

区块链集训营

二期

登链社区 - Tiny熊

练习题

- 部署自己的 ERC20 合约 MyToken
- 编写合约 MyTokenMarket 实现：
 - AddLiquidity(): 函数内部调用 Uniswap V2Router 添加 MyToken 与 ETH 的流动性
 - buyToken(): 用户可调用该函数实现购买 MyToken

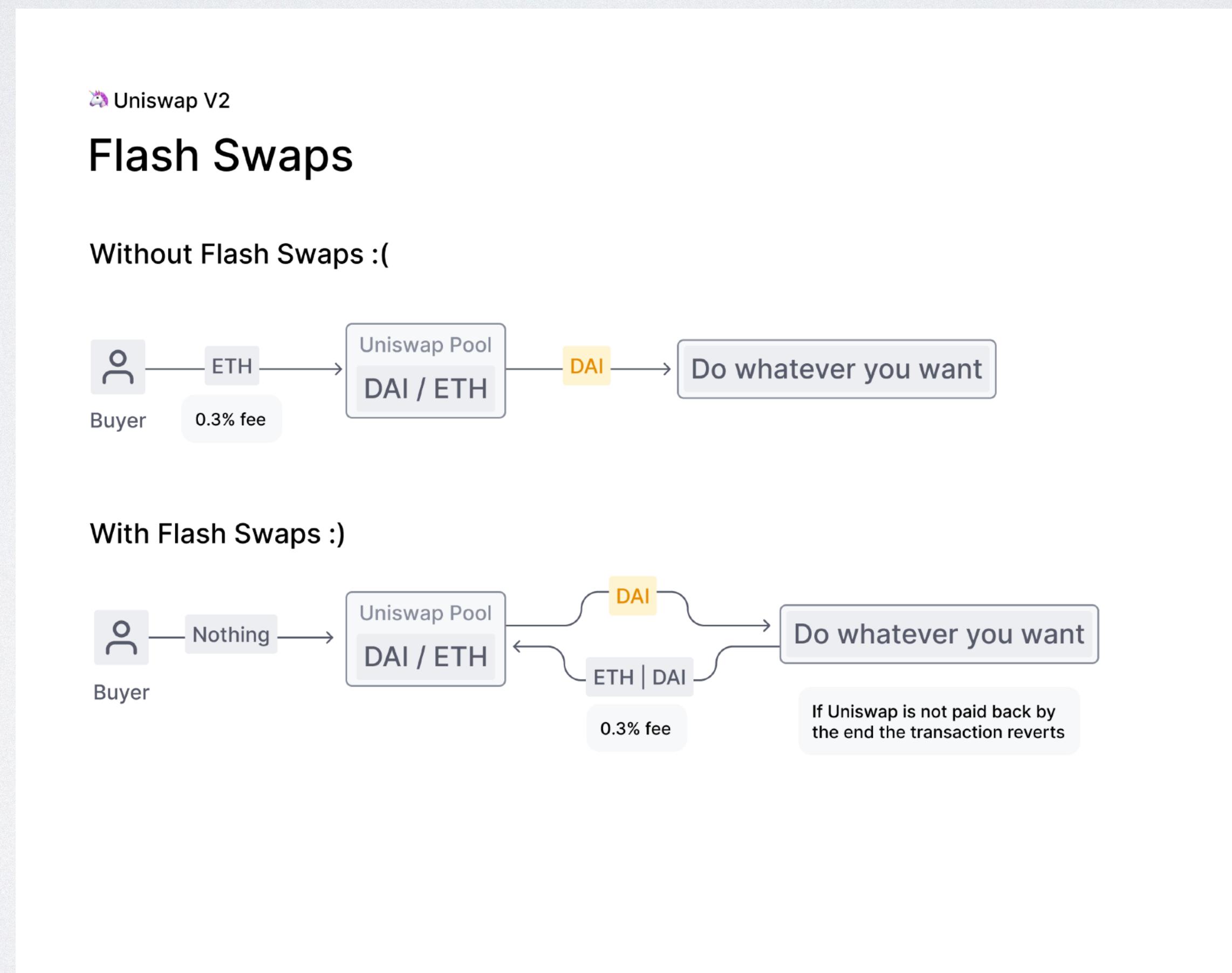
习题解答

代码库：https://github.com/xilibi2003/training_camp_2
文件夹：w5-dex/contracts/MyTokenMarket.sol

Uniswap 集成使用

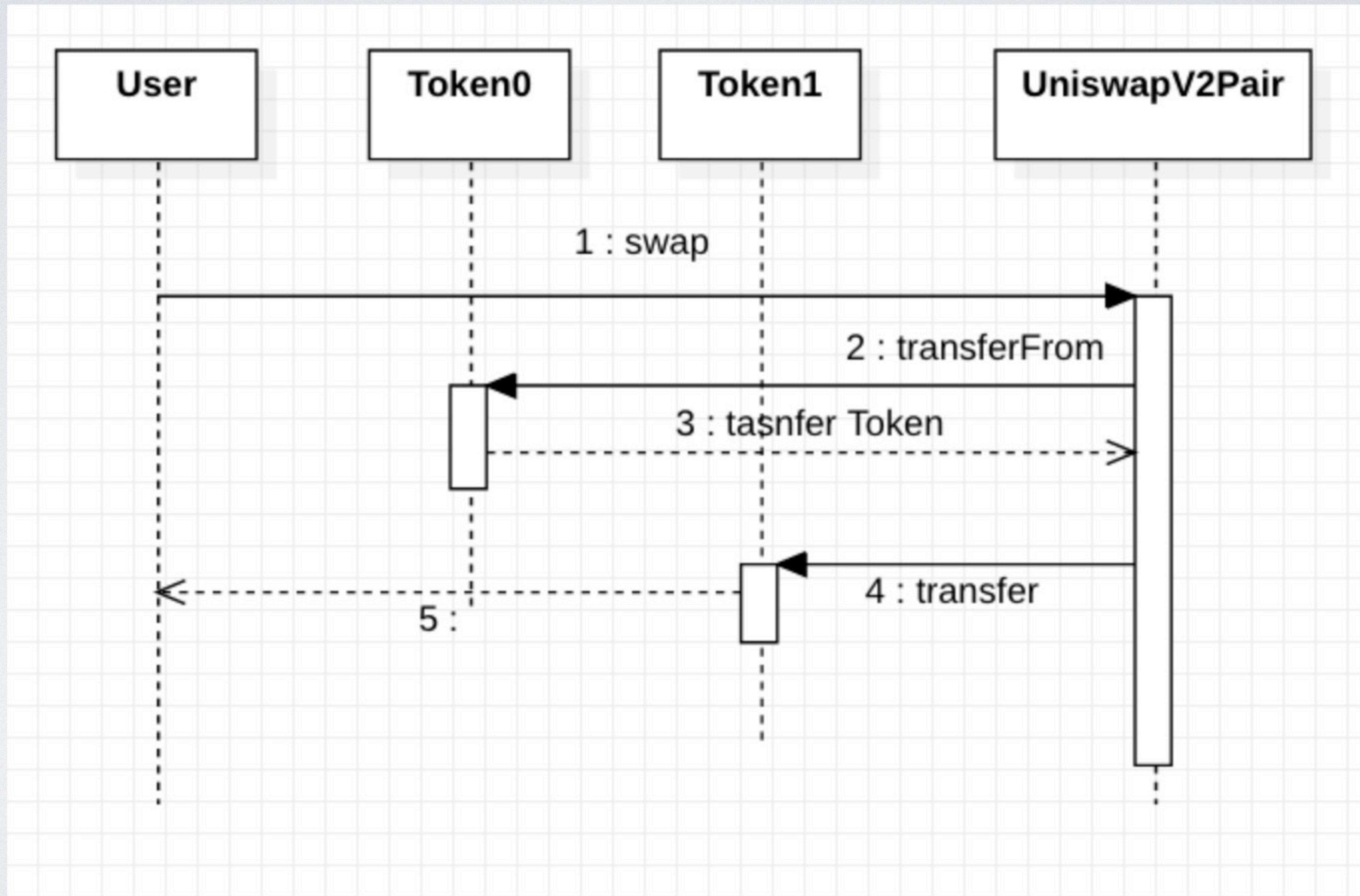
- 交易平台（通过 Router 与 Token 进行交易）
- 闪电兑套利（flash swap）
- 价格预言机（获取时间加权平均价格）

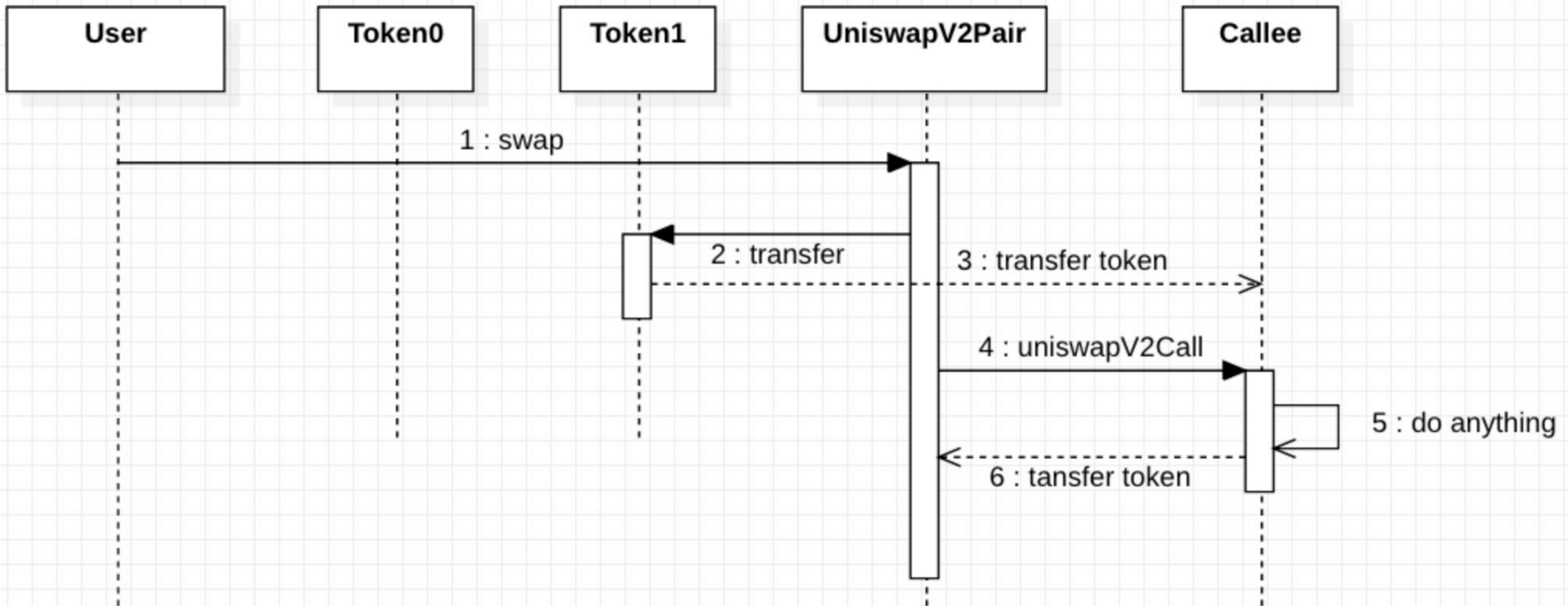
Uniswap 闪电兑



第6周

普通兑换





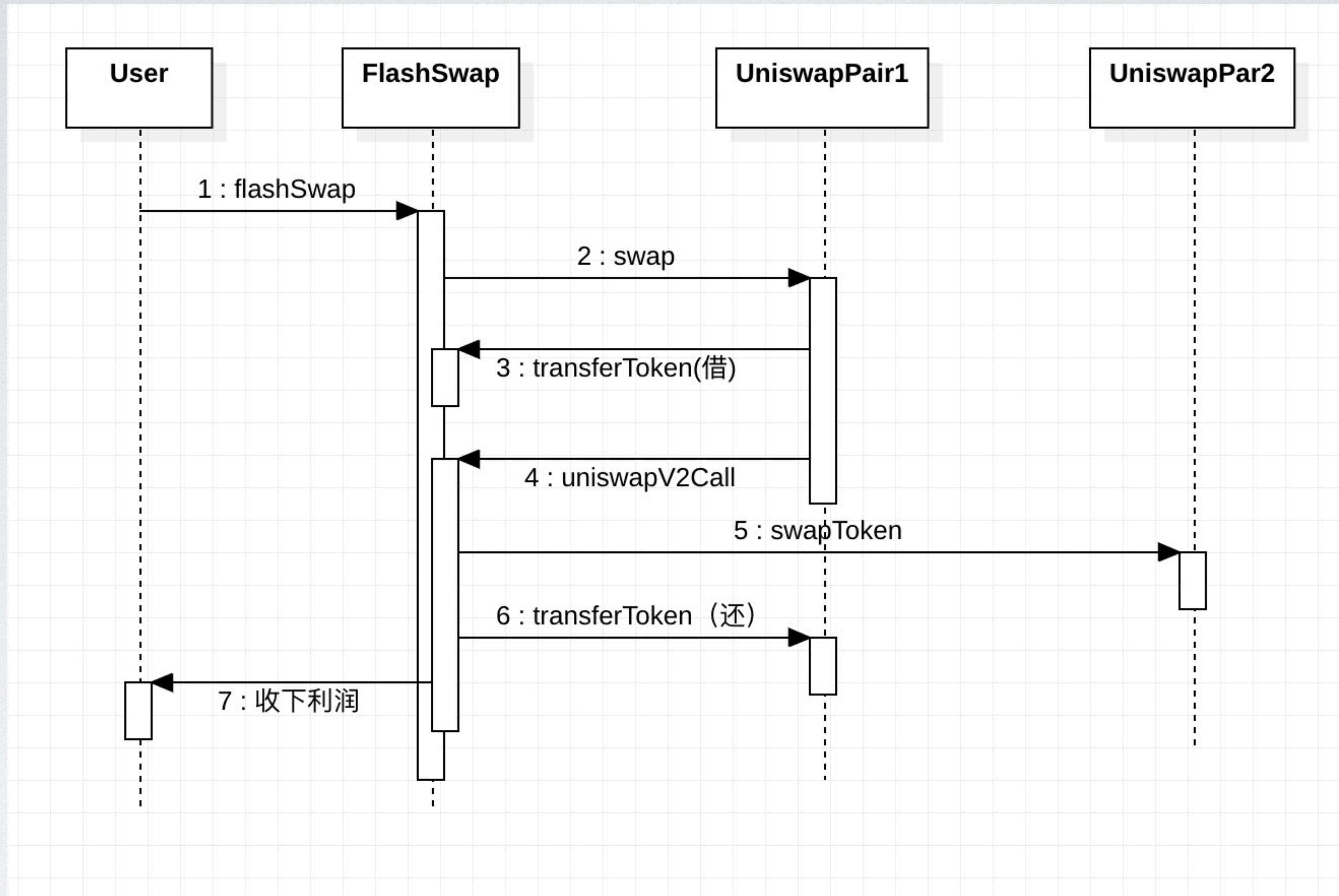
闪电兑换

Uniswap 闪电兑换套利

- 假设有两个 DEX 有价差：
 - 在 Pair1 中 $1 \text{ TokenA} = 2 \text{ TokenB}$
 - 在 Pair2 中 $1.5 \text{ TokenA} = 2 \text{ TokenB}$
- 如何在这两个 DEX 中套利？
 - 我们可以从 Pair1 中借出 2 TokenB , 在 Pair2 中兑换处 1.5 TokenA , 把 1 TokenA 还回 Pair1
 - 净赚 0.5 TokenA

