更多 GAS 优化技巧

登链社区 - Tiny熊

如何写优秀的合约

- 优秀：更少 Gas 、更安全、用户体验好（如更少的交互次数、更容易获取链上数据的变化）
- 更少 Gas 消耗:
  - 理解 EVM 的运行、存储布局
  - 合理的数据设计、链上链下数据的平衡
    - 应用优秀的设计模式（Permit、默克尔树、Multical、 最小代理…）

保持学习、积累经验

Gas 优化技巧

- 修改变量顺序 -> 合并槽（但尽量使用 uint256 的变量）、 需要时使用 unchecked
- 控制交易内状态（如重入锁），用瞬态存储 transient，常量或 immutable 代替变量
- 精确声明 Solidity 合约函数的可见性
- 避免无限制的循环、线性增长
- 合约中没有引用的变量，用事件 （https://learnblockchain.cn/article/880）
- 减少链上数据：IPFS 链下存储
- 使用代理进行大规模部署（复用实现合约）
- 区分交易 Gas （变量） 和 部署 Gas (字节码) //
- **链下计算、链上验证（如: 数组用链表实现、可迭代的链表）**
- **在合约验证数据，而不是存储数据（如使用 Merkle 树）**

https://learnblockchain.cn/column/1

https://decert.me/tutorial/rareskills-gas-optimization/

考虑需求

创建一个“学校”智能合约来收集学生地址。合约必须具有3个主要功能：

1. 在合约中添加或删除学生。
2. 询问给定的学生地址是否属于学校。
3. **获取所有学生的名单**。

Demo

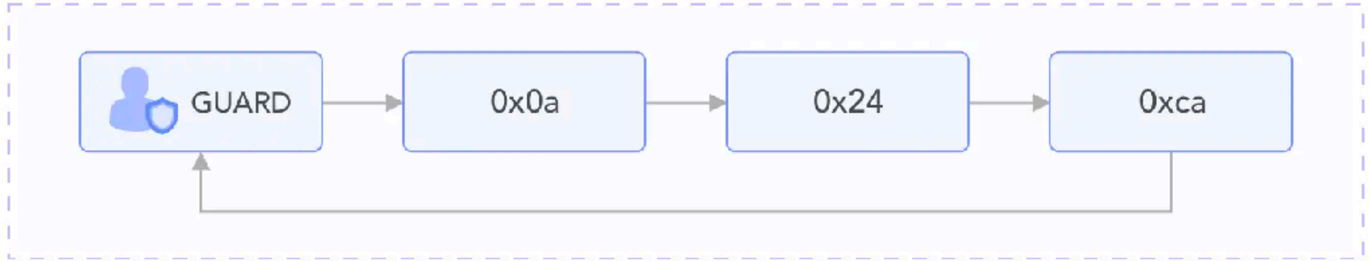
Mapping实现: SchoolMapping.sol

数组实现： SchoolBaseArray.sol

Gas 对比

Total number student in list	10 students		100 students	
	mapping	array	mapping	array
add	43,704	57,064	43,704	138,364
remove	13,721	59,612	13,721	416,662

