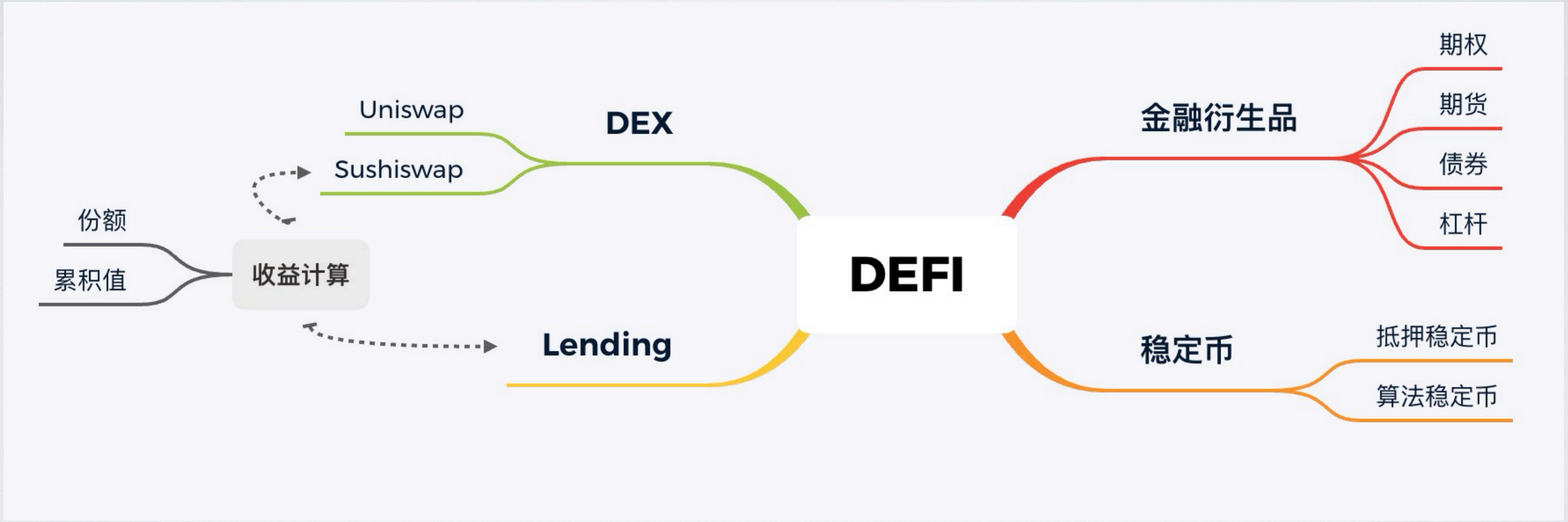

DEFI
DEX
Uniswap
Sushiswap
收益计算
份额
累积值
Lending
金融衍生品
期权
期货
债券
杠杆
稳定币
抵押稳定币
算法稳定币

流动性挖矿（质押奖励）及借贷

Tiny 熊

要点

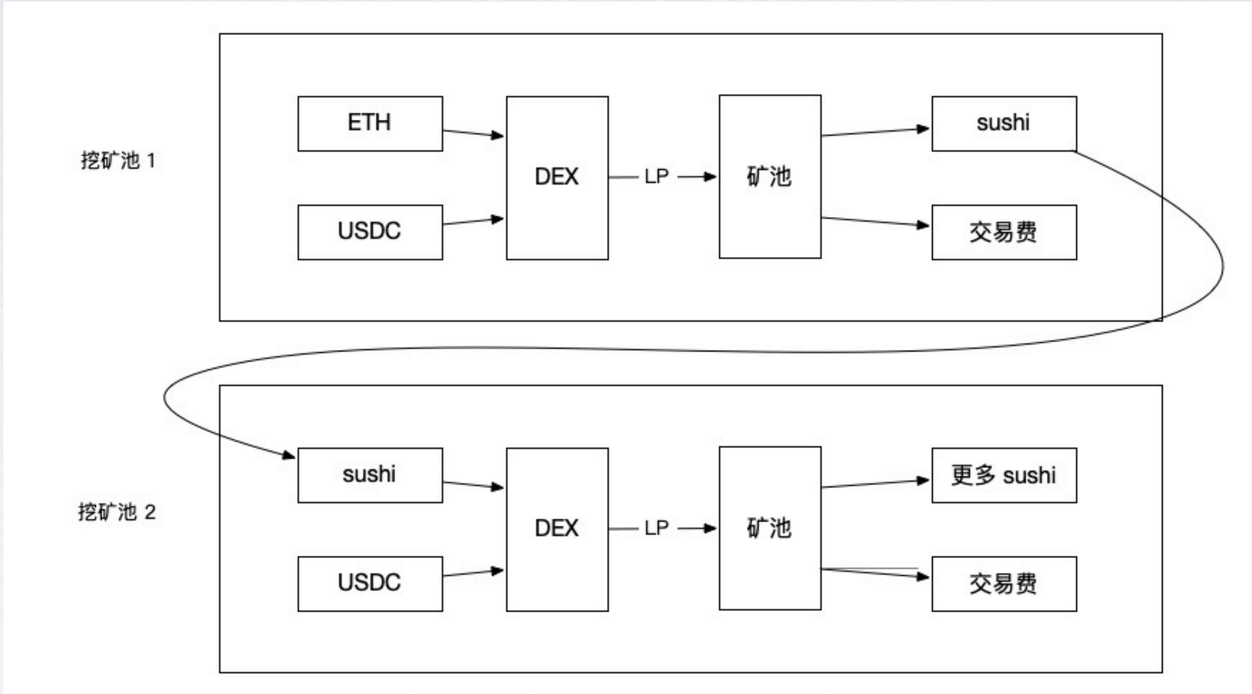
- Sushi 流动性挖矿 (Farming)
- 理解挖矿的奖励计算方式
- 借贷协议：以 Compound 为例，利率模型、清算
- Compound 协议演进及其他的借贷协议
- Vault - ERC4626

SushiSwap

- 早期很成功的 Uniswap 仿盘，对通过“吸血攻击”（vampire attack）吸引 Uniswap 用户
- Uniswap:
  - 提供流动性时才赚取资金池的交易费，撤回流动性不再获得相应的收入
- SushiSwap = Uniswap + 流动性挖矿 (Yield Farming)
- 流动性挖矿：
  - 1. 为流动性提供者奖励
  - 2. 锁定流动性、弥补流动性损失
  - 公平发行协议币（sushi）：去中心化治理、sushi 代币将持续给参与者带来协议收益