代码库：<https://github.com/xilibi2003/v2-periphery>
文件：contracts/FlashSwap.sol

第6周



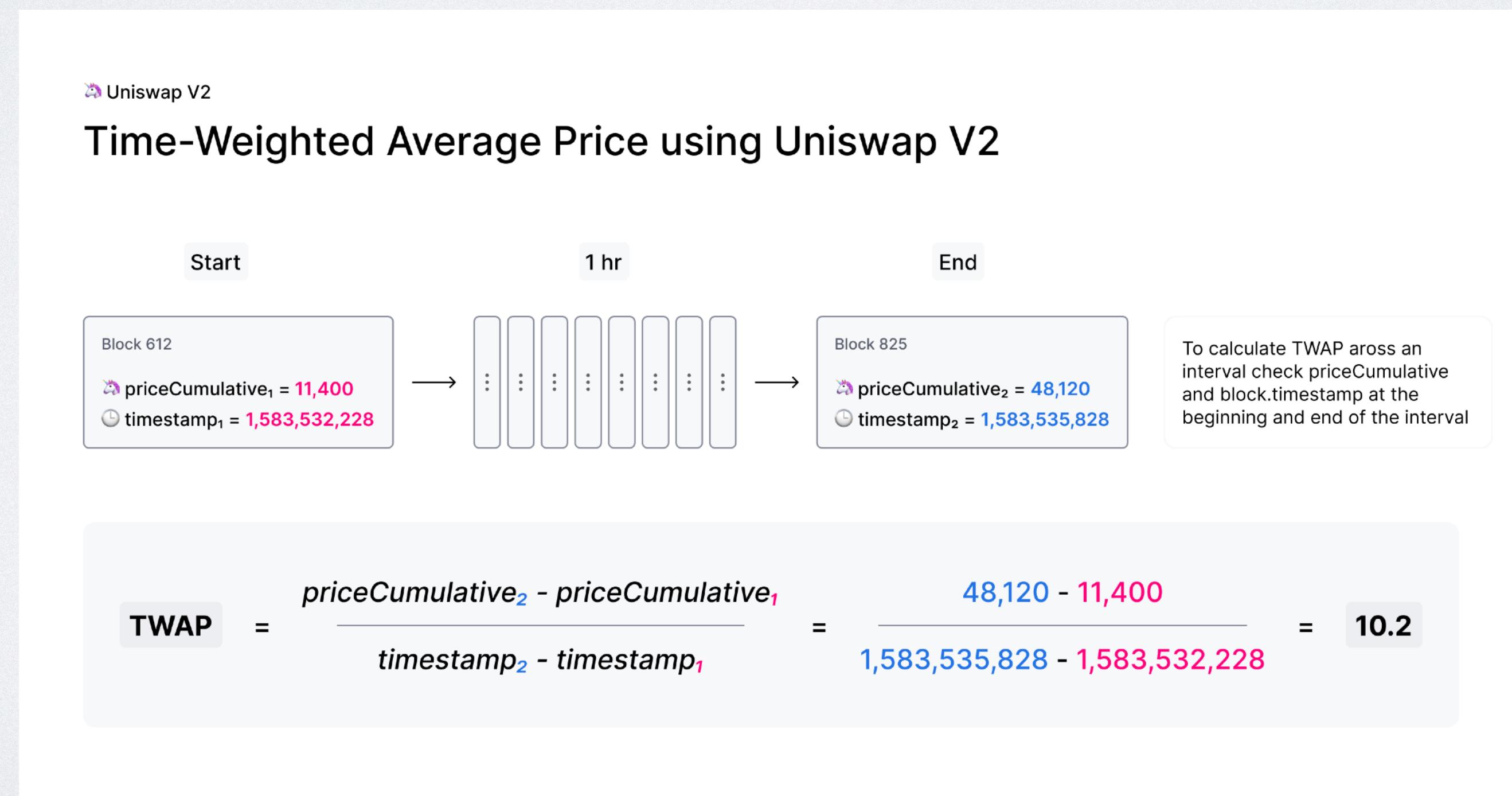
代码库：<https://github.com/xilibi2003/v2-periphery>
文件：contracts/FlashSwap.sol

Uniswap 价格预言机

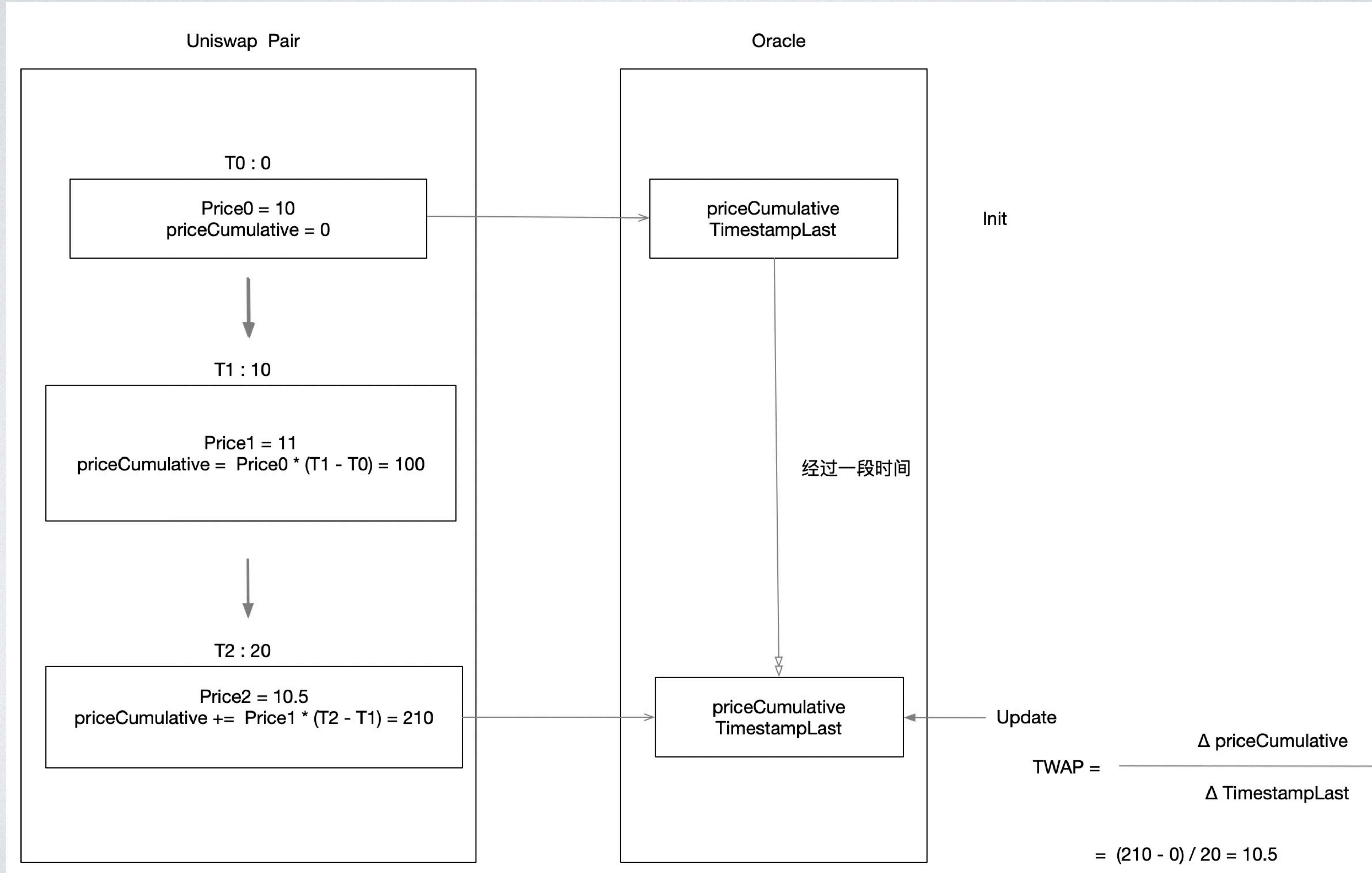
- 即时价格是不安全的，为什么？
 - 当流动性低时，价格波动很大
- 应该使用时间加权价格（一段时间内的价格）

Uniswap TWAP 价格

- TWAP = Time-Weighted Average Price, 即时间加权平均价格
- `price0CumulativeLast`、`price1CumulativeLast` 记录了 token 的累加价格
 - `priceCumulativeLast1 = priceCumulativeLast1 + price2 * timeElapsed;`



第6周



Uniswap TWAP 价格

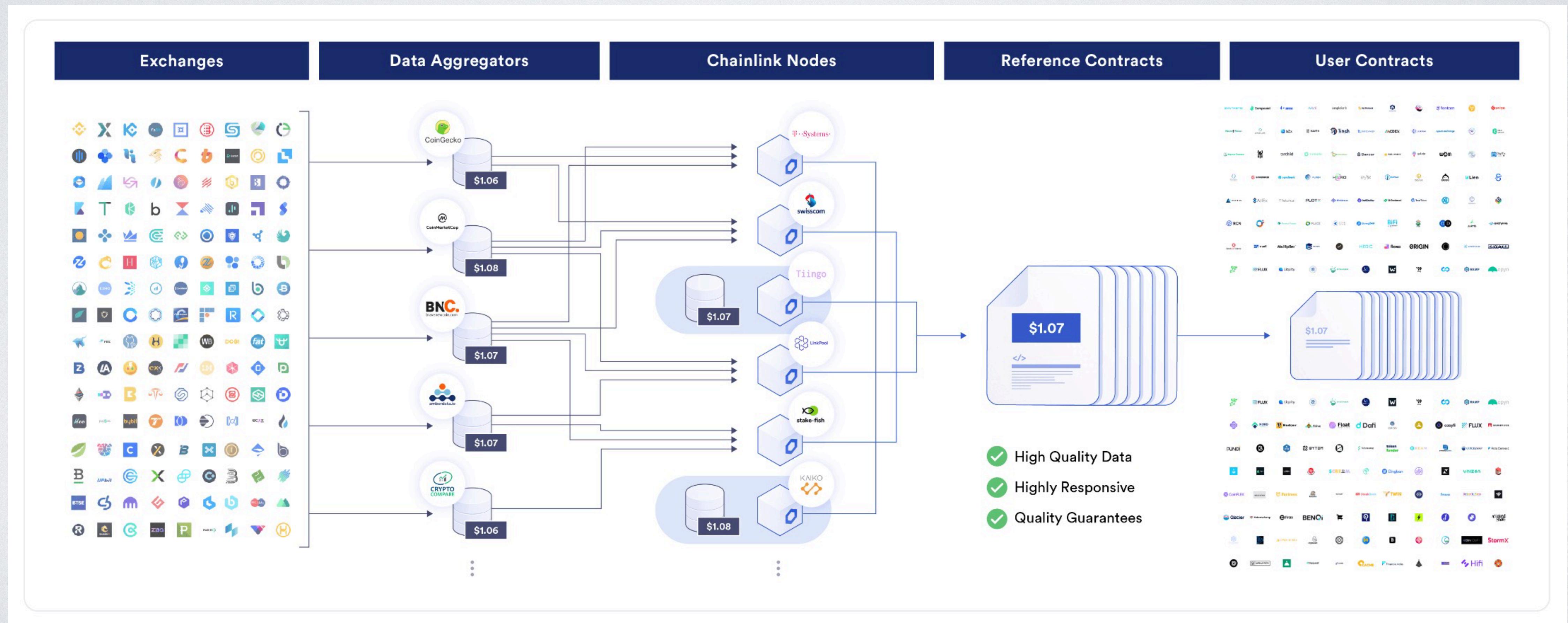
- 在合约里间隔调用 update() 记录累计价格并计算平均价格

代码库：<https://github.com/xilibi2003/v2-periphery>
文件：contracts/examples/ExampleOracleSimple.sol

使用 ChainLink 获得价格

- Data Feeds: 由一组分散的独立节点运营商从许多数据源汇总而来（通常），用户链接对应的Feeds合约即可获取到价格。
- Data Feeds
 - 提供主流资产（数字资产及股票）的价格或指数
 - NFT 地板价
 - 贮备证明
- 文档：<https://docs.chain.link/data-feeds/api-reference/>

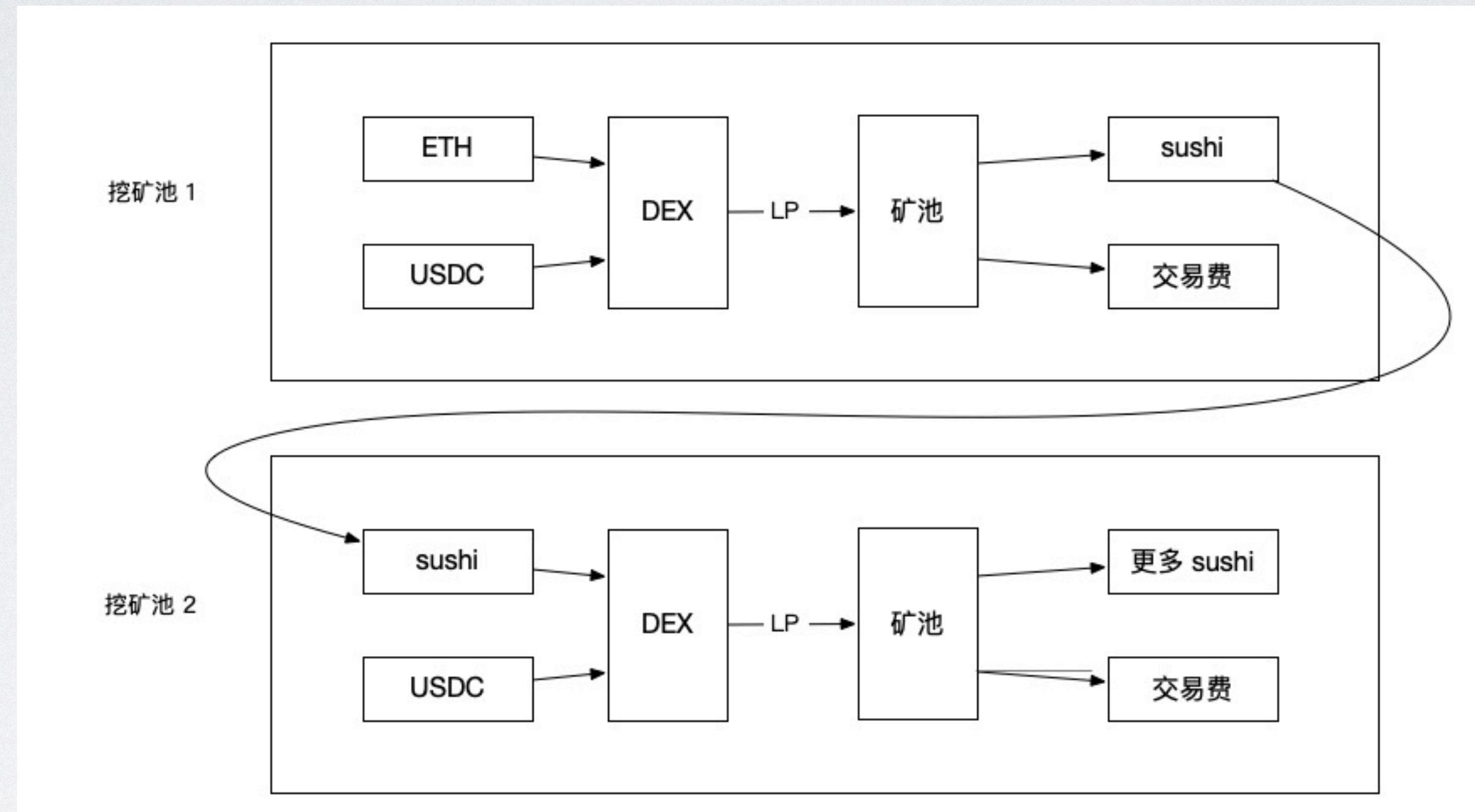
使用 ChainLink 获得价格



SushiSwap

- 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
 - 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 代币可持续给参与者带来协议收益