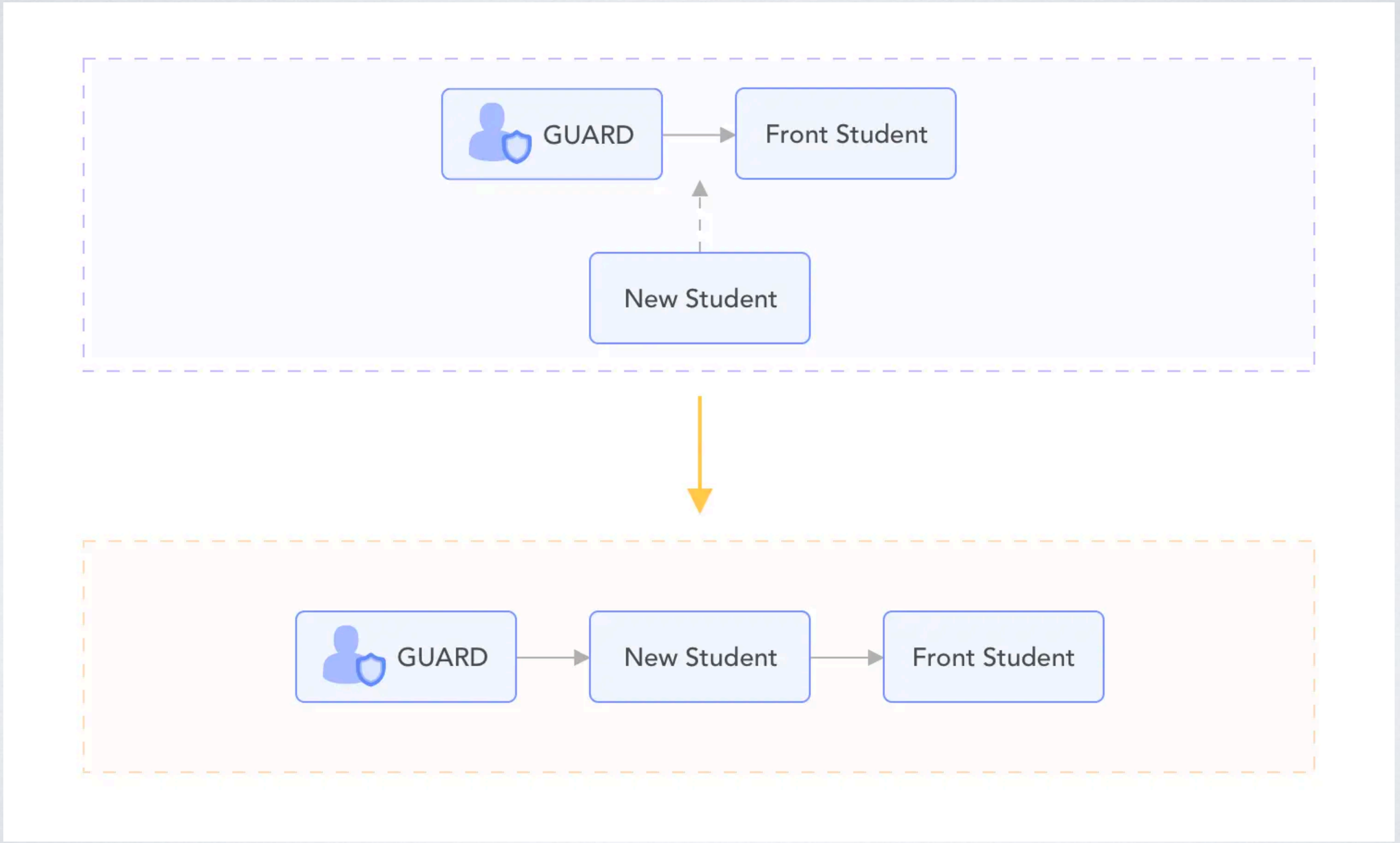
可迭代链表



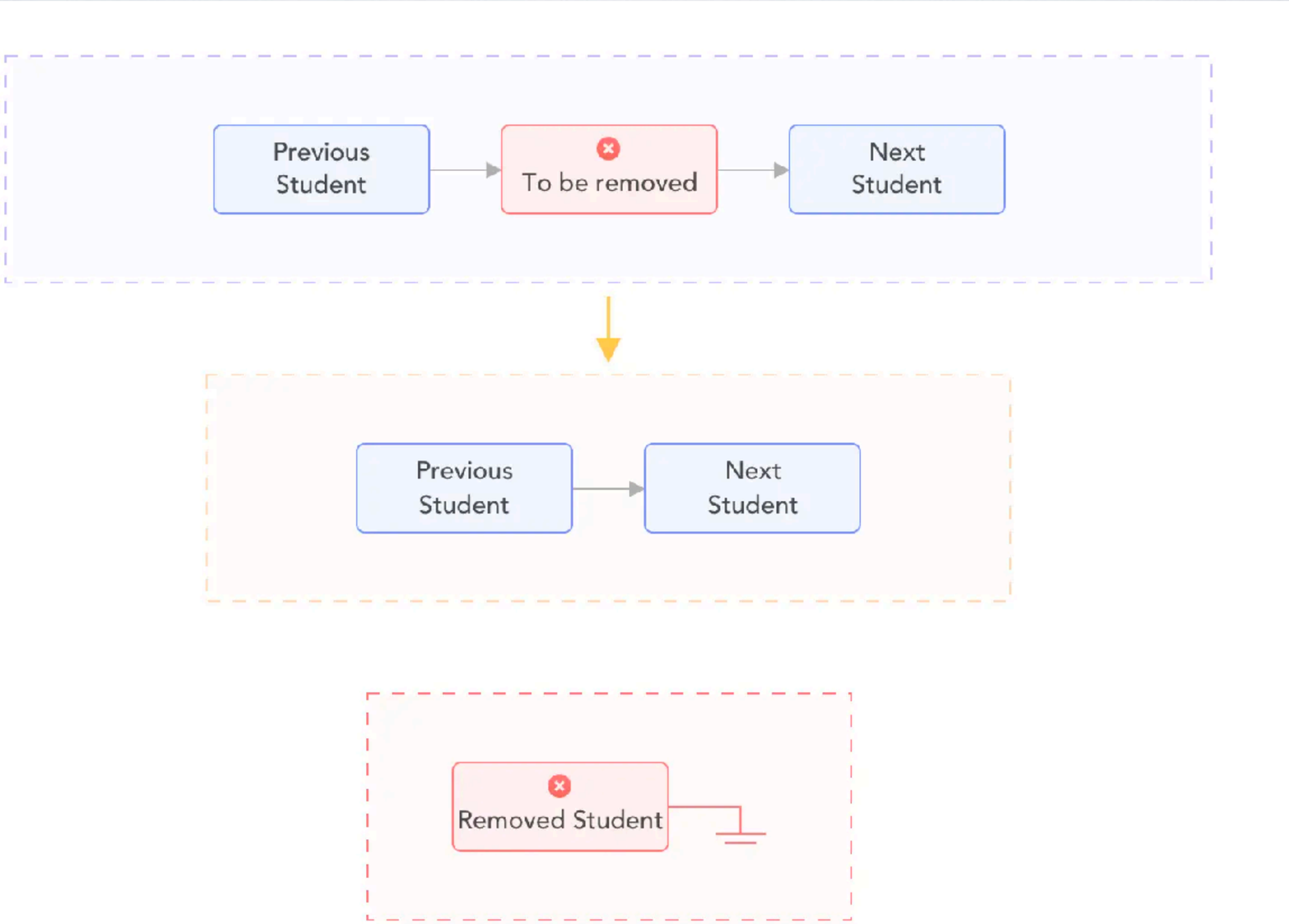
Key	Value
GUARD	0x0a
0x0a	0x24
0x24	0xca
0xca	GUARD

Mapping

可迭代链表 – 添加


GUARD
Front Student
New Student
GUARD
New Student
Front Student

可迭代链表 – 删除


Previous Student
To be removed
Next Student
Previous Student
Next Student
Removed Student

可迭代链表 – 性能

Total number student in list	10 students	100 students
add	54,667	54,667
remove (unoptimized)	31,680	83,040
remove (optimized)	23,974	23,974