Sushi 流动性挖矿

- 用户质押（Staking）LP 或 Sushi，来瓜分每个区块的 100 sushi 奖励

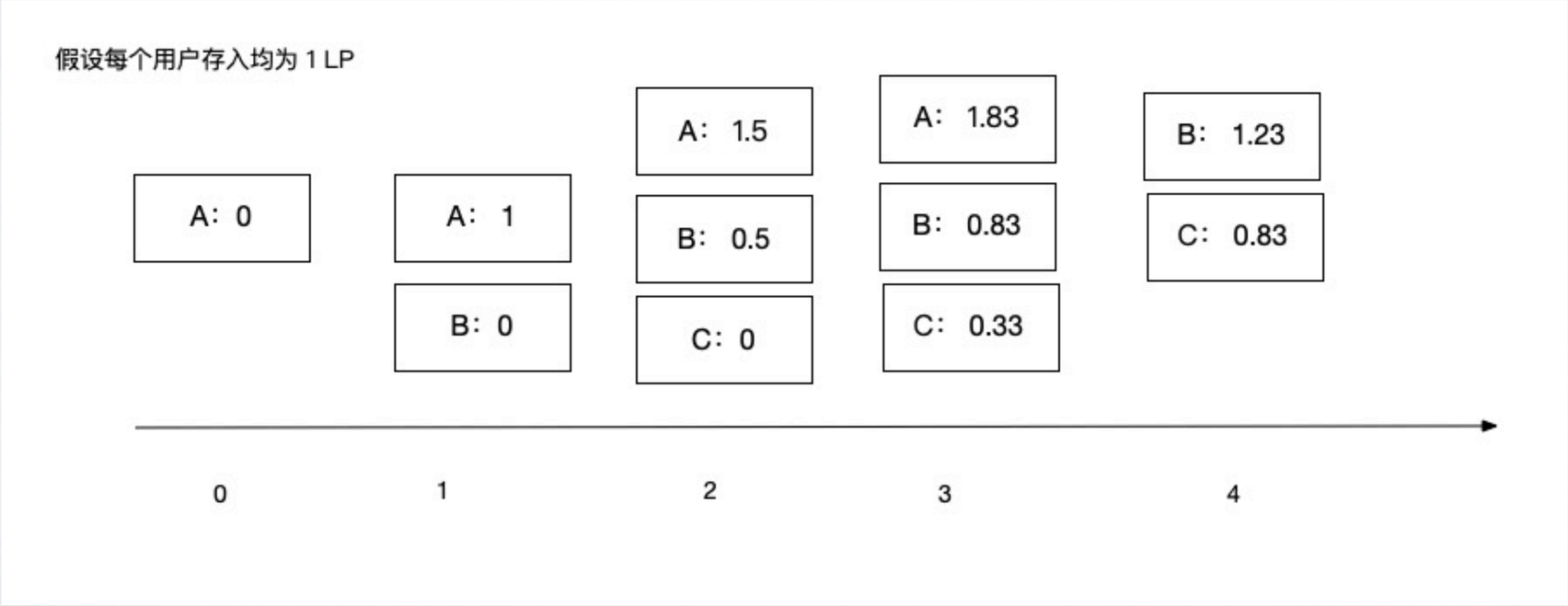


- 正向：有效的激励工具，尤其是更高激励的池2 ，促使人们购买 sushi，推高sushi 价格，同时提高了挖矿池的收益。促进更多人购买 sushi

- 反向：随着挖出的SUSHI越来越多，LP越多 矿工卖SUSHI，某个时间点，卖出的SUSHI比买入的SUSHI多，SUSHI价格开始下降，收益下跌、抛售、进一步下跌，“矿塌了”。