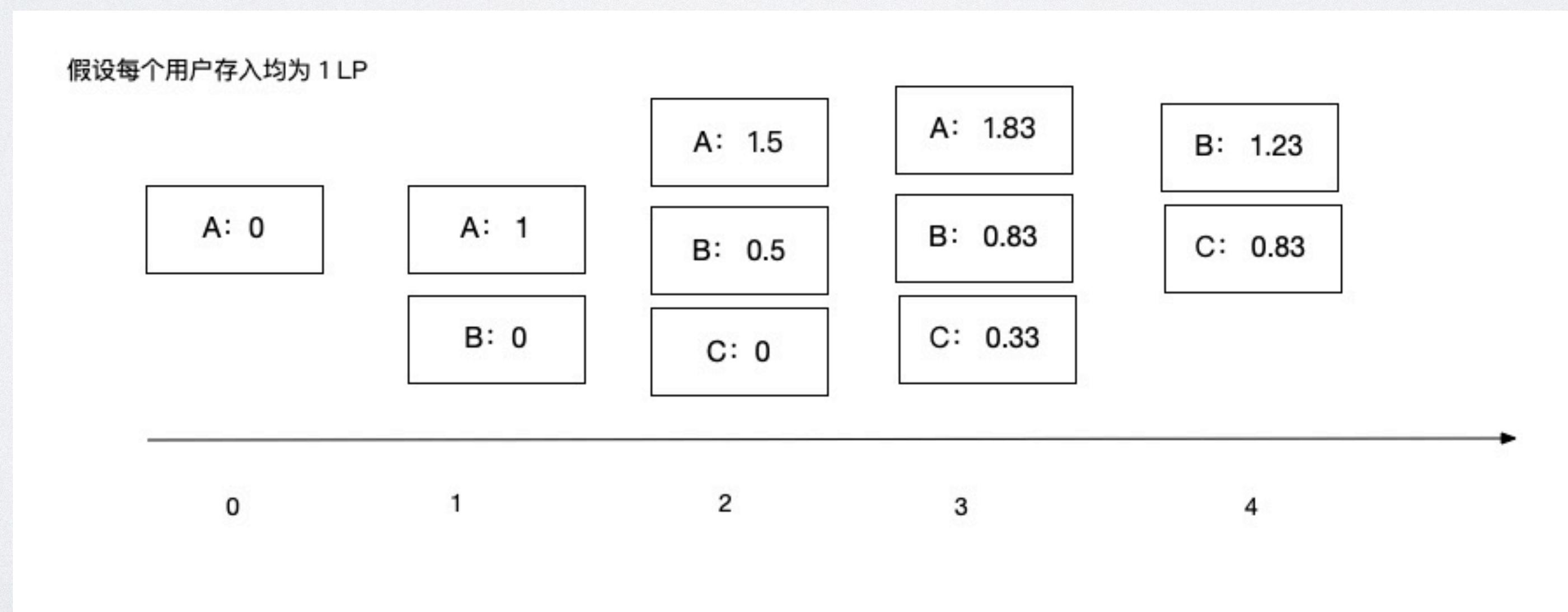
流动性挖矿



- 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，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”。

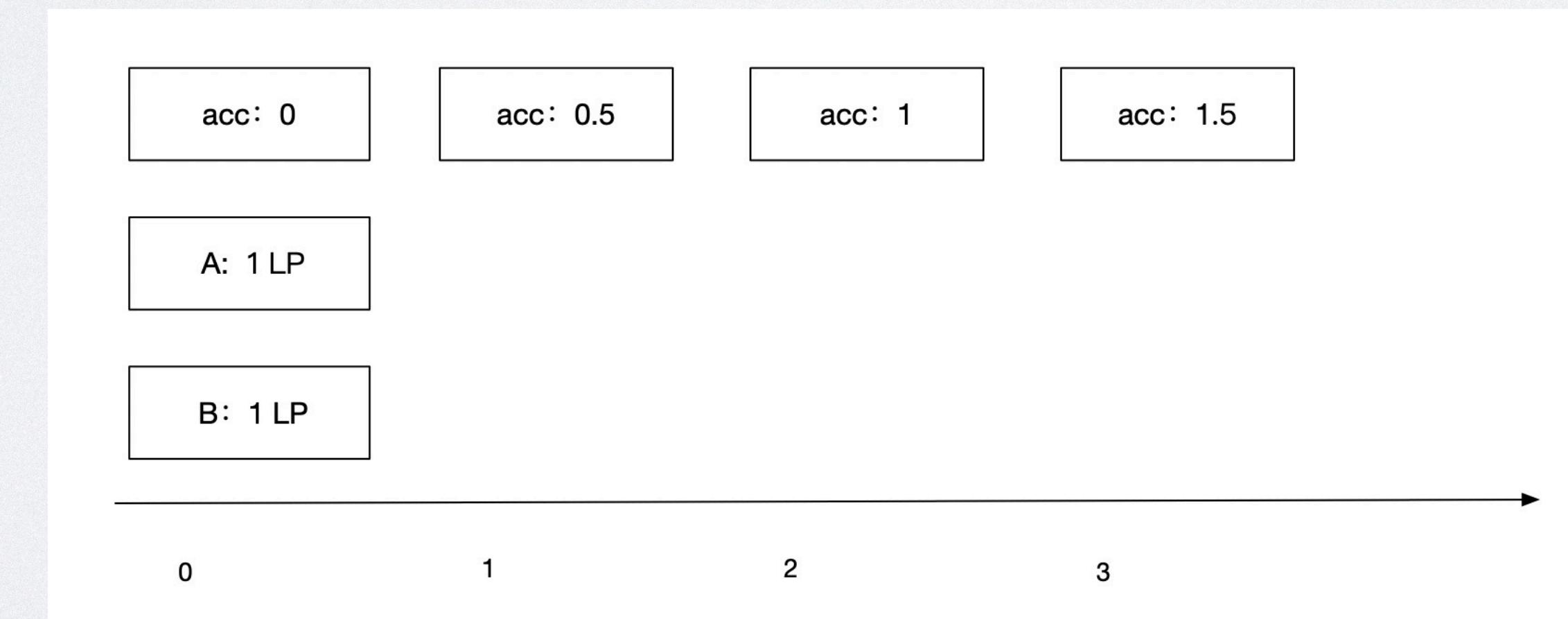
SushiSwap 挖矿算法

- 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奖励 (累计) * 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



SushiSwap 挖矿算法

- $accSushiPerShare$: 每个份额可以拿到的**累计奖励**

$$accSushiPerShare = R \cdot \sum_{t=0}^t \frac{1}{L(t)}$$

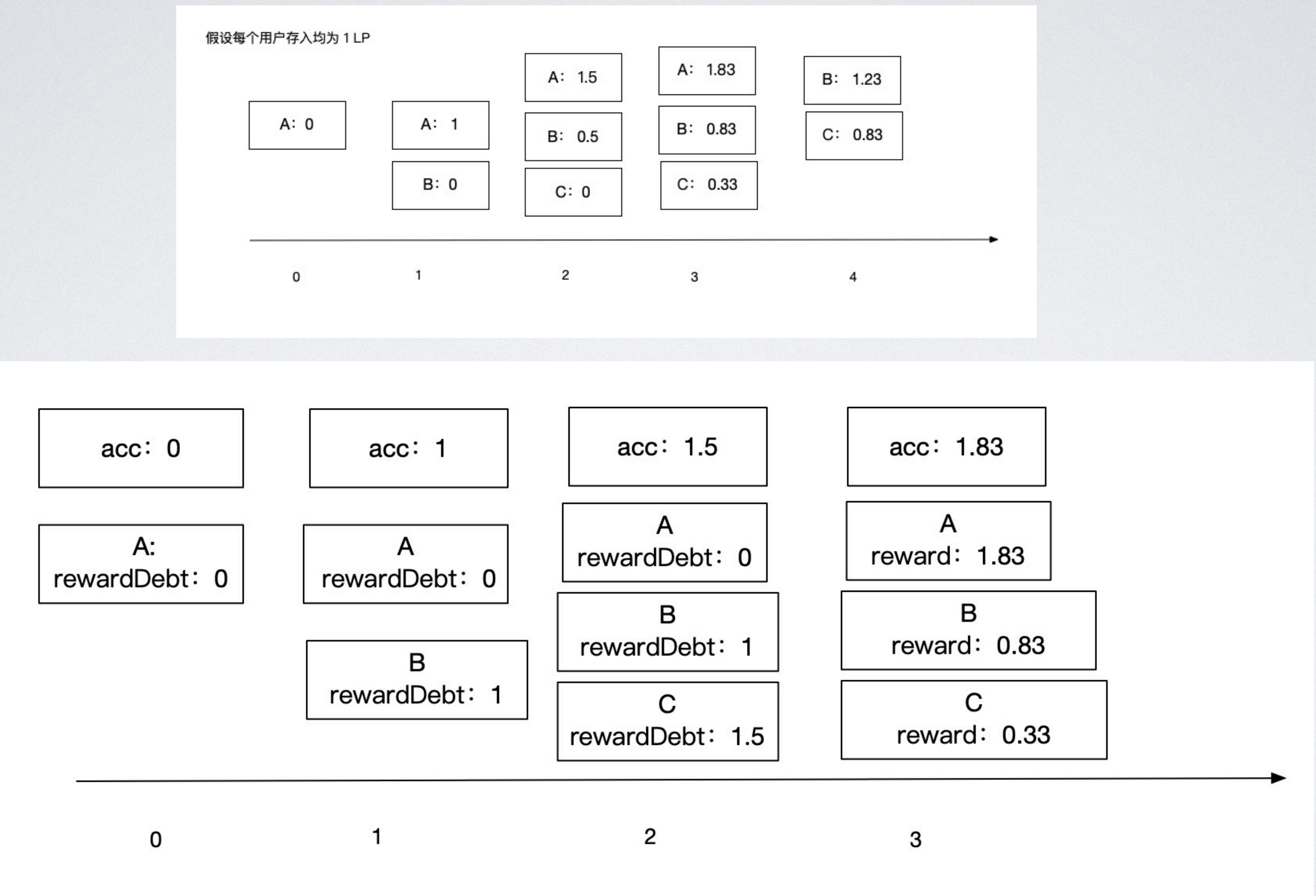
- 记录进入时刻已奖励部分: $rewardDebt$

$$rewardDebt = l \cdot R \cdot \sum_{t=0}^{t_0} \frac{1}{L(t)}$$

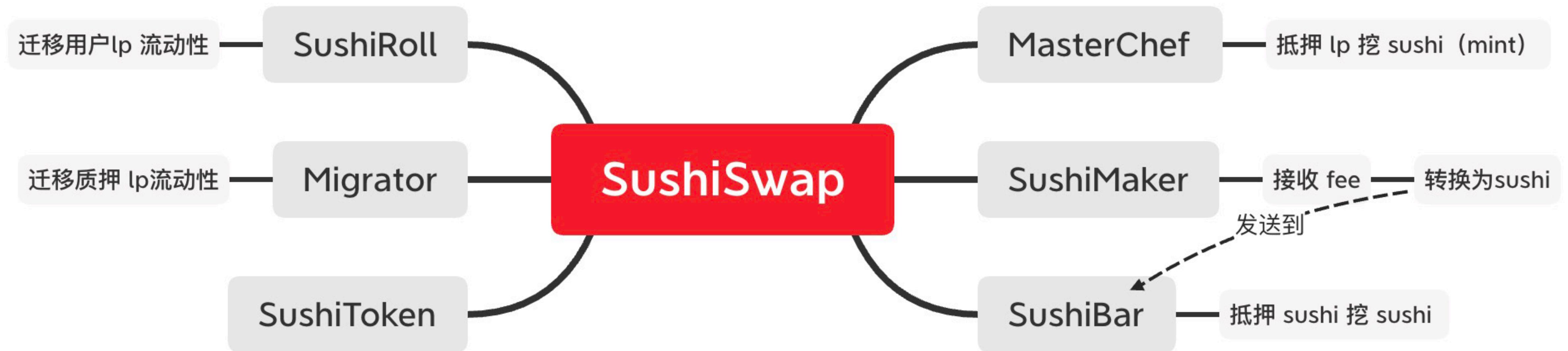
- 奖励为: $accSushiPerShare * amount - rewardDebt$

$$Reward_{alice} = l \cdot (R \cdot \sum_{t=0}^t \frac{1}{L(t)} - R \cdot \sum_{t=0}^{t_0} \frac{1}{L(t)})$$

第6周



Sushiswap – 代码分析



<https://github.com/sushiswap/sushiswap>

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 交易手续费) 转 sushi
 - convert()
- SushiBar: 质押 sushi, -> xSushi, -> 更多 sushi
 - enter: 质押

<https://github.com/sushiswap/sushiswap>

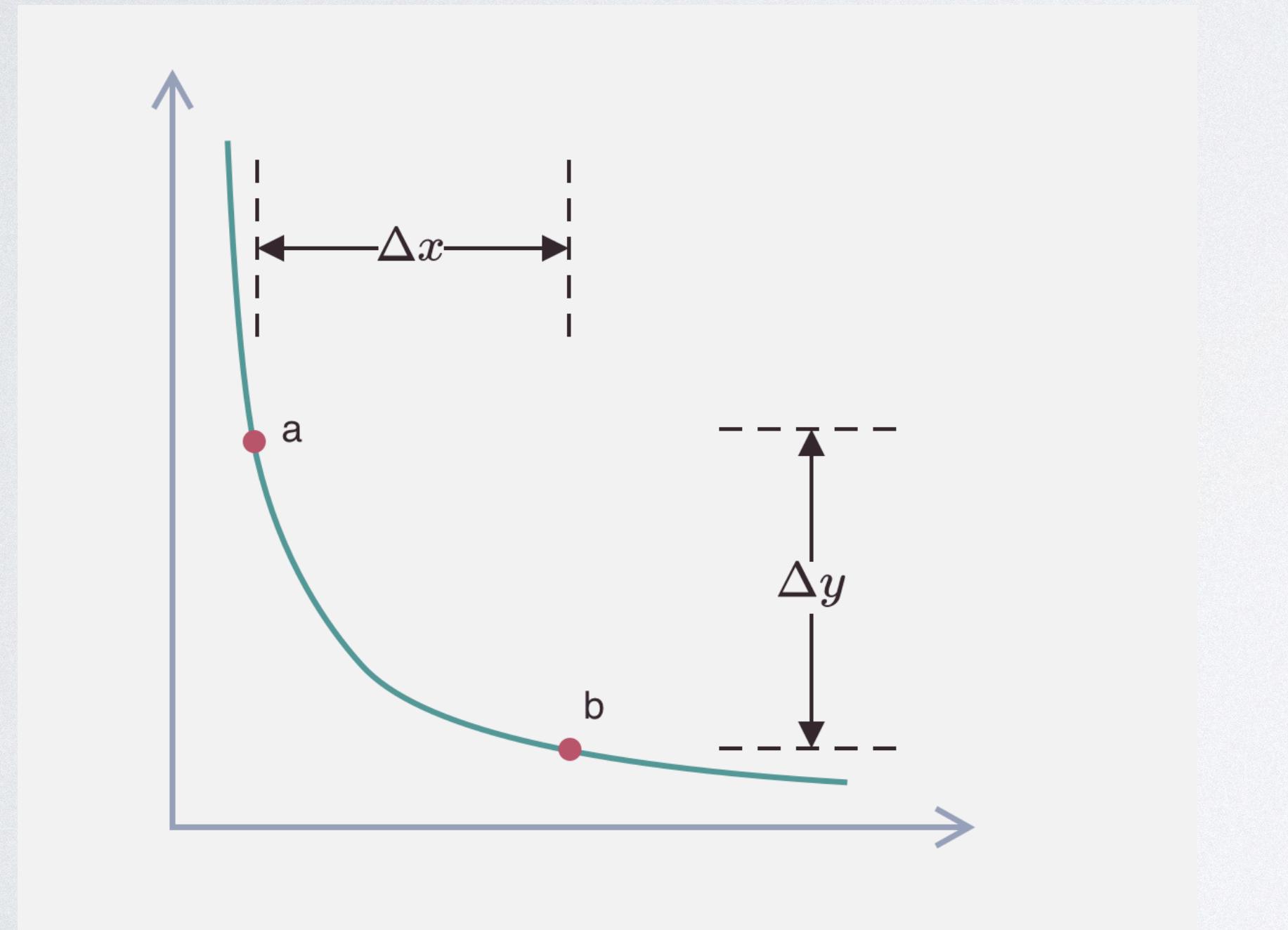
Q & A

练习题

- 在上一次作业的基础上：
 - 完成代币兑换后，直接质押到 MasterChef 挖矿
 - withdraw()：从 MasterChef 提取 Token 方法