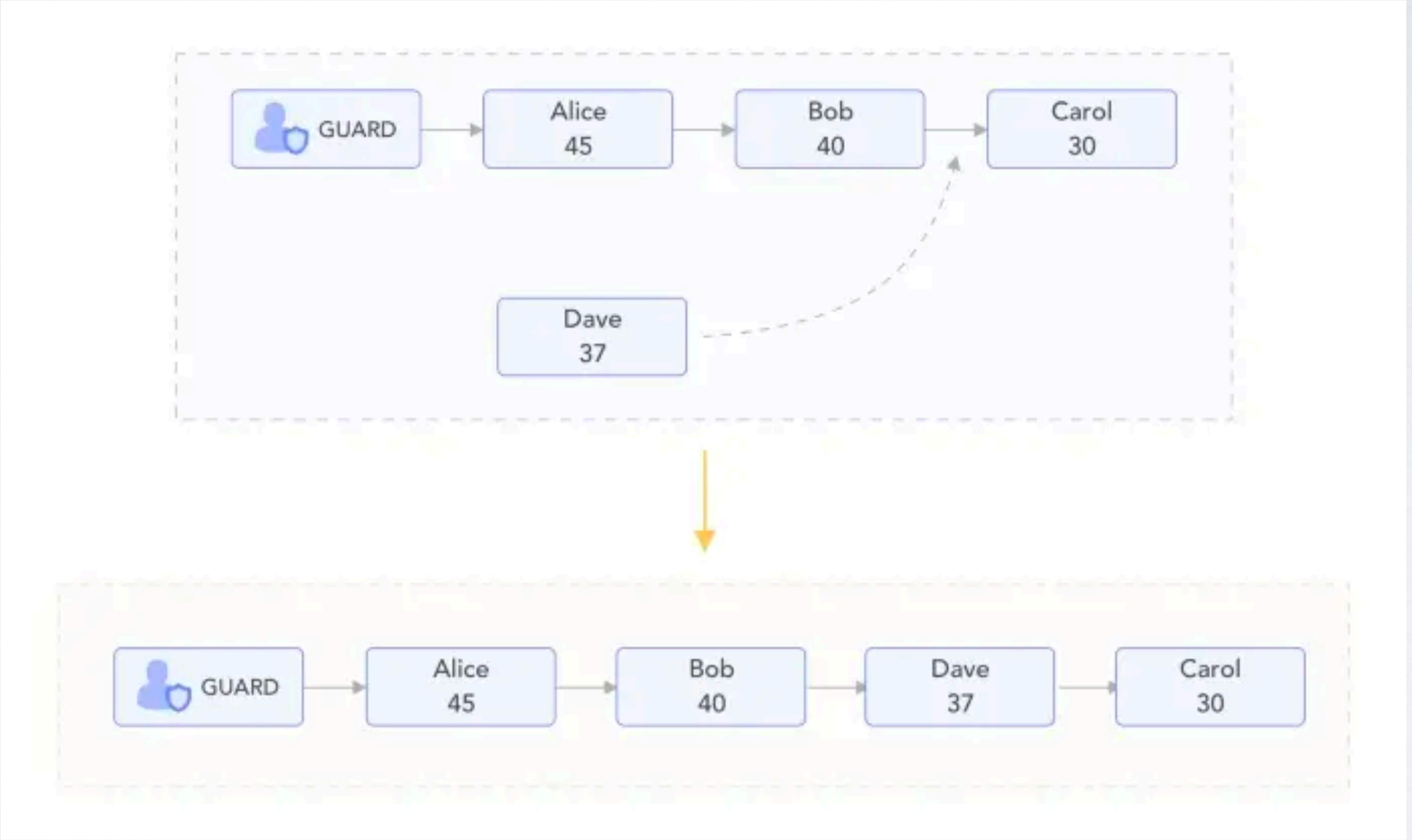
SchoolMappingList.sol

考虑新需求

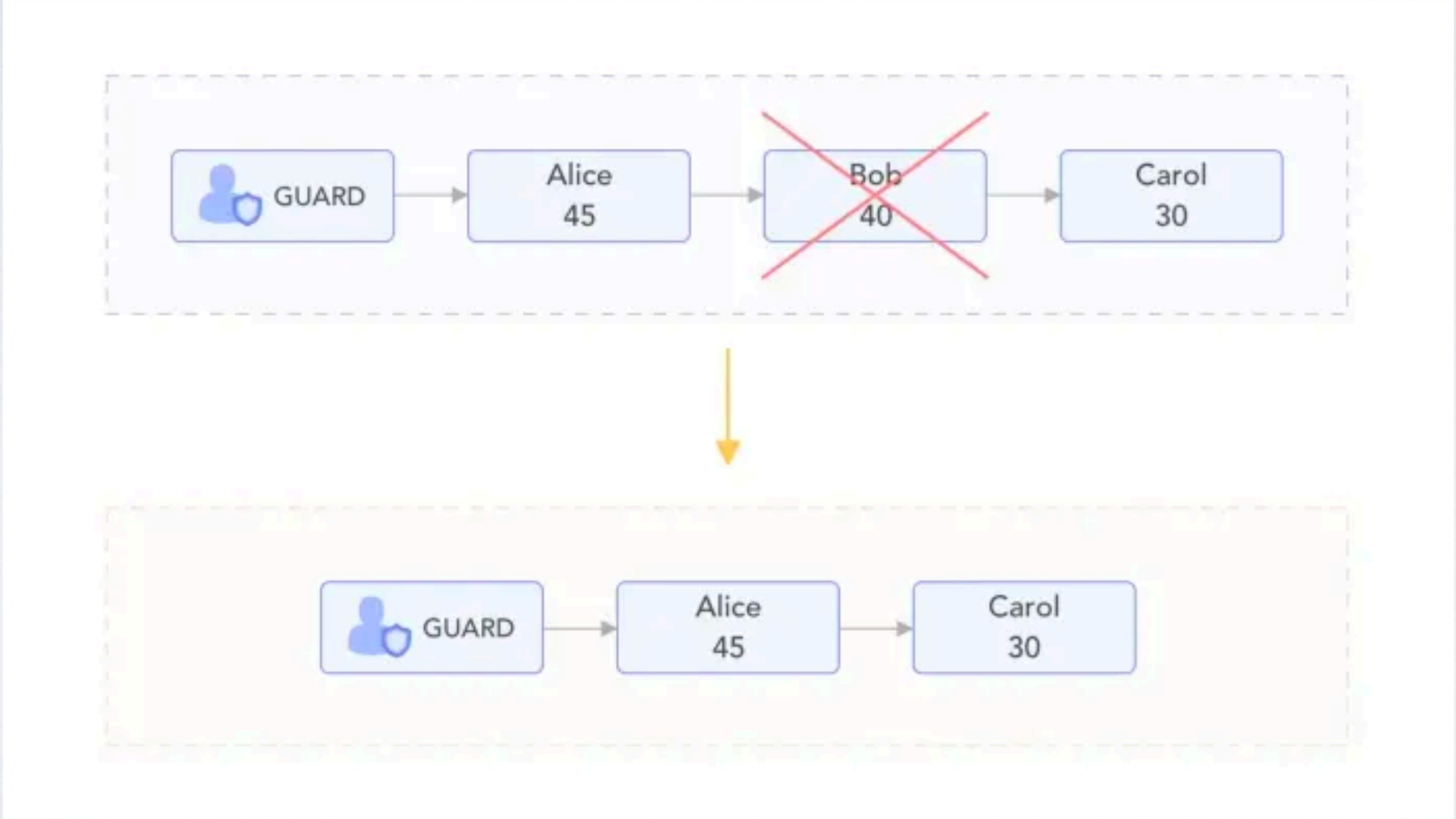
需要根据分数来维持学生的**排序**，功能需求如下：

1. 将新学生添加到根据分数排序的列表中
2. 提高学生分数（保持列表有序）
3. 降低学生分数（保持列表有序）
4. 从名单中删除学生
5. **获取前K名学生名单**

可迭代链表 – 有序插入


GUARD
Alice
45
Bob
40
Carol
30
Dave
37
GUARD
Alice
45
Bob
40
Dave
37
Carol
30

可迭代链表 – 列表删除



可迭代链表 – 更新分数


We want to add Bob 9 points
GUARD
Alice
45
Bob
40
Carol
30
Remove Bob from list first
(temporary)
GUARD
Alice
45
Carol
30
Bob
40 + 9
Add Bob back to the list where he should be
GUARD
Bob
49
Alice
45
Carol
30

优化的可迭代链表

https://github.com/OpenSpace100/blockchain-tasks/blob/main/solidity_sample_code/SchoolOptimized.sol

练习题

- 之前的 TokenBank 合约的前 3 名，修改为前 10 名，并用可迭代的链表保存。

https://decert.me/challenge/753d5050-e5e4-4a0d-8ad9-9ecd7e0e0788

考虑需求

- 给所有的集训营学员（白名单）发纪念勋章?