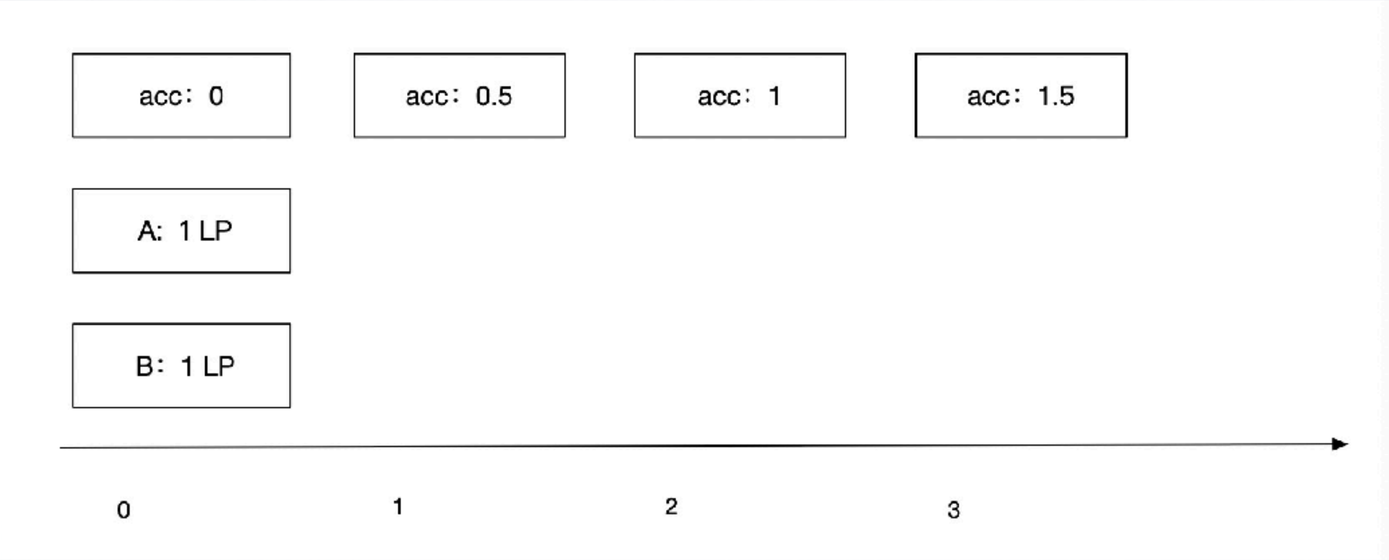
SushiSwap 挖矿算法

- 参与质押的地址，如何分配奖励?
- 常规思路： 用数组记录每个用户的质押，for 循环给每一个地址计算奖励
- GAS 问题…

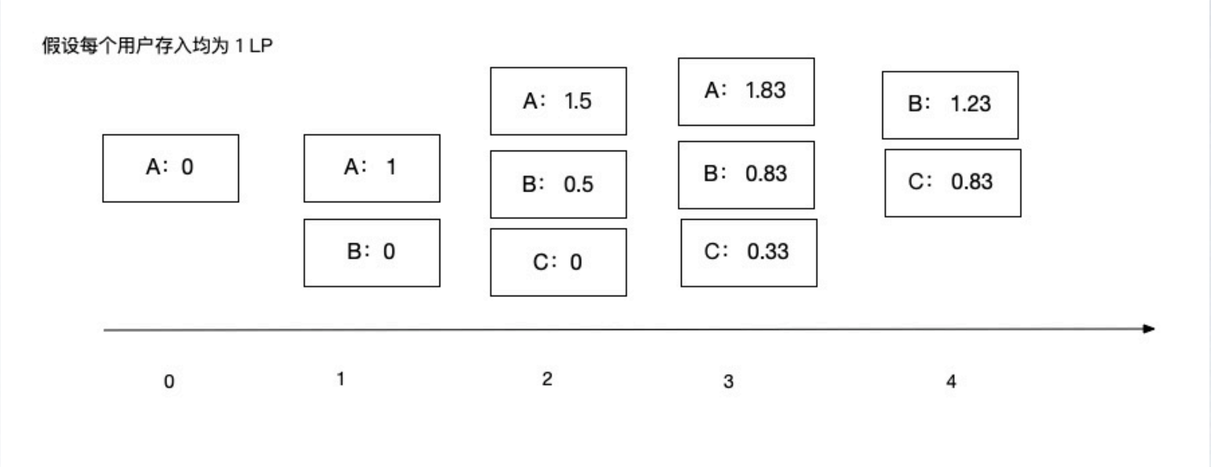