Uniswap V2 的问题

- 在所有价格段提供资金 $(0, \infty)$ ，资金利用率低



资金利用率 (Capital Efficiency) = 实际使用的资金 / 所提供的资金

4500 DAI (X) 和 3 ETH (Y)

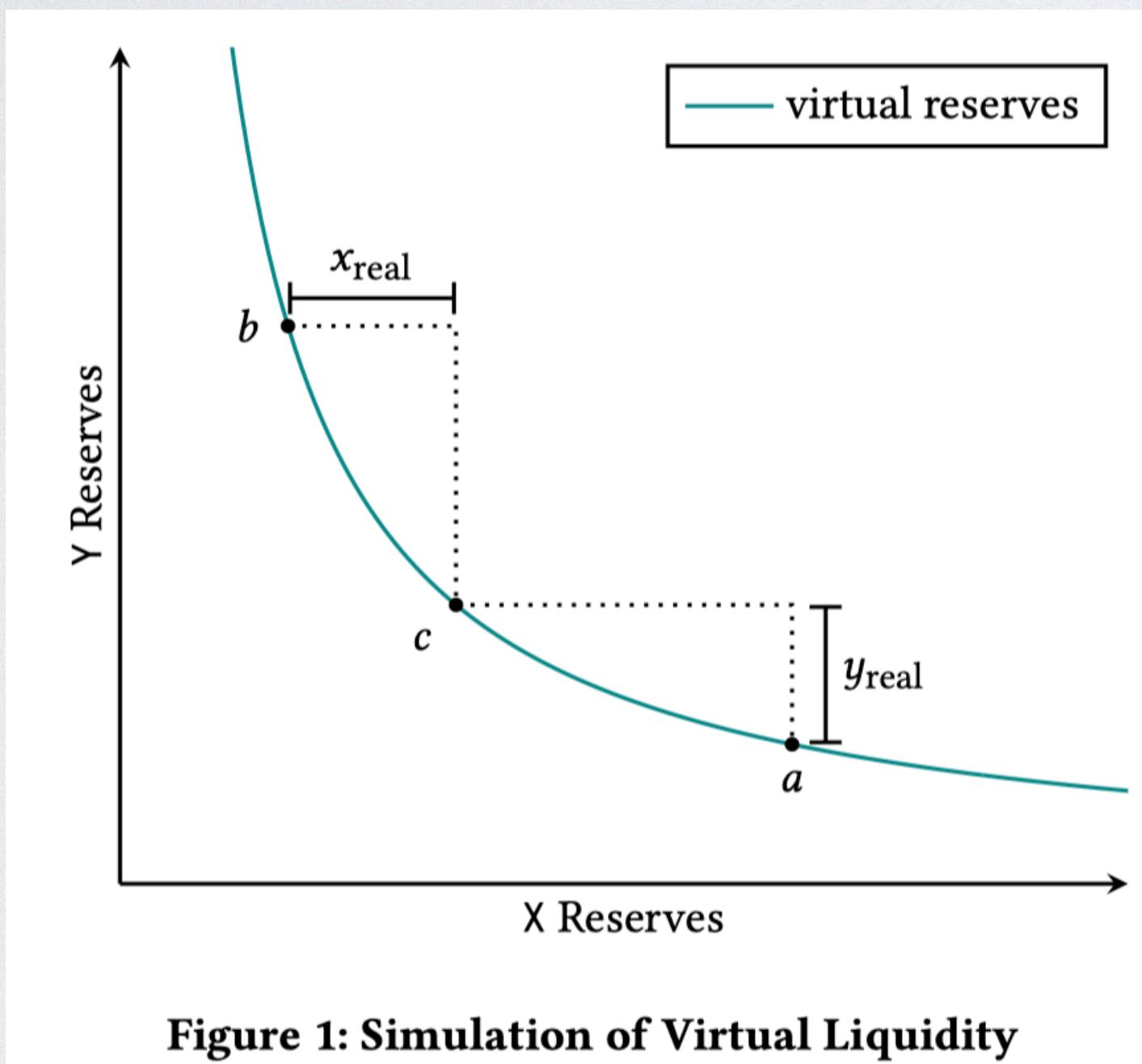
$$K = 4500 * 3 = 13500$$

$$P_1 = 1500 : X_1 = 4500; Y_1 = 3$$

$$P_2 = 1300 : X_1 = 4192.54; Y_1 = 3.22 \quad CF = \Delta X / X_1 = 6.84\%$$

Uniswap V3

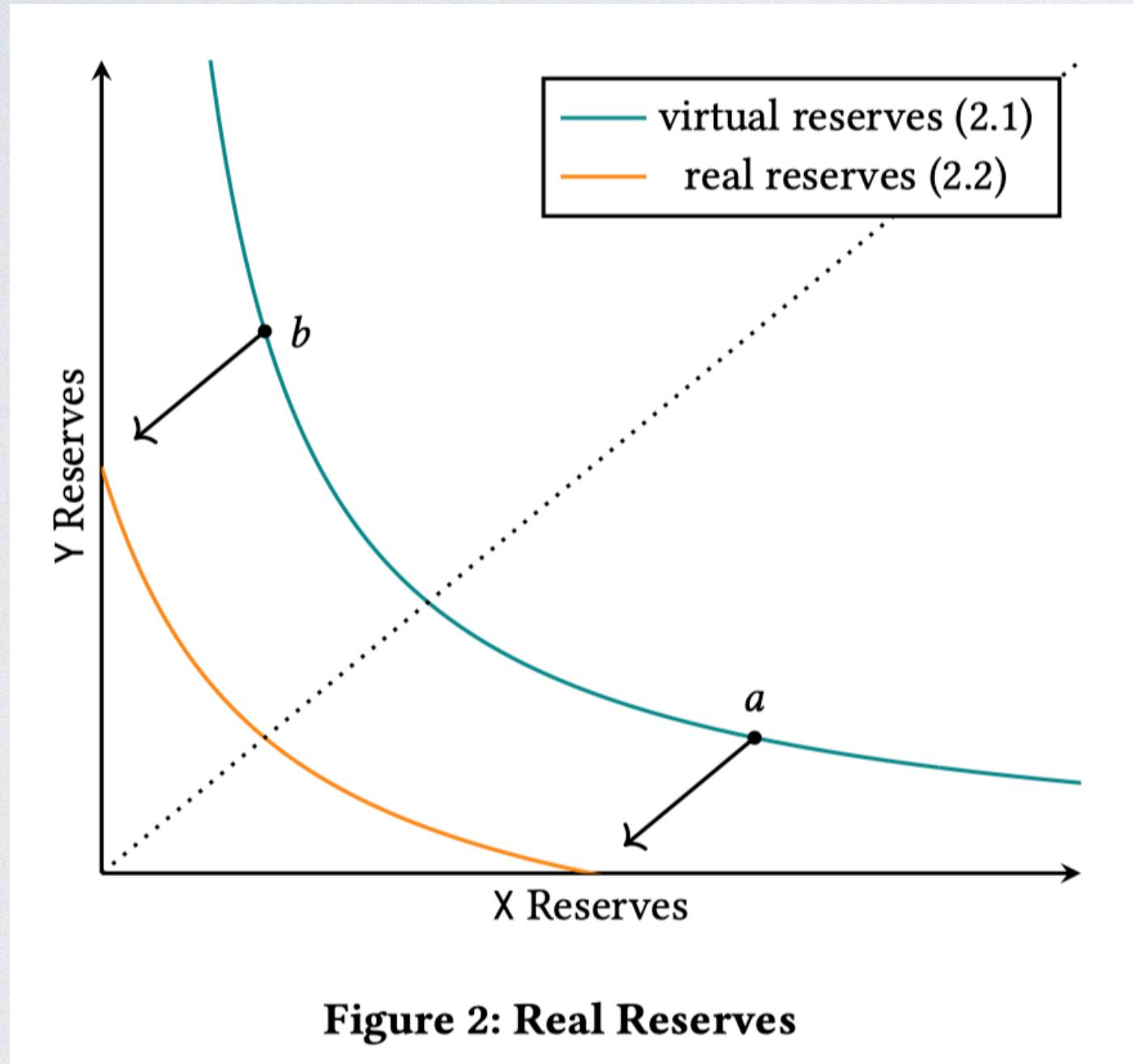
- 最大创新：通过集中式流动性提高资金利用率



1. 价格通常两点之间波动 (a点和b点)
2. c 到 a, 消耗资金为 y_{real}
3. c 到 b, 消耗资金为 x_{real}
4. 若在 $[a, b]$ 提供流动性, 仅需提供 $x_{real} \ y_{real}$

Uniswap V3

- 虚拟流动性



$$(x + \frac{L}{\sqrt{p_b}})(y + L\sqrt{p_a}) = L^2$$

- 引入虚拟流动性 $K = (x+x_{\text{virtual}}) * (y+y_{\text{virtual}})$
- L: 流动性, p: 价格 (y的)

$$K = L^2$$

$$p = y/x$$

$$x * y = L^2$$

$$x * (p * x) = L^2$$

$$p * x^2 = L^2$$

$$x^2 = L^2/p$$

若 x_{real} 用完, 需要的虚拟流动性为

$$x = L/\sqrt{p}_b, y = L * \sqrt{p}_a$$

Uniswap V3

- 流动性

1. L: 流动性, p: y的价格

$$K = L^2$$
$$p = y/x$$

$$x = L/\sqrt{p}$$
$$y = L * \sqrt{p}$$

代码中存 L和sqrt(P)

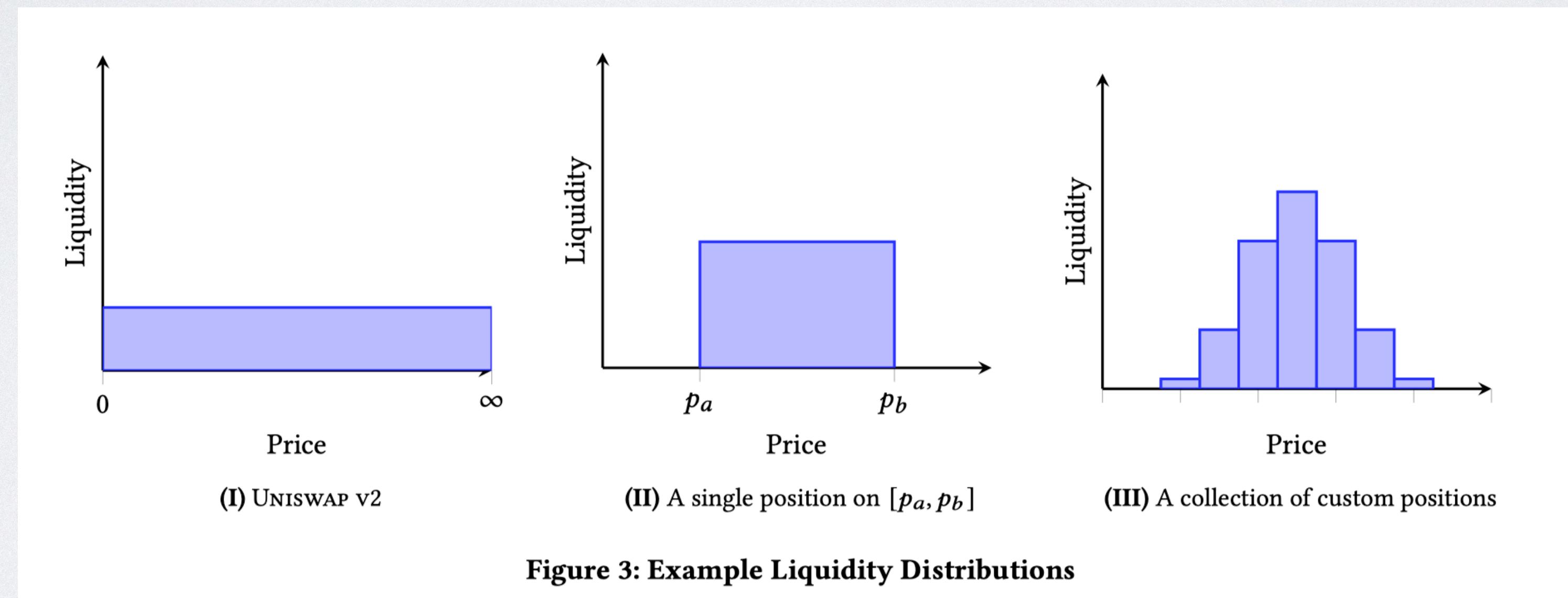
若已知L和sqrt(P), 就可以推导出资金使用量 $x \times y$

$$L = \frac{\Delta Y}{\Delta \sqrt{P}}, \Delta x = L * \Delta \frac{1}{\sqrt{P}}$$

流动性是价格变化所需的资金量

Uniswap V3

- 流动性聚合



Uniswap V3

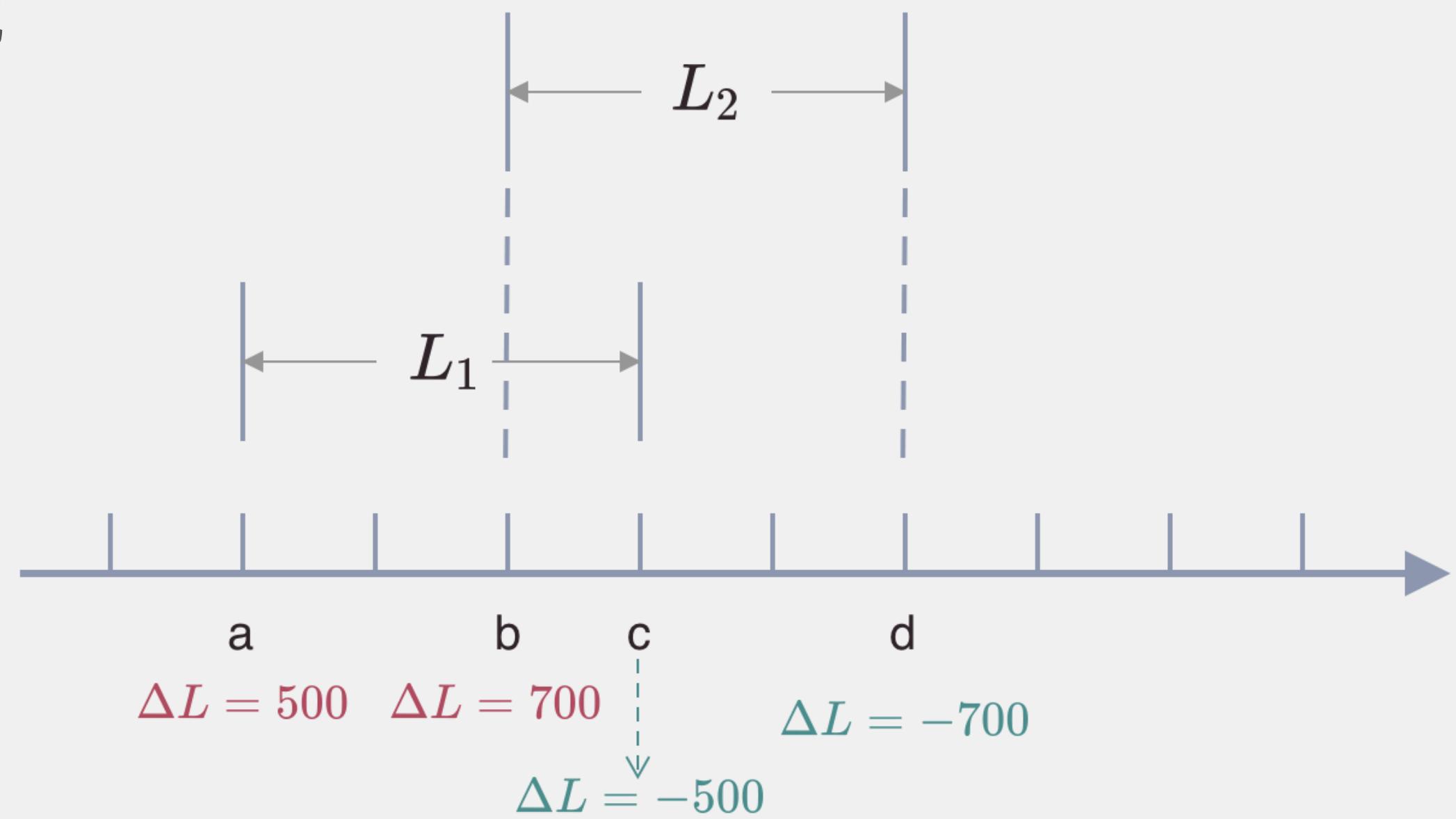
- 叠加流动性：引入Tick，使用等比数列将价格区间分段，添加流动性只能选择对应的离散点作为价格边界
- $P_i = 1.0001^i, \sqrt{P} = \sqrt{1.0001}^i, \sqrt{P_{i+1}} = \sqrt{1.0001} \cdot \sqrt{P_i}$
- 价格序号 i 称为 Tick (可为负数)
- 价格区间： $(\sqrt{1.0001}^{int24.min} - \sqrt{1.0001}^{int24.max})$
- tickspacing: 并不是每一个价格需要都可用，定义价格密度