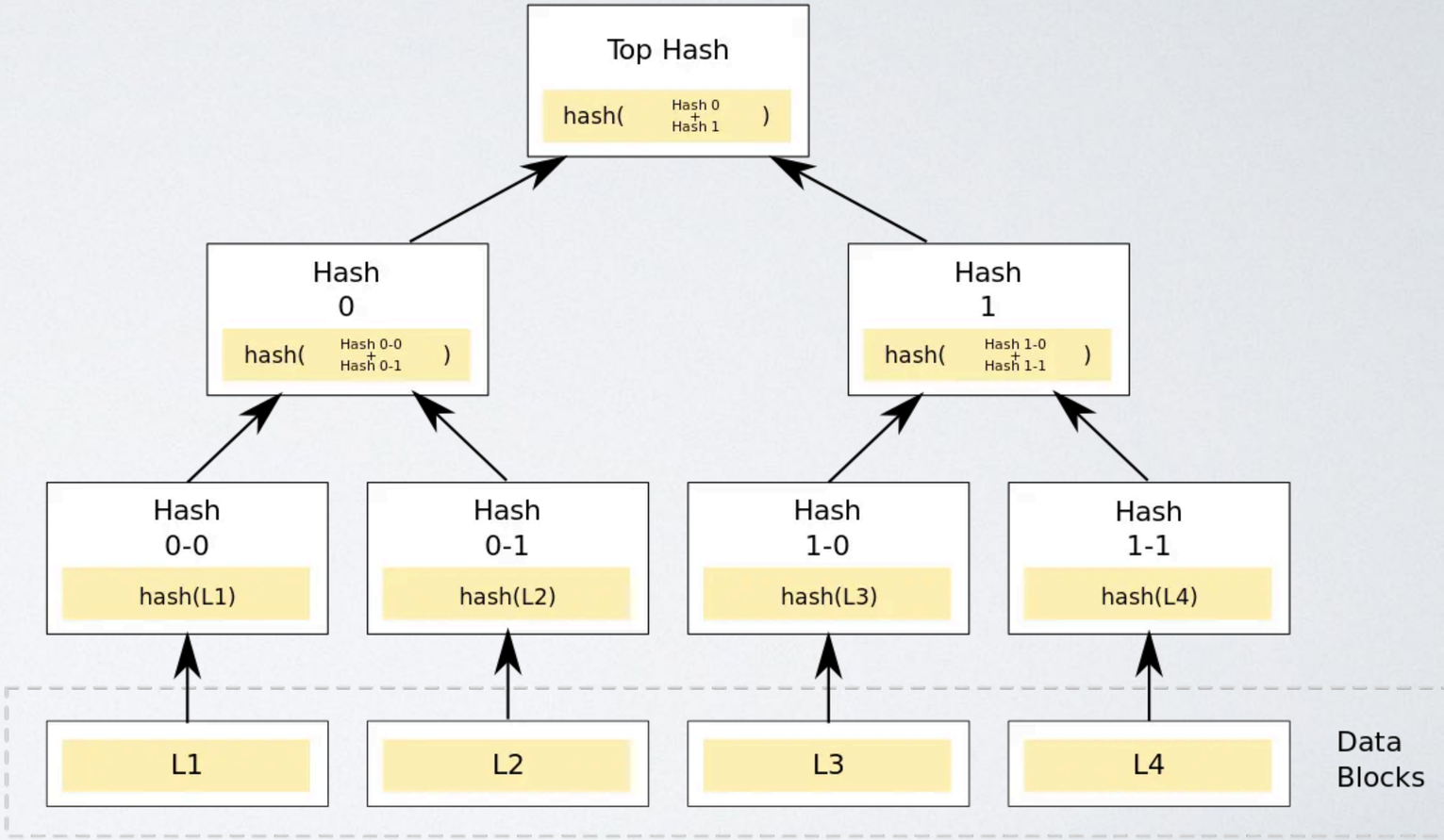
白名单实现

- 映射： mapping(address => bool) public whitelist; //
- 后台离线签名，省 gas ，比较灵活（需要从后端获取）
- **默克尔树 （Merkle Tree）**

默克尔树

构建好树之后，
验证证明只需要对数级的时间！

验证 L1 只需要 Hash0-1 和 Hash 1



默克尔树

- 构建白名单默克尔树 （Merkle Tree）
  - https://github.com/OpenSpace100/blockchain-tasks/tree/main/merkle_distributor
- 在合约里验证是在默克尔树中
  - https://github.com/OpenSpace100/blockchain-tasks/blob/main/solidity_sample_code/MerkleAirdrop.sol

参考文章：https://learnblockchain.cn/article/4613

MultiCall

- 可否多个交易(调用) 封在一个交易里
  - 减少每笔手续费的 21000 base gas
  - 减少数据的冷加载
- 如何在同一个交易里同时调用合约里的多个（次）函数?

