minimum spanning tree

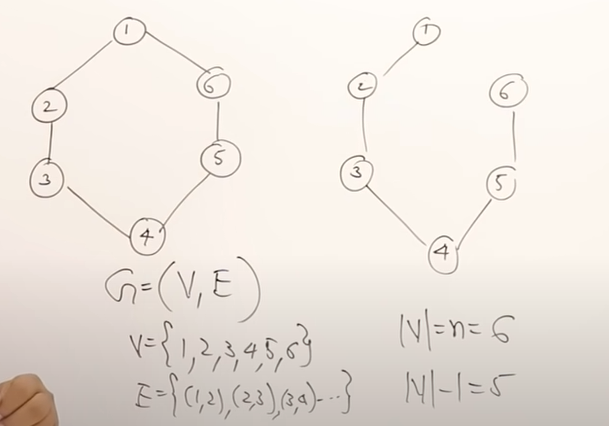
graph (G,E)

G 代表着端点集合，

E代表edge 集合

Spanning tree要求G相等，同时Tree没有cycle

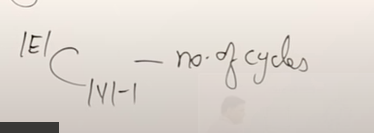
E是子集，也就是说用最小的edge数量经过所有点



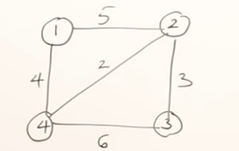
最少需要的Edge数量等于 V-1



有几种可能性来生成minimum tree：EC(N-1)-CYCLE

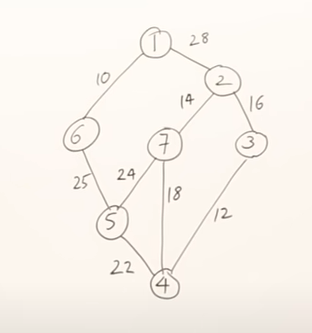


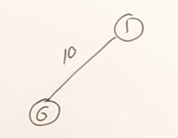
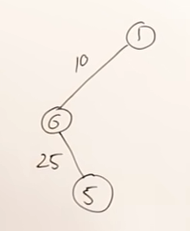
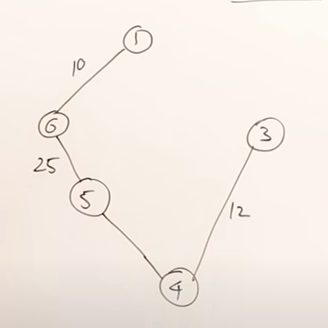
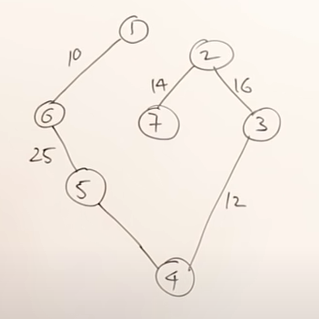
但是对于weighted graph呢

每条edge都有自己的cost，

Prim-Jarnick法则

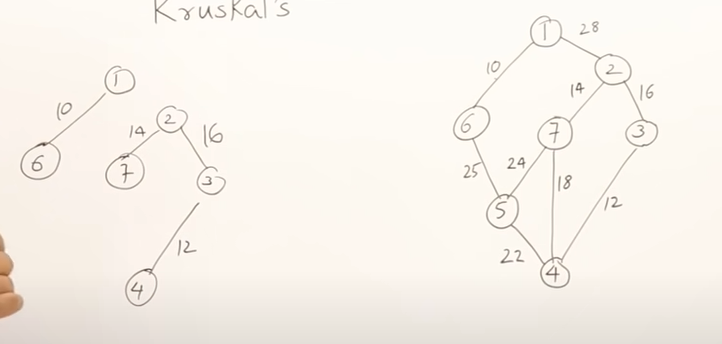
先选最短的一条边，然后选于这条边相邻的最短的边



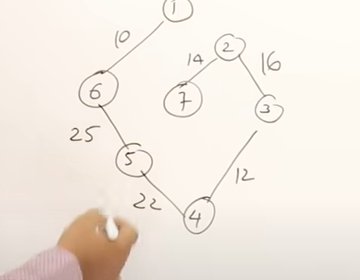
先选10 邻近的是25，28，选25，临近的是24,22,28，选22，在接下来临近的是12在接下来是16,14 