费率	tickspacing	建议的使用范围
0.05%	10	稳定币交易对
0.3%	60	适用大多数交易对
1%	200	波动极大的交易对

Uniswap V3

- 管理流动性的变化量

跟踪中的流动性及每个 tick 引起的流动性变化



Uniswap V3

- 兑换

$$L = \frac{\Delta Y}{\Delta \sqrt{P}}, \Delta x = L * \Delta \frac{1}{\sqrt{P}}$$

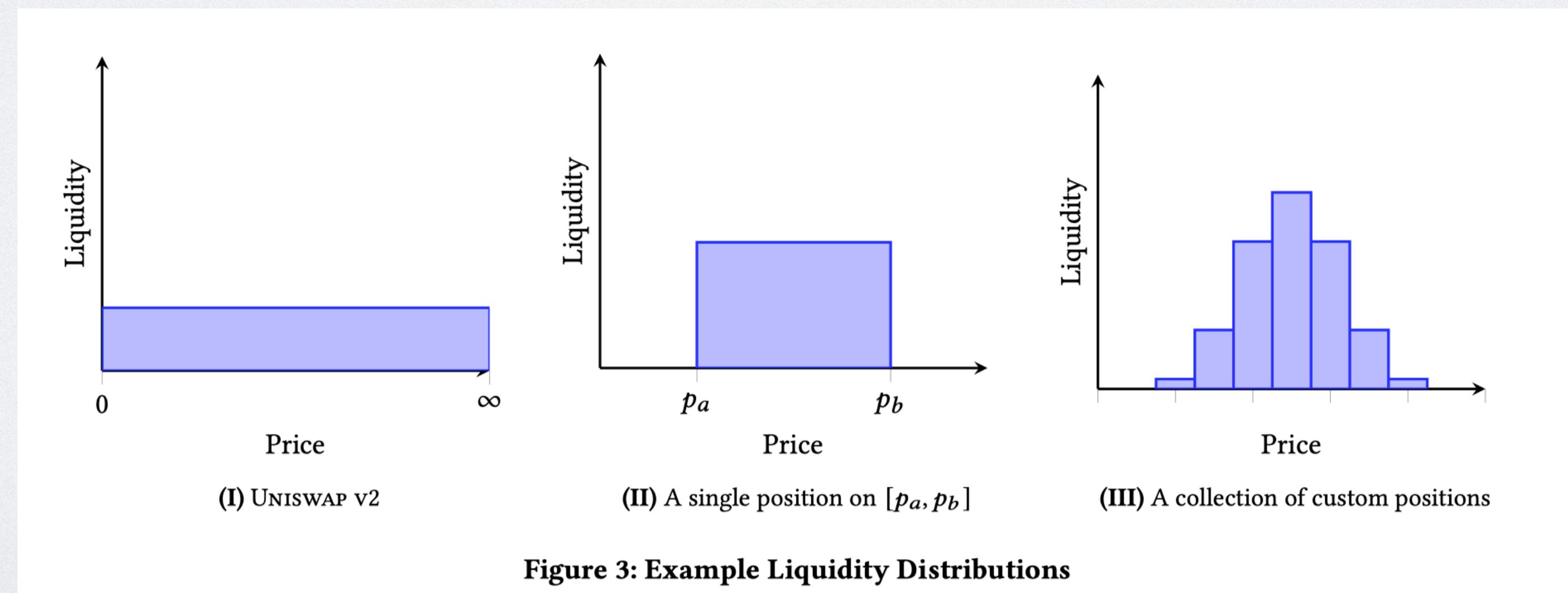
Uniswap V3

- 手续费：多个不同LP如何计算各自的手续费？

若统一价格区间：

用户手续费： $fee = deltaL * (fee_{global}/liquidity_{global})$

交易手续费累加值更新： $feeGrowthGlobal += deltafee/liquidity$



Uniswap V3

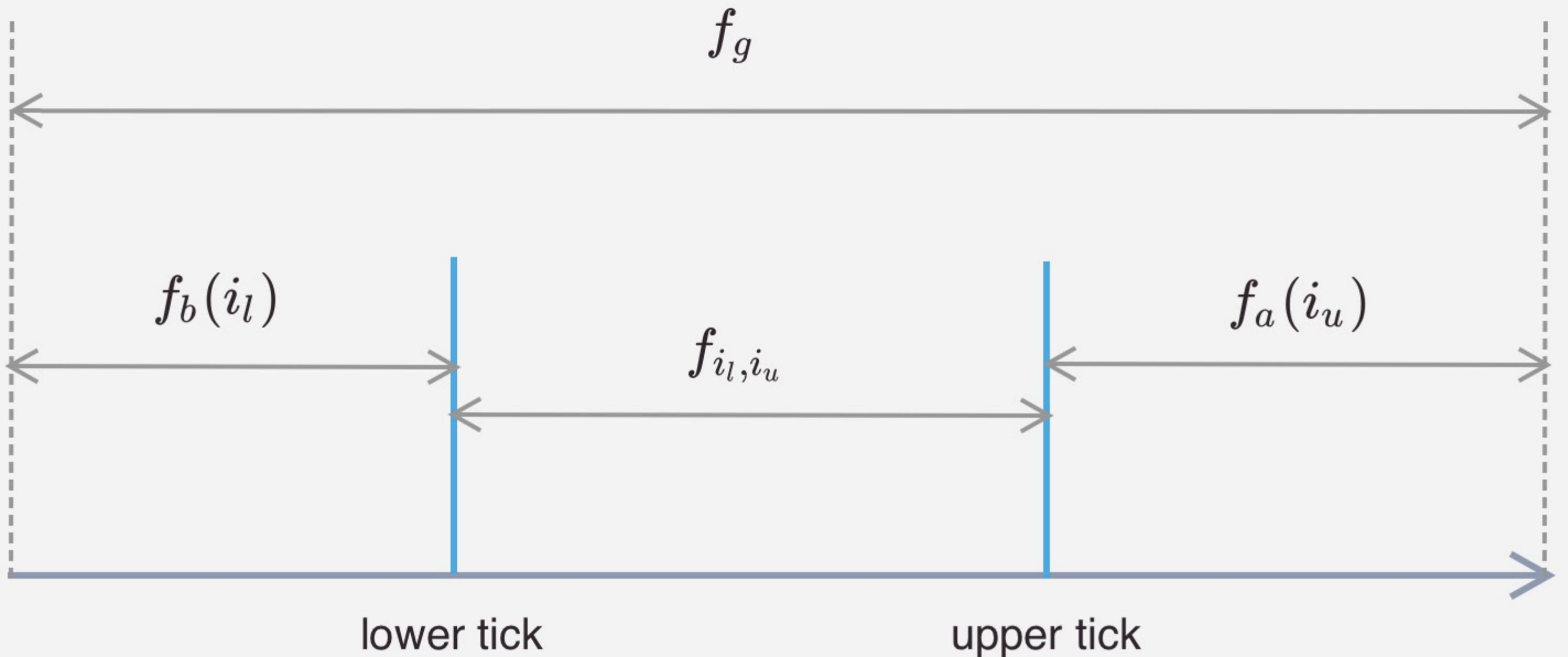
- 有多个区间时，如何获得某个区间的交易手续费（单位LP手续费累计值）。

$$f_{i_l, i_u} = f_g - f_b(i_l) - f_a(i_u)$$

feeGrowthOutside

在 (i_l, i_u) 区间发生的交易手续费累加，但是外侧的手续费是不变的，只需要累加 f_g

因此可以在tick上，加入
 $feeGrowthOutside(f_o)$ ，记录该 tick **外侧**
的手续费总量



Uniswap V3

- 初始化和更新外侧手续费： $feeGrowthOutside(f_o)$

在加入流动性时，进行初始化 (i_c : 表示当前价格 i : 表示 tick 边界) :

当 $i_c \geq i$ 时，则: $f_o = f_g$ ，假定所有手续费在该刻度之下生成

当 $i_c < i$ 时，则: $f_o = 0$

在交易时，若价格穿过某个 tick， f_o 需要翻转，因为价格外侧手续费永远要在当前价格的另一侧：

$$f_o = f_g - f_o$$

Uniswap V3

- 用户的手续费最终计算

则用户的手续费计算方式如下，先获得一个区间内手续费累加：

$$f_{i_l, i_u} = f_g - f_b(i_l) - f_a(i_u)$$

$$f_b(i) = \begin{cases} f_o(i) & i_c \geq i \\ f_g - f_o(i) & i_c < i \end{cases}$$

$$f_a(i) = \begin{cases} f_g - f_o(i) & i_c \geq i \\ f_o(i) & i_c < i \end{cases}$$

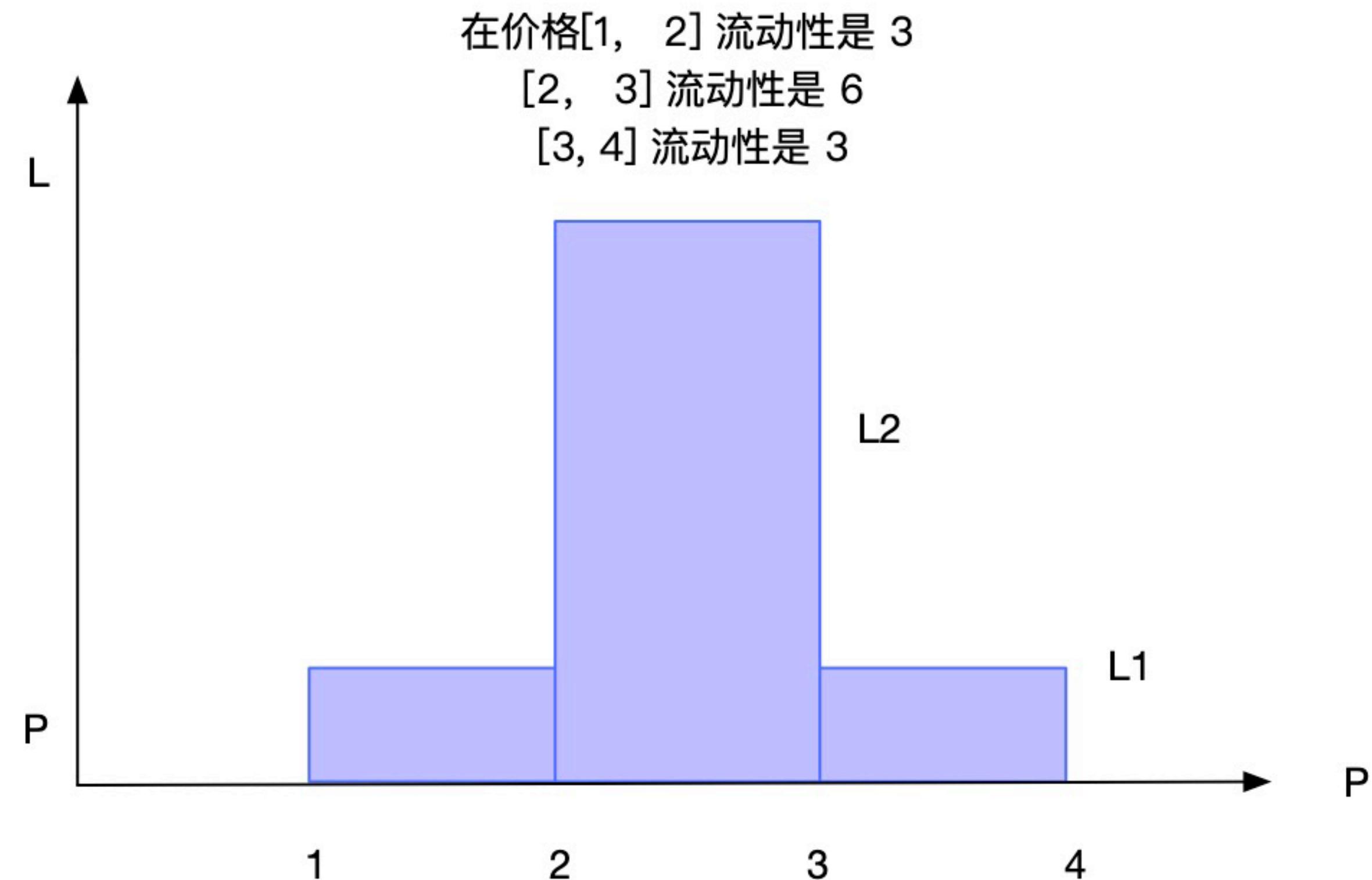
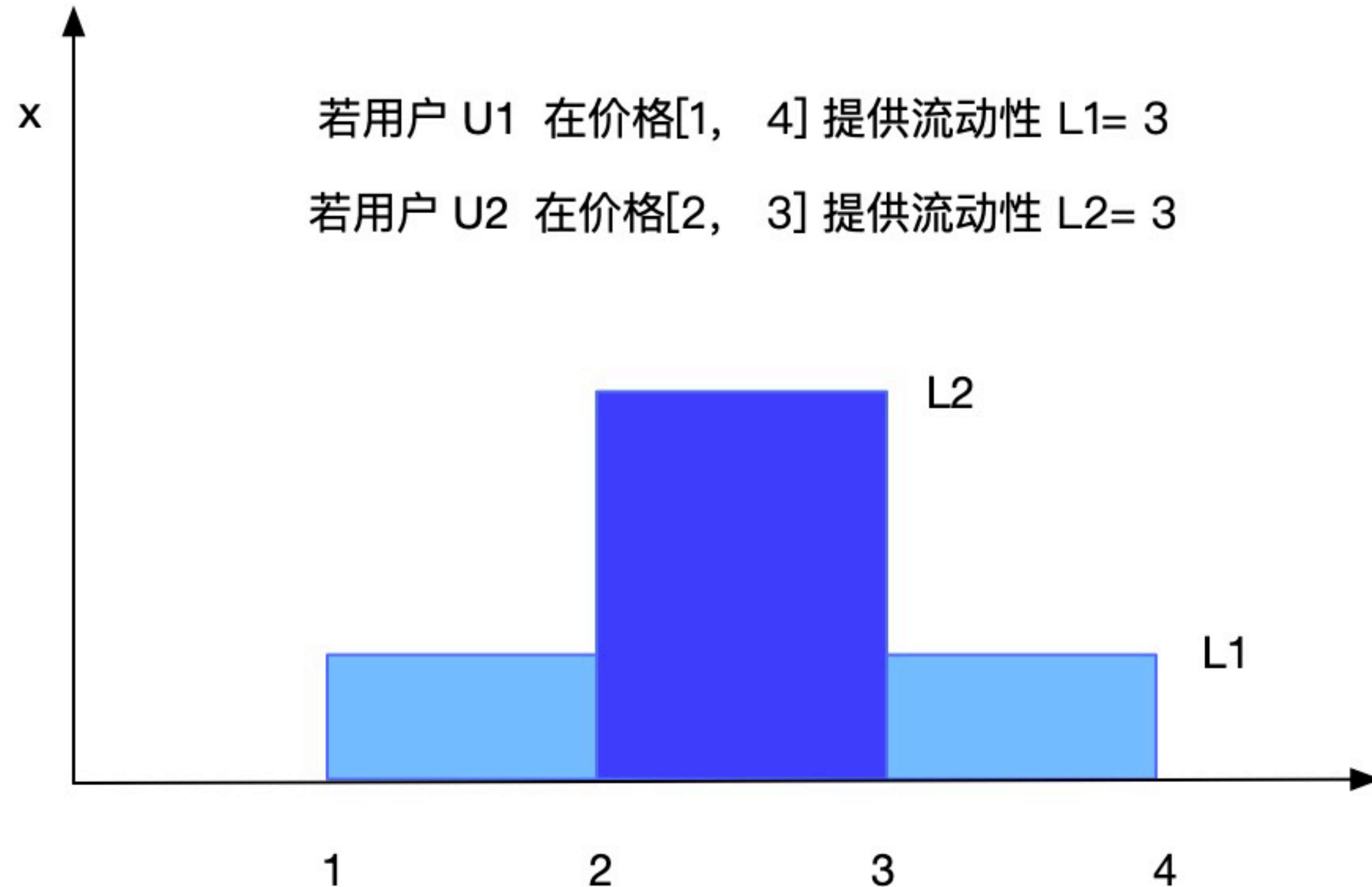
$f_b(i)$: 在 i 点以下(below)累计的手续费

