机分配给这 100 个节点



进行随机投票并将节点按获得的票数排序，得出排名前 30 的节点

```
票数最高的30个节点：
1. NnV59+EoQhmeuSEJBV0M98jNq00Z541aEYyI6oMpncU= - 票数：851
2. 2weS3b703KCW5J8Jg1eaGYvoB0NRGw2mmq/eUUhb/tc= - 票数：722
3. g/lmFGnyMSXBsfVAx8uVe+QlDTuMP1IKWY7d88Dgbvw= - 票数：556
4. aMWbCW/lwGENhqT8OpYPAjJ/gfeNpR+tPEzGNUU4rfo= - 票数：519
5. UMWzF8xNOvvmOFdwkdxos2eMLFIb7f2EYpD06a7+ch0= - 票数：446
6. 95oeZUsb/b/ECWtn7KXollHUJgZUKeDp/0aQMPzf6AI= - 票数：445
7. /t7w/xiv87T+sWHsduE6FK5AiNwZ6+2qH4+Nvijq3IA= - 票数：419
8. dodhtduHeRSygKVIK+BWcjawmuSqrWzT0P810yGso8s= - 票数：401
9. D3fQRqRwgBmDfEvc0hW/Q5GUL4JbJ/dDSv/ElgSC94Q= - 票数：373
10. RWfRRRWOi3lGBGRUtall9zT2ZYb31P2wBsMDuKBQJI8= - 票数：364
11. fCVONhPYbhILD6ZsTHt7Odyxe/Ey9WIafScHrQvD4S4= - 票数：362
12. TK3MnTOGeh8EdATzUKlXW3xv0ToQjtvIXep8tXF0FSw= - 票数：361
13. bUoJuKm+88BxKbZnTC+2UeUcA2UqcsiACf2E3TCj4FA= - 票数：298
14. Wa0xQuiJWSi77hDU2Q7LYkRvYsfFuzCNl94qXQ7N4ps= - 票数：290
15. nfhuc4/IFmKArQXv1uxsTMfPYd2LkqcuG65T5vIIKL4= - 票数：262
16. C3R09Tj46wa2hXsUEg5schGD/uQXaWWcGRvT8kNRhzM= - 票数：181
17. WM1kwWOugxNT9BRW9kkNq9evOUW/hnZathrHDwOTzAo= - 票数：180
18. 6m3ZbGBN8eVnjsKje3gc0fPLRVhjHdXnYo0iJymcDRw= - 票数：161
19. nd8rqnXkr/UFtFwUvdUiylT7mvGmUkQ85Gf7H3EcAzg= - 票数：159
20. jVo74pymUA/VP2JRZM5fngmNzHhX7UVxCmvCuLuwD64= - 票数：159
21. y59BchQZmeZhGvAvfAFgcZbk/4EILbMxvW5bonPwD18= - 票数：154
22. e3xi1oOoPj8Ccl7CQ4p8zBNYZd/jYyNkOHihNfmN500= - 票数：132
23. 7WgV121j6aVqu+F/wU6uF64Cb330iKqWYpg6j79YsCw= - 票数：115
24. DcYnWK0b70ly2F/wZPTAlC+cmK9qvulHA+D+j6RfyUw= - 票数：104
```

最后进行出块并验证合法性

```
等待添加区块1：
Block mined: 000W5RO5W+ZY1ST3sMwMz1XKL1+w7p6bE64OqwkrzlY=
区块 1已添加，区块哈希为：000W5RO5W+ZY1ST3sMwMz1XKL1+w7p6bE64OqwkrzlY=
等待添加区块2：
Block mined: 000hFDksh28Ds9BiwnwDCQ9jYZAigOZuwHL99iXzkmU=
区块 2已添加，区块哈希为：000hFDksh28Ds9BiwnwDCQ9jYZAigOZuwHL99iXzkmU=
区块链的合法性为：true
```