SushiSwap 挖矿算法

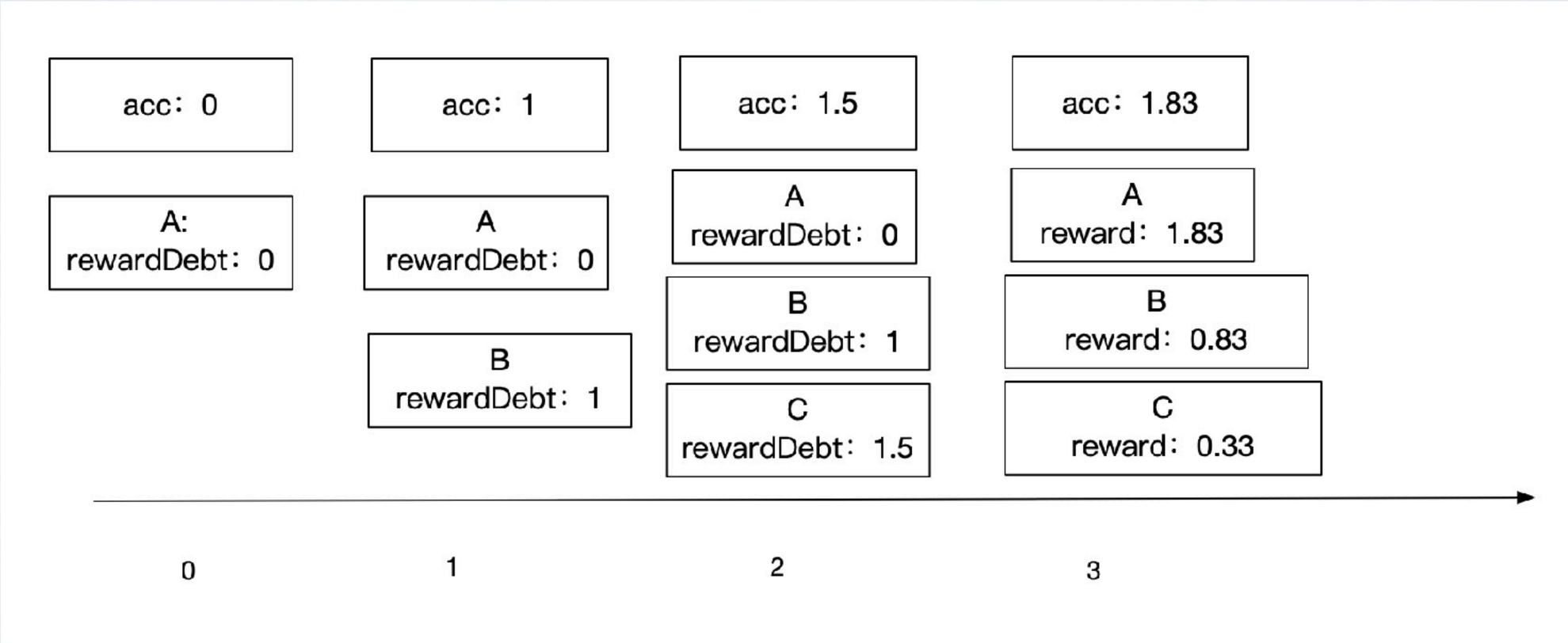
- 奖励 = 每份额 lp 累计奖励 * 份额（数量）
- 每份额lp奖励（累计） += 一段时间的累计sushi奖励 / 当前的中的质押lp数量
- 若用户从中间某点进入，之前的每份额的奖励不能作为用户的奖励，应扣除