$f_a(i)$: 在 i 点以上(above)累计的手续费

$$fee = f_{i_l, i_u} * \Delta L$$

手续费是单独保存的，需要自己去 collect

第6周



假设起始手续费为 0，交易发生在[2,3]，产生手续费为 6，因此在价格[2,3]：

- 单位手续费为 $f_g = 6/6 = 1$
- 外侧手续费 $f_o(1), f_o(2), f_o(3), f_o(4)$ 都为 0

此时，U1 的手续费为 3： $L_1 * (f_g - f_b(1) - f_a(4)) = L_1 * (f_g - f_o(1) - f_o(4)) = 3 * 1 = 3$

同样：U2 手续费也为 3

若价格由 3 变化到 4，则 $f_o(3) = f_g - f_o(3) = 1$ ，若交易发生在[3,4]，产生手续费为 6，此时：

单位手续费为 $f_g + = 6/3 = 3$ ，

用户 1 手续费： $L_1 * (f_g - f_o(1) - f_o(4)) = 3 * 3 = 9$

用户 2 手续费： $L_2 * (f_g - f_o(2) - f_o(3)) = L_2 * (f_g - f_o(2) - (f_g - f_o(3))) = 3 * (3 - 0 - (3 - 1)) = 3$

Uniswap V3

- 预言机

V2 合约中记录了 `price0CumulativeLast`, `price0CumulativeLast` , 但是需要外部更新保存, 然后计算算术平均价格

V3 使用 跟踪价格的 tick (价格的对数值 $\log_{1.0001}P$) 累计和: $a_t = \sum_{i=1}^t \log_{1.0001}P_i$

在使用时, 求 tick 算术平均值: $\frac{\sum_{i=t_1}^{t_2} \log_{1.0001}P_i}{t_2 - t_1}$ 可以转换为:

$$\frac{\sum_{i=t_1}^{t_2} \log_{1.0001}P_i}{t_2 - t_1} = \log_{1.0001} \left(\prod_{i=t_1}^{t_2} P_i \right)^{\frac{1}{t_2 - t_1}}$$

(几何平均值其实是其对数域上的算数平均值)

$\left(\prod_{i=t_1}^{t_2} P_i \right)^{\frac{1}{t_2 - t_1}}$ 即在 $[t_1, t_2]$ 时间段的几何平均值价格。

算术平均数 $A = \frac{A_1 + A_2 + \dots + A_n}{n}$

几何平均数 $A = \sqrt[n]{A_1 \times A_2 \times \dots \times A_n}$

Uniswap V3

- 预言机

V2 合约中记录了 `price0CumulativeLast`, `price0CumulativeLast` , 但是需要外部更新保存, 然后计算算术平均价格

V3 使用 跟踪价格的 tick (价格的对数值 $\log_{1.0001}P$) 累计和: $a_t = \sum_{i=1}^t \log_{1.0001}P_i$

在使用时, 求 tick 算术平均值: $\frac{\sum_{i=t_1}^{t_2} \log_{1.0001}P_i}{t_2 - t_1}$ 可以转换为:

$$\frac{\sum_{i=t_1}^{t_2} \log_{1.0001}P_i}{t_2 - t_1} = \log_{1.0001} \left(\prod_{i=t_1}^{t_2} P_i \right)^{\frac{1}{t_2 - t_1}}$$

(几何平均值其实是其对数域上的算数平均值)

$\left(\prod_{i=t_1}^{t_2} P_i \right)^{\frac{1}{t_2 - t_1}}$ 即在 $[t_1, t_2]$ 时间段的几何平均值价格。

算术平均数 $A = \frac{A_1 + A_2 + \dots + A_n}{n}$

几何平均数 $A = \sqrt[n]{A_1 \times A_2 \times \dots \times A_n}$

$\log_a(MN) = \log_a M + \log_a N$

$\log_a \sqrt[n]{M} = \frac{1}{n} \log_a M$

Uniswap V3

- 预言机

假设 P_{t_1, t_2} 为 $[t_1, t_2]$ 区间的交易对价格的几何平均值, a_1, a_2 分别为 t_1, t_2 时间点 tick index 的时间加权累积值, 那么 P_{t_1, t_2} 的计算为:

$$\log_{1.0001}(P_{t_1, t_2}) = \frac{\sum_{i=t_1}^{t_2} \log_{1.0001} P_i}{t_2 - t_1}$$

$$\log_{1.0001}(P_{t_1, t_2}) = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

$$P_{t_1, t_2} = 1.0001^{\frac{a_{t_2} - a_{t_1}}{t_2 - t_1}}$$

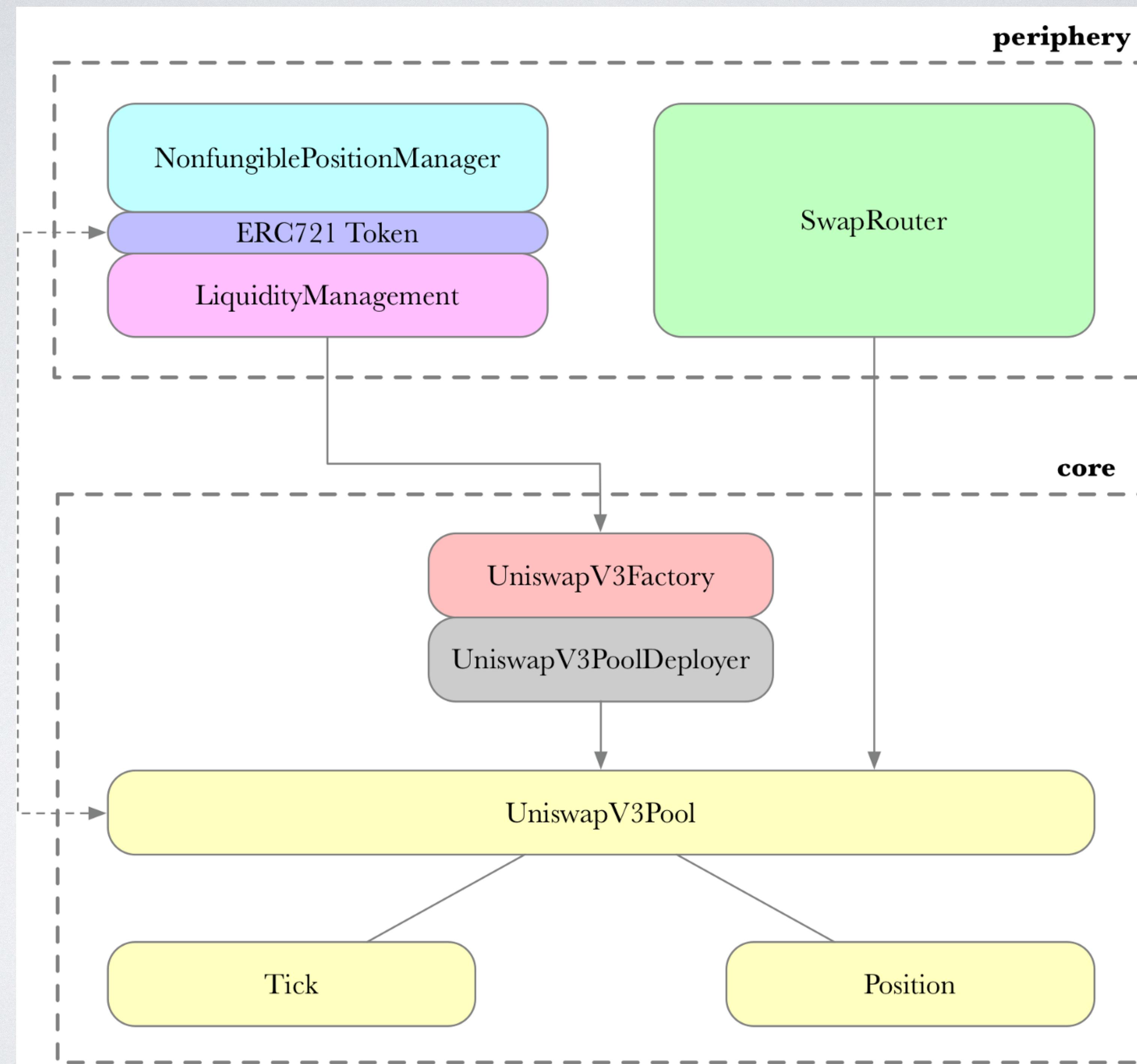
Uniswap V3

- 预言机

为节省交易 gas , 默认V3 只保留最近一个 tick 累积值, 由Oracle 有需求的开发者来扩展存储的历史数量。

```
struct Observation {  
    // 记录区块的时间戳  
    uint32 blockTimestamp;  
    // tick index 的时间加权累积值  
    int56 tickCumulative;  
    // 价格所在区间的流动性的时间加权累积值  
    uint160 liquidityCumulative;  
    // 是否已经被初始化  
    bool initialized;  
}
```

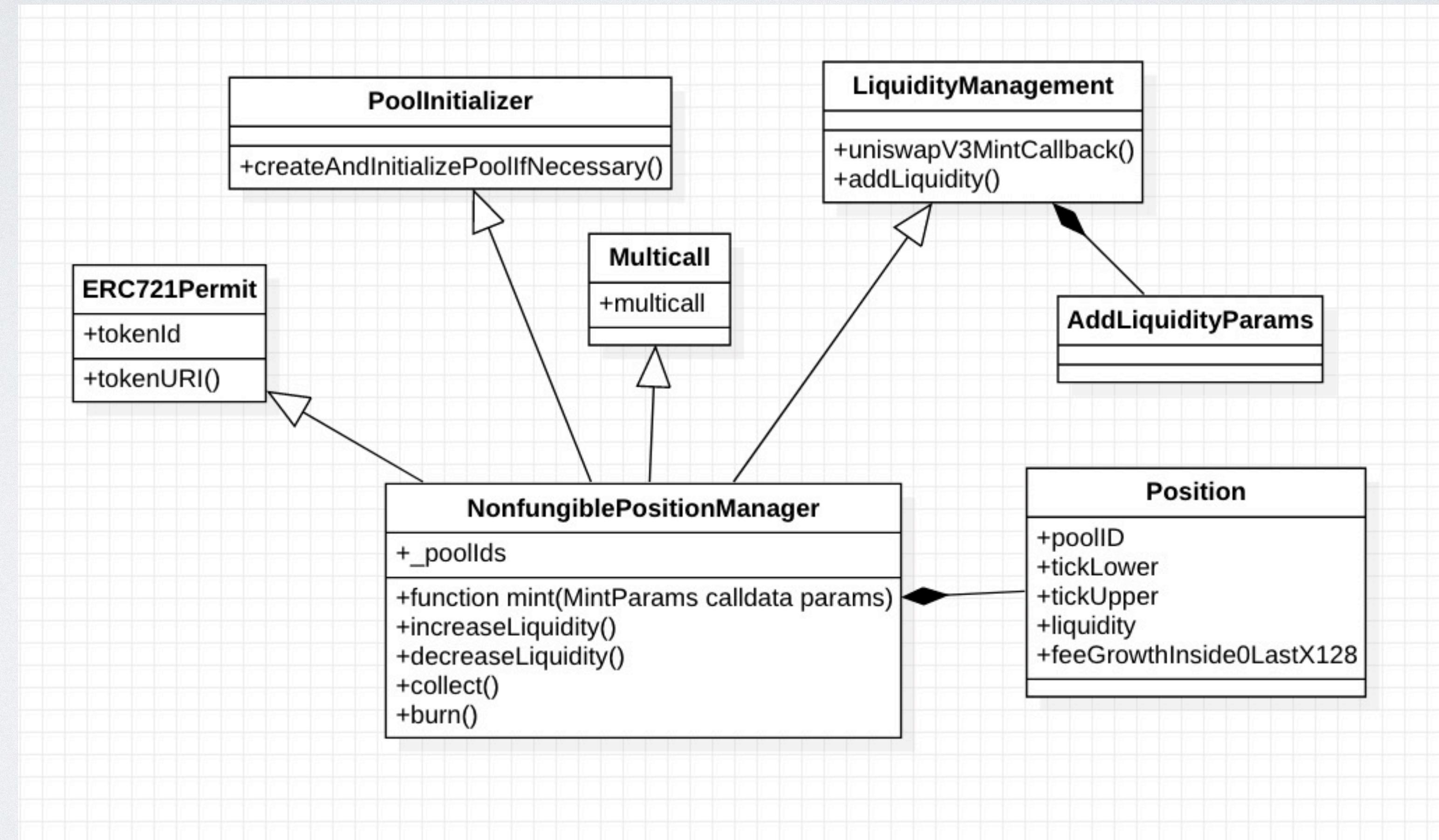
第6周



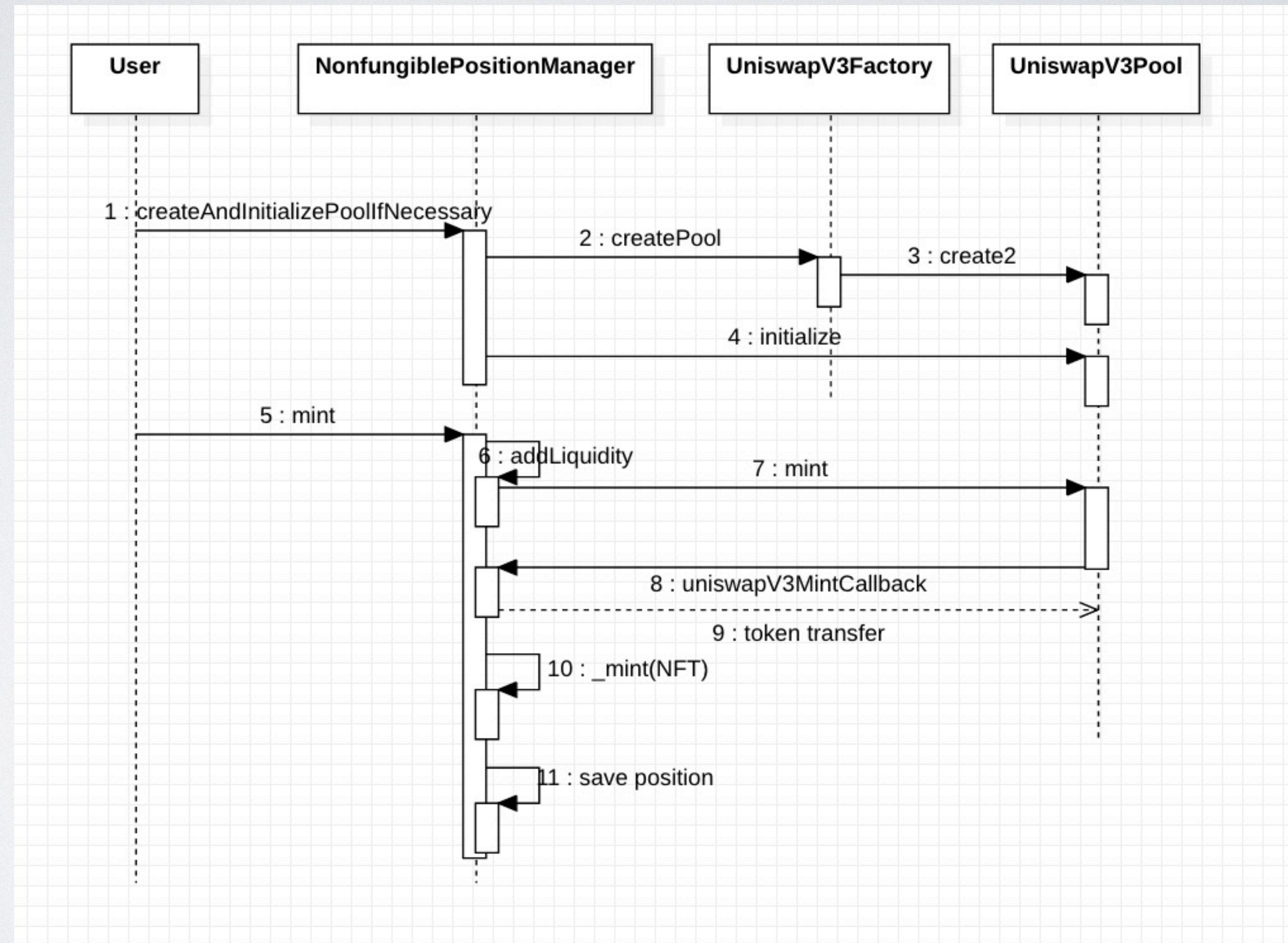
Uniswap V3

- NonfungiblePositionManager:
 - positions: 获取仓位信息
 - createAndInitializePoolIfNecessary: 创建 Pool 池
 - mint: 新增流动性 (发行NFT)
 - increaseLiquidity、decreaseLiquidity: 添加、减少流动性
 - collect: 收集手续费
 - burn: 删除流动性
- SwapRouter
 - exactInputSingle、exactOutputSingle: 单个交易对兑换
 - exactInput、exactOutput: 可设置 path, 对多个交易对兑换
 - uniswapV3SwapCallback: swap 时进行回调, 支付真正的 token