MultiCall 写 – 交易

```
function multicall(bytes[] calldata data) external returns (bytes[] memory results) {
        results = new bytes[](data.length);
        for (uint256 i = 0; i < data.length; i++) {
            results[i] = delegateCall(data[I]);
        }
        return results;
}
```

在自己合约实现

openzeppelin-contracts/blob/master/contracts/utils/Multicall.sol

testMultiCall.sol

MULTICALL – 读

- 打包读取和打包写入， 在一次 RPC 请求封装多个请求
  - 降低网络请求
  - 保证数据来自一个区块
  - Viem 有原生集成：https://viem.sh/docs/contract/multicall#multicalladdress-optional
- 打包写入，仅使用于不关注 msg.sender 情况。

https://github.com/mds1/multicall/blob/main/src/Multicall3.sol

MultiCall – 读

```
function aggregate3(Call3[] calldata calls) public payable returns (Result[] memory returnData) {
        uint256 length = calls.length;
        returnData = new Result[](length);
        Call3 calldata calli;
        for (uint256 i = 0; i < length;) {
            Result memory result = returnData[i];
            calli = calls[i];
            (result.success, result.returnData) = calli.target.call(calli.callData);
            assembly {
                  // …
            }
            unchecked { ++i; }
        }
    }
```

公共合约

https://cn.etherscan.com/address/0xcA11bde05977b3631167028862bE2a173976CA11#code

NFT 批量铸造

基于 OpenZepplin 实现

```
function mintNFT(uint256 quantity) public {
    for (uint i = 0; i < quantity; i++) {
        uint index = totalSupply();
        _safeMint(msg.sender, index);
    }
}

function _safeMint( address to,
    uint256 tokenId) internal {
        _balances[to] += 1;
        _owners[tokenId] = to;
        emit Transfer(address(0), to, tokenId);
}
```

mint 1个NFT就需要进行至少2次SSTORE

2 N * SSTORE Gas

_owners 存储

TokenId	owner
1	alice...
2	alice...
3	alice...
4	alice...
5	alice...

ERC721A – Lazy Minting

ERC721A 实现: Lazy Minting， 等到需要的时候再更新槽

铸造 N 个NFT 与1 NFT 的成本相当

```
function mintNFT(address to, uint256 quantity) {
        uint index = totalSupply() + 1;

        _owners[index] = to;
        _balances[to] += quantity;
     …
}

function ownerOf(uint256 _tokenId) external view returns (address)
{
    for (uint256 curr = tokenId; curr >= 0;curr--) {
      address owner = _owners[curr];
      if (owner != address(0)) {
         return owner;
      }
   }
}
```

_owners
存储

TokenId	owner
1	alice...
2	
3	
4	
5	
6	Xxx
7	
8	

ERC721A – 转移

```
function _transfer(address from,address to,uint256 tokenId) private {
    require(from == ownerOf(tokenId));

    balance[from] -= 1;
    balance[to] += 1;
    _owners[tokenId] = to;

    uint256 nextTokenId = tokenId + 1;
    if (_owners[nextTokenId] == address(0) && _exists(nextTokenId)) {
        _owners[nextTokenId] = from;
    }
    emit Transfer(from,to,tokenId);
}
```

TokenId	owner
1	alice...
2	
3	Bob..
4	alice...
5	

练习题

- 组合使用 Multicall (delegateCall) 、 默克尔树、erc20 permit 授权， 实现一个默克尔树优惠价格（100 Token）购买名单， 使用 Multicall. 调用 2 个方法：
  - 合约： AirdopMerkleNFTMarket:
  - permitPrePay : 完成预授权
  - claimNFT( ) ： 验证白名单，如果通过，直接使用 permitPrePay 的授权，转入 100 token， 并转出 NFT

https://decert.me/quests/faa435a5-f462-4f92-a209-3a7e8fdc4d81