acc = 每份额 lp 累计奖励


假设每个用户存入均为 1 LP
A：0
A：1
B：0
A：1.5
B：0.5
C：0
A：1.83
B：0.83
C：0.33
B：1.23
C：0.83
0
1
2
3
4

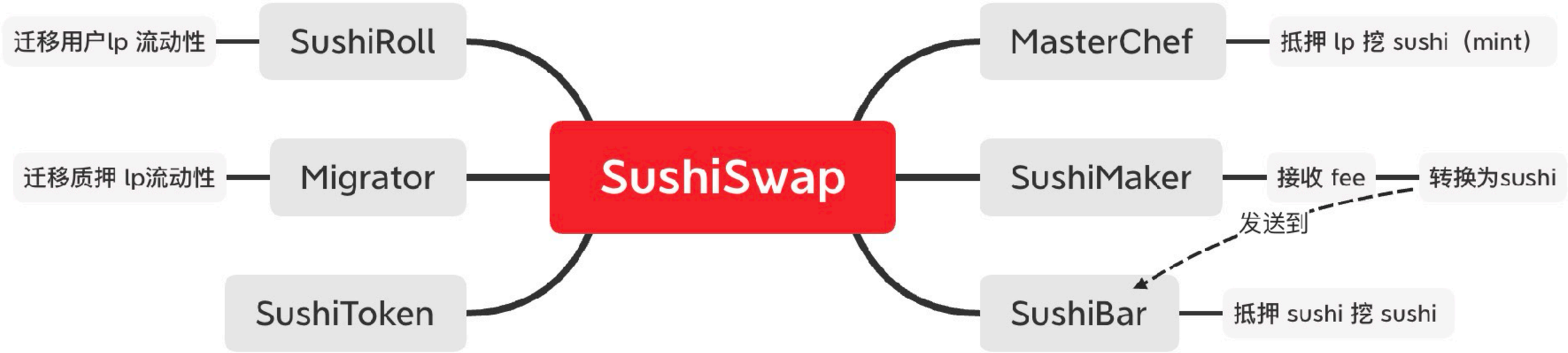
Acc * lp


acc：0
A:
rewardDebt：0
acc：1
A
rewardDebt：0
B
rewardDebt：1
acc：1.5
A
rewardDebt：0
B
rewardDebt：1
C
rewardDebt：1.5
acc：1.83
A
reward：1.83
B
reward：0.83
C
reward：0.33
0
1
2
3

SushiSwap 挖矿算法

- accSushiPerShare： 每个份额可以拿到的**累计**奖励 $$accSushiPerShare = R \cdot \sum_{t=0}^{t} \frac{1}{L(t)}$$
- 记录进入时刻已奖励部分： rewardDebt $$rewardDebt = l \cdot R \cdot \sum_{t=0}^{t_0} \frac{1}{L(t)}$$
- 奖励为： accSushiPerShare * amount - rewardDebt $$Reward_{alice} = l \cdot (R \cdot \sum_{t=0}^{t} \frac{1}{L(t)} - R \cdot \sum_{t=0}^{t_0} \frac{1}{L(t)})$$