Uniswap V3

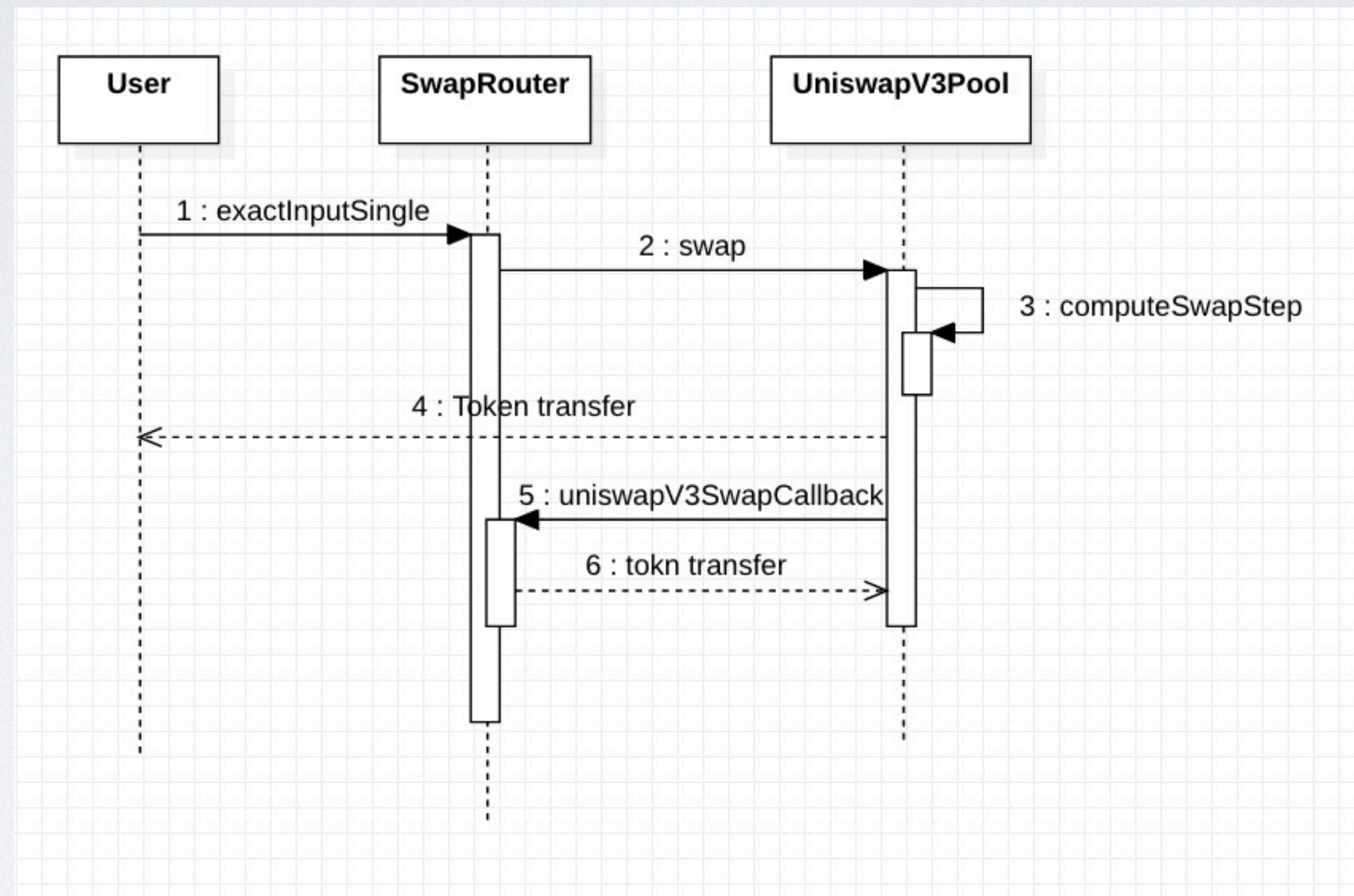


• 添加流动性



Uniswap V3

- 兑换



Uniswap V3 参考文章

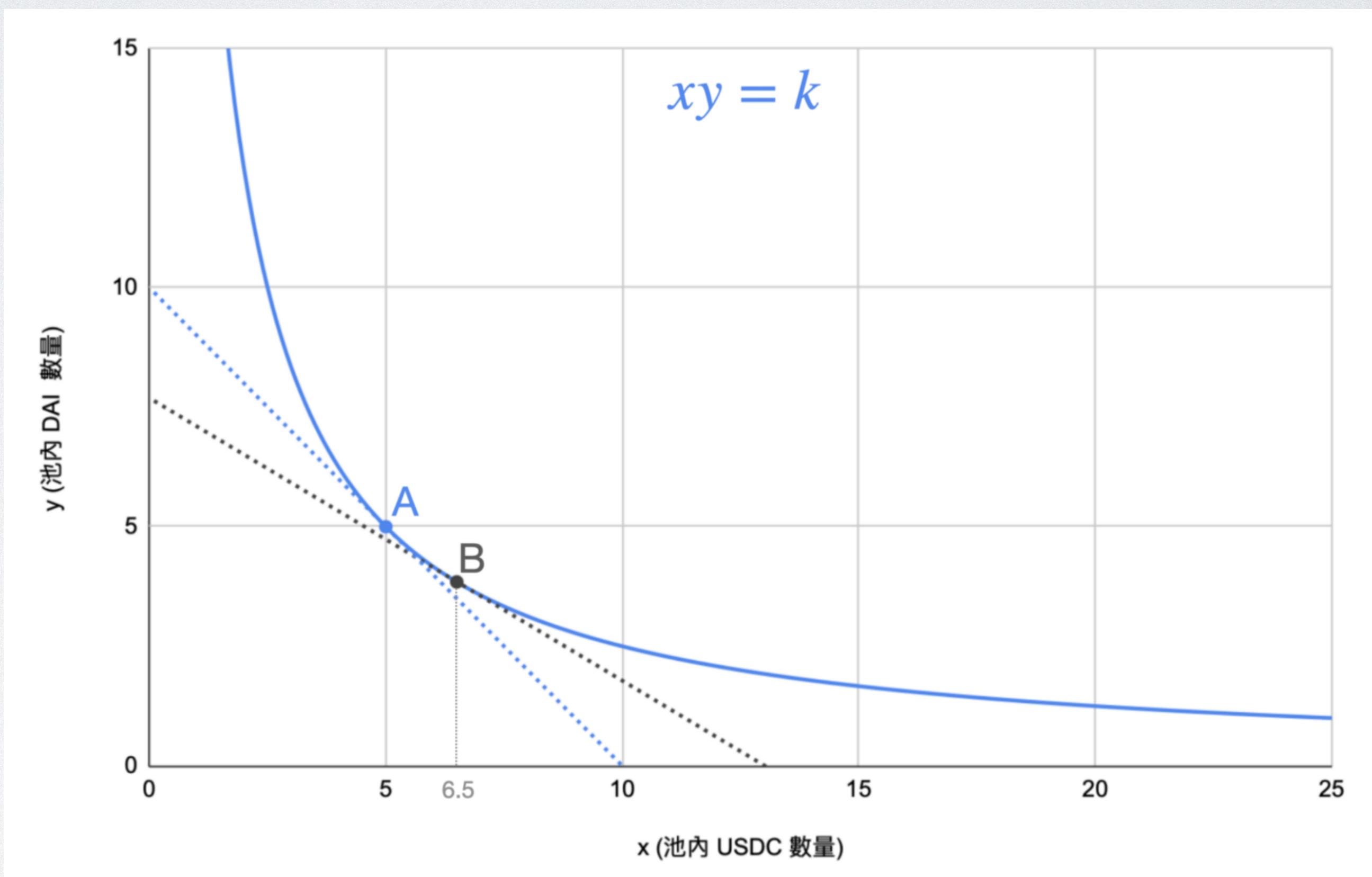
- 白皮书：<https://uniswap.org/whitepaper-v3.pdf>
- 中文版白皮书：<https://f00wcc7jp0.feishu.cn/file/boxcnKWqffRmWsIUkik6jlfqZ7b>
- Uniswap V3 文档：<https://docs.uniswap.org/protocol/reference/smart-contracts>
- 大佬的 Uniswap v3 分析：<https://liaoph.com/tags/#Uniswap>
- Uniswap v3 b 站：<https://www.bilibili.com/video/BV1go4y1X7Kx>
- 合约代码结构：<https://jlmmy.fi/uniswapv3/pool>
- 函数分析：<https://www.desmos.com/calculator/oiv0rti0ss?lang=zh-CN>

第6周

Curve 协议

- 稳定币兑换可以使用 $xy=k$?

要价格波动小（2%以内），
一次交易的数量需要小于池
内的 1%



参考文章：<https://medium.com/@cic.ethan/淺談穩定幣互換機制-從-balancer-到-curve-f638f29b33f9>

第6周

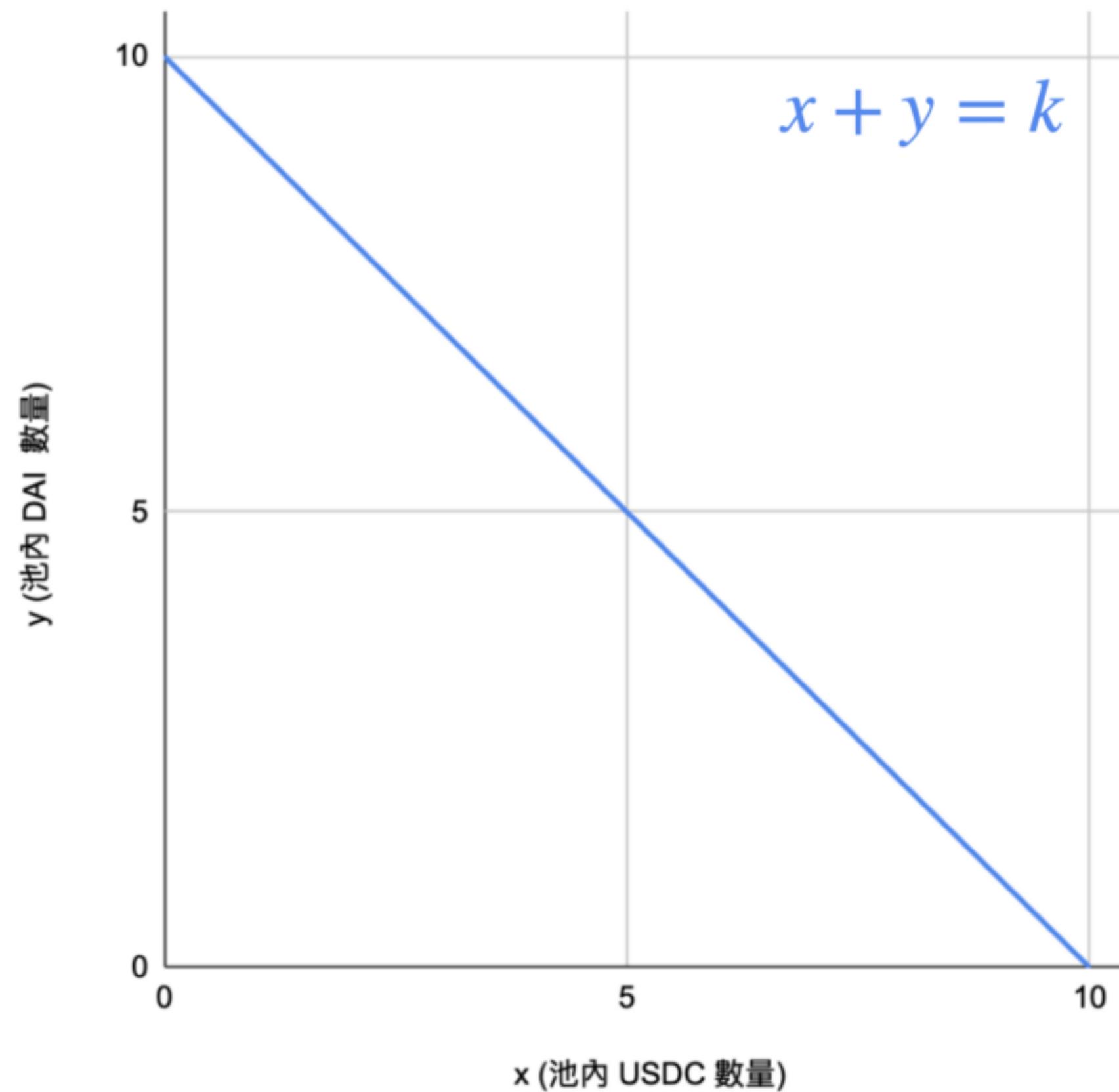
Curve 协议

- 常量和模型 $x + y = k$

优点：永远1:1兑换，没有滑点：

$$\Delta X = \Delta Y$$

缺点：价格不应供需改变，出现以外部价差时，出现流动性枯竭。



第6周

Curve 协议

- 稳定币兑换

恒定乘积：双曲线，两端将接近无穷

恒定总和：直线，0 滑点、流动性枯竭

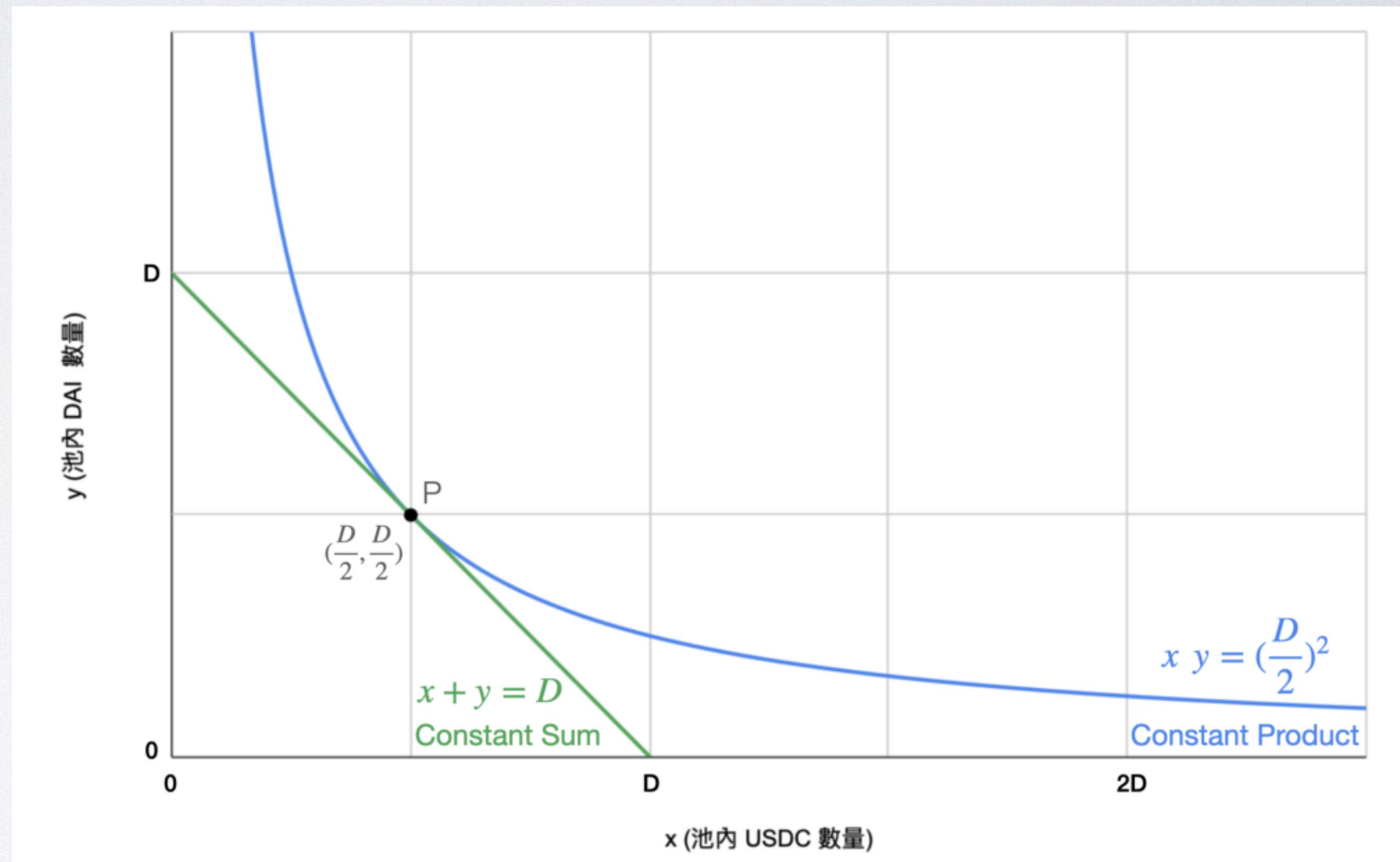
$$x \cdot y = \left(\frac{D}{2}\right)^2 \quad \times \beta$$

$$x + y = D \quad \times \alpha D$$

[数量] [数量]

理想模型于两者之间，简单方式加权平均：

$$\alpha D(x + y) + \beta(xy) = \alpha DD + \beta\left(\frac{D}{2}\right)^2$$



第6周

Curve 协议

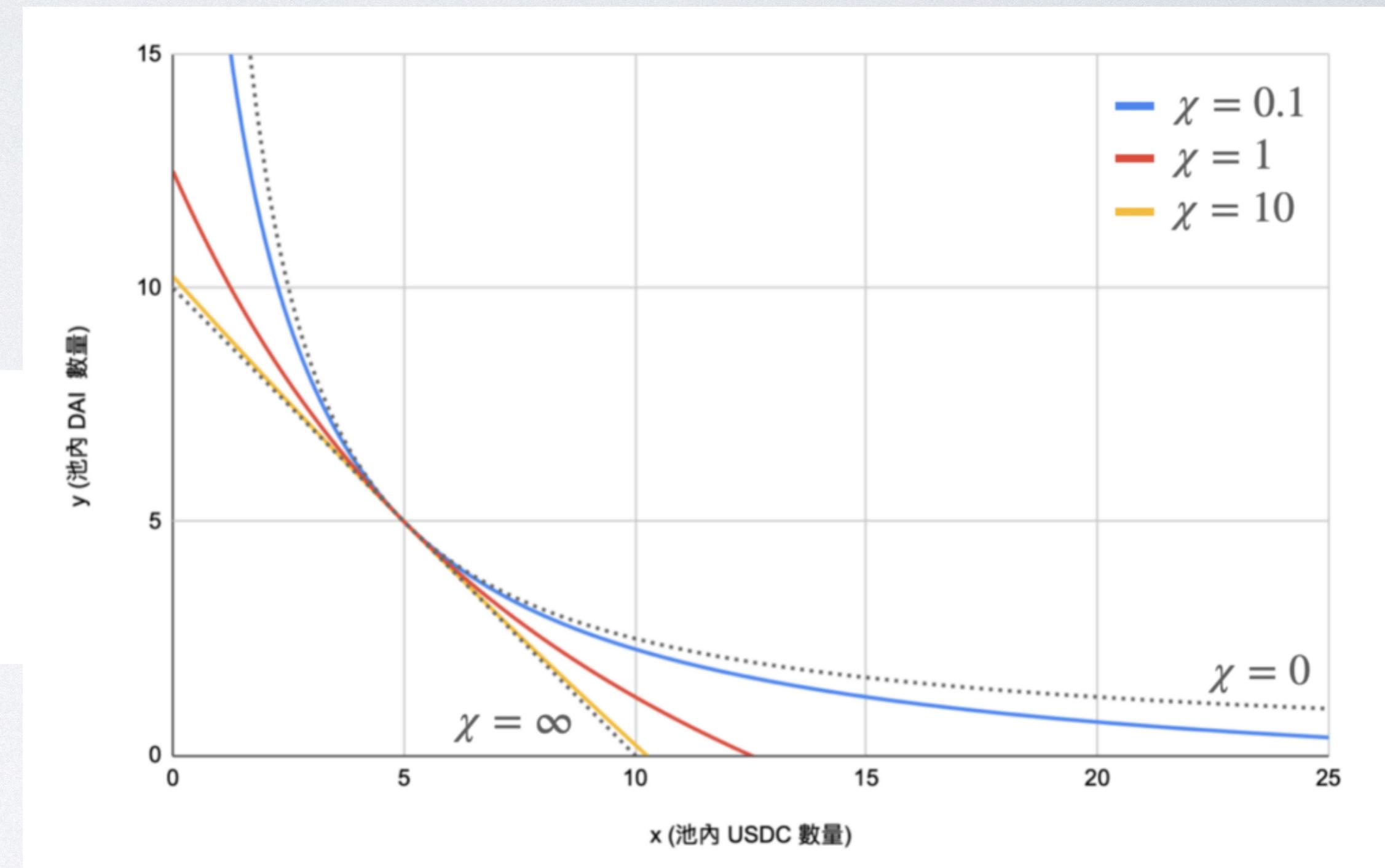
- 稳定币兑换

$$\alpha D(x + y) + \beta(xy) = \alpha DD + \beta\left(\frac{D}{2}\right)^2$$

$$\chi D(x + y) + (xy) = \chi D^2 + \left(\frac{D}{2}\right)^2 , \chi = \frac{\alpha}{\beta}$$

$\chi = 0$, 方程退化为常量乘积模型, 当 $\chi = \infty$, 方程退化为常量和模型

可以通过设置 χ 来设计一条曲线



Curve 协议

- 引入动态参数

模型仍有一个缺陷：若希望价格越稳定，则曲线越接近直线，此时会面临流动性枯竭问题。能否在靠近平衡点时， χ 越大，远离平衡点时越小。

$$\chi = A \frac{xy}{(\frac{D}{2})^2}$$

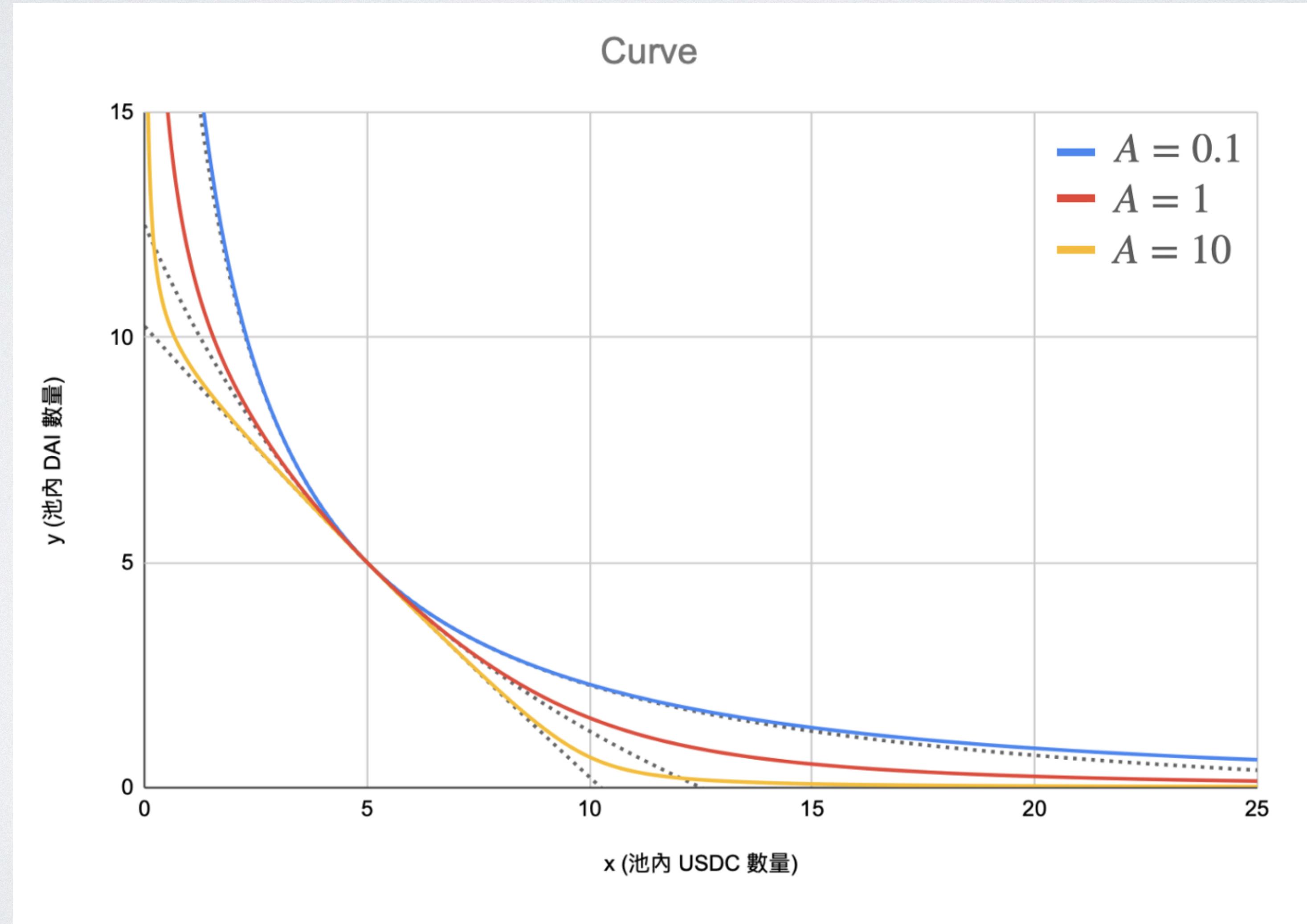
A 为放大系数，决定平衡点附件的曲率， A 越大滑点越小。蓝色部分为动态参数，在 $(0,1)$ 之间：
离平衡点越接近时，接近1，接近常量和模型。离平衡点越远时，接近0，接近常量乘积模型。

$$A \frac{xy}{(\frac{D}{2})^2} D(x + y) + (xy) = A \frac{xy}{(\frac{D}{2})^2} D^2 + (\frac{D}{2})^2$$

$$A2^2 (x + y) + D = AD2^2 + \frac{D^3}{2^2 xy}$$

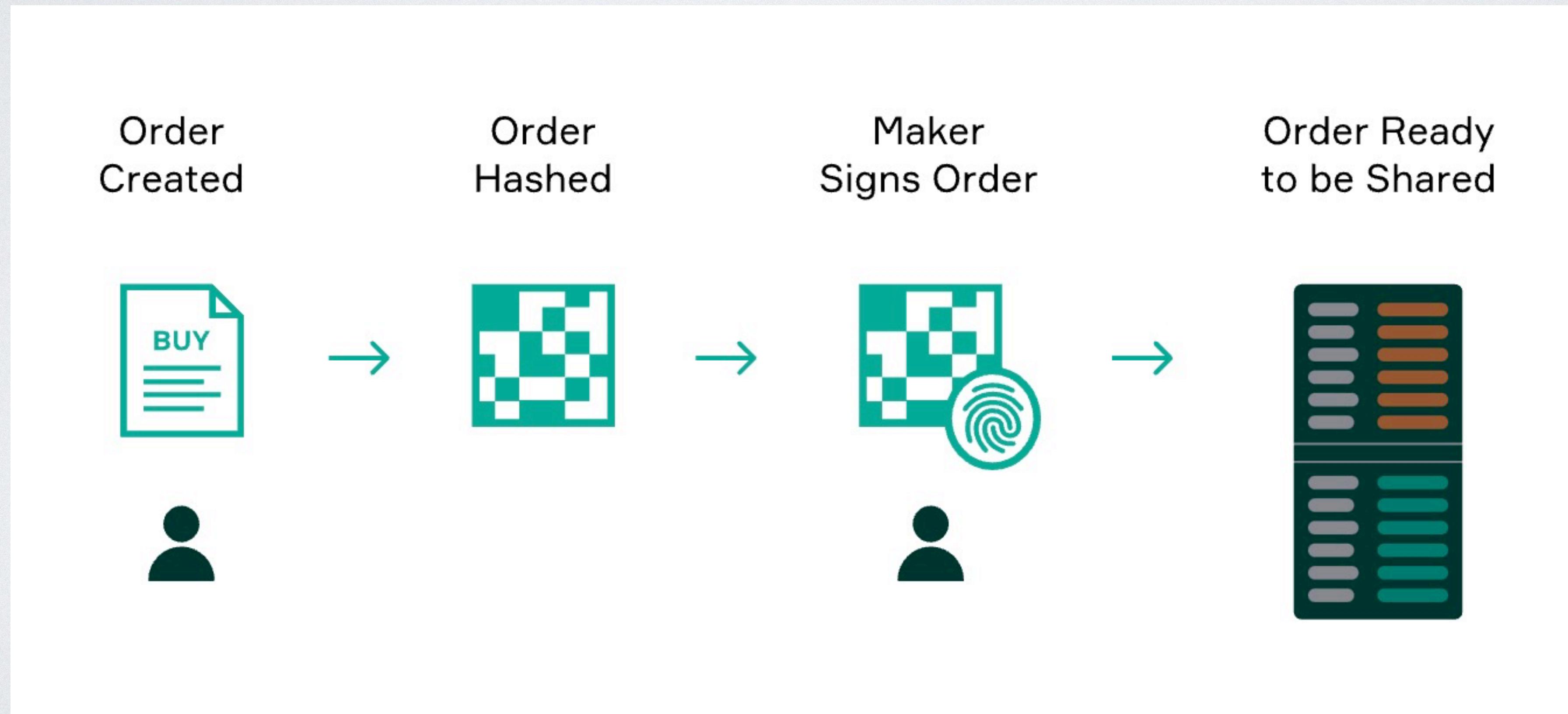
Curve 协议

- 稳定币兑换



0x 协议

- 链下撮合，链上结算



<https://0x.org/docs/core-concepts#digital-signature>

Q & A