Sushiswap – 代码分析



https://github.com/sushiswap/sushiswap

https://github.com/sushiswap/masterchef/blob/master/contracts/MasterChef.sol

Sushiswap – 代码分析

- MasterChef：流动性挖矿
  - add() : 添加某个LP 作为可质押资产
  - deposit()： 存 LP
  - Withdraw(): 取 LP
  - pendingSushi(): 查看收益
- SushiMaker：LP代币(DEX 交易手续) 转 sushi
  - convert()
- SushiBar: 质押 sushi，-> xSuishi, -> 更多 sushi
  - enter： 质押
  - leave(): 取sushi

https://github.com/sushiswap/sushiswap

Vault - ERC4626

- DeFi 协议中大量存在“存入资产后获得份额资产”的机制， 存储资产的合约通常也叫 Vault
- ERC4626 是包装和标准化“存入资产 → 获得收益资产” 的金库接口， 统一“收益金库”的存取逻辑，提升组合性和可集成性。
- ERC4626 继承 ERC20， 本身也是 Token，Token 作为份额凭证，也是退出的依据

$$\text{shares} = \frac{assets \times totalShares}{totalAssets}$$

https://learnblockchain.cn/docs/eips/EIPS/eip-4626/

Vault - ERC4626

```
interface IERC4626 is IERC20 {

    function asset() external view returns (address assetTokenAddress);

    function deposit(uint256 assets, address receiver) external returns (uint256 shares);
    function mint(uint256 shares, address receiver) external returns (uint256 assets);

    function withdraw(uint256 assets, address receiver, address owner) external returns (uint256 shares);
    function redeem(uint256 shares, address receiver, address owner) external returns (uint256 assets);


    function totalAssets() external view returns (uint256 totalManagedAssets);
    function convertToShares(uint256 assets) external view returns (uint256 shares);
    function convertToAssets(uint256 shares) external view returns (uint256 assets);

    function maxDeposit(address receiver) external view returns (uint256 maxAssets);
    function previewDeposit(uint256 assets) external view returns (uint256 shares);
    function maxMint(address receiver) external view returns (uint256 maxShares);
    function previewMint(uint256 shares) external view returns (uint256 assets);
    function maxWithdraw(address owner) external view returns (uint256 maxAssets);
    function previewWithdraw(uint256 assets) external view returns (uint256 shares);
    function maxRedeem(address owner) external view returns (uint256 maxShares);
    function previewRedeem(uint256 shares) external view returns (uint256 assets);
}
```

实现参考：https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/ERC4626.sol

关注 Vault 的通胀攻击

- 在实现 Vault 时，需关注 Vault 在净值极低（或初始阶段）时的逻辑漏洞，通过极小成本铸造大量 shares，稀释其他用户收益，从而获利。

$$\text{shares} = \frac{assets \times totalShares}{totalAssets}$$

攻击推演

1.攻击者存入 1 个 T，获得 1 share

2.攻击者向金库转账（捐赠）100000 个 T

   a. totalAssets = 100001

   b. totalShare = 1

3.若此时 Alice 存入 10000 个 T， 得到的份额：0

   a. 10000 * 1 / 100001 向下取整

4.攻击者赎回 1 share 时，拿回所有的池中资金

避免通胀攻击的几个方式

- 设置最小存款（或流动性）门槛（常见做法），使得攻击者的捐赠成本大幅增加。
- 使用虚拟资产与份额， 模拟金库始终具有一定存款（或流动性）。
- 内部记账（Internal Accounting）- 忽略“捐赠”资产， 但可能引起其他问题。
- 设置初始化时的权限，如totalSupply == 0，只有管理员可操作。

练习题

- 编写质押挖矿合约 StakingPool （理解和掌握单位累计的计算方法），实现如下功能：
  - 用户随时可以质押 ETH (也可用其他 Token 模拟)，赚取项目方 KK Token；
  - 每个区块出块 10 个 KK Token，由质押时长和质押数量来公平分配。
  - 可随时领取 KK Token 奖励
- （加分项）用户质押 ETH 的可存入的一个借贷市场赚取利息
  - 参考思路：找到 借贷市场 进行一笔存款，然后查看调用的方法，在 Stake 中集成该方法。

https://decert.me/challenge/e76599d5-a30c-4678-ba92-fe43c56df1db