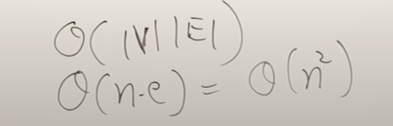
Kruskals方法

永远选择最短的边，如果这个边形成了cycle，舍去



先选10,12,14,16，都完美，接下来应该选18，但是形成cycle，舍去，选22

24形成cycle选25，完成

直接最短选取用时是

因为我们需要V-1条边，每次在E条边里找最短的

但我们可以用minheap,,

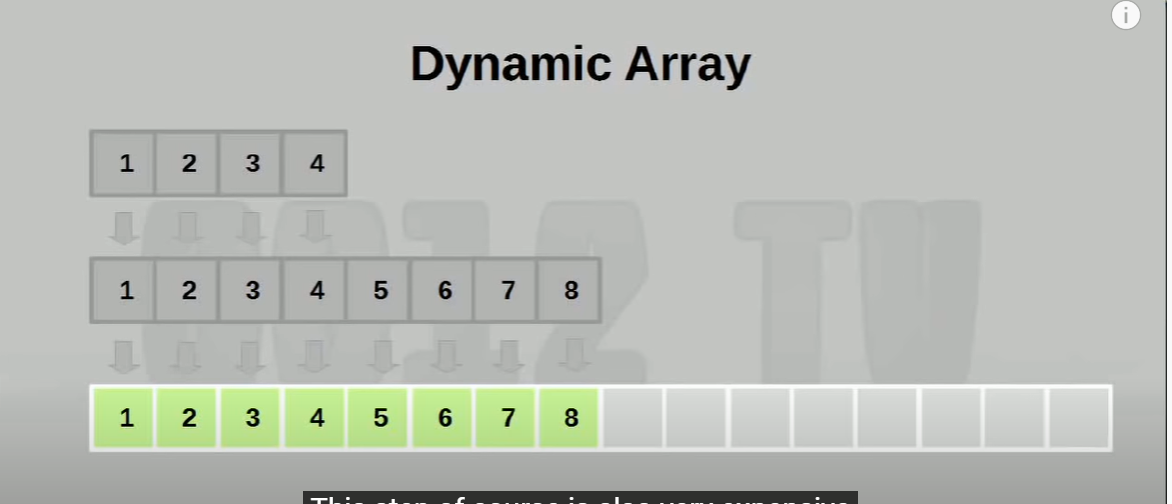
每次都会return最短的边，用时为logn

寻找n次，nlogn

Amortized analysis  
分析那种有多个operation的算法，对于有的算法来说，他每一个operation用时是不一样的，有的时间长有的时间短，Amortized analysis 希望求得每一个operation的平均用时

，哪怕部分operation实际很大

和average case analysis的不同在于并不考虑probability,每个operation都将发生

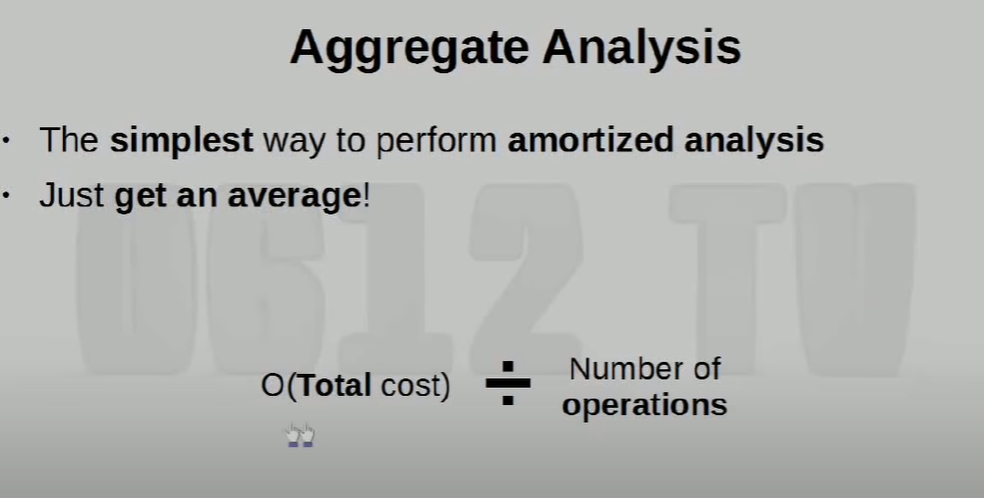


1234填进array，如果满了，翻倍，一个一个复制进去，再填，满了，翻倍，复制…

一个一个填进去很简单，就是O（1）,总共填N个就是O（n）

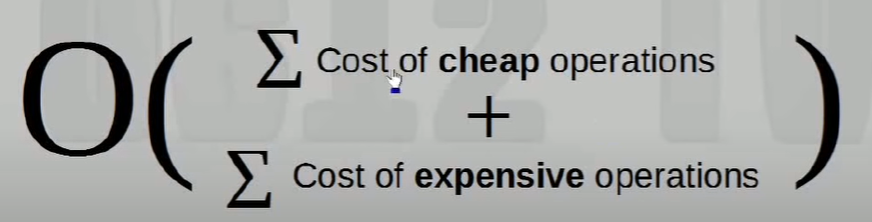
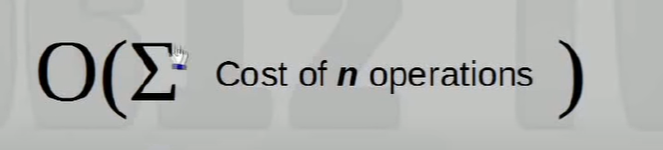
但是复制并不直观，

1.Aggregate analysis



最简单的进行amortized analysis的方法，就是平均

而Total实际上就是复杂的加上简单的



复杂操作发生点



我们可以看作初始容量是1，那么就是124816

每次相加

1

3

7

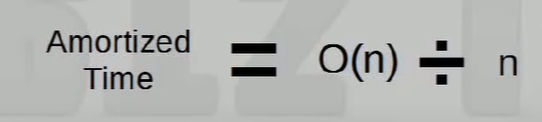
15

31，

每次都是对应Number翻倍减1

所以就是2n就是O（n）

而简单的一共复制n此减掉这些复杂的，也是O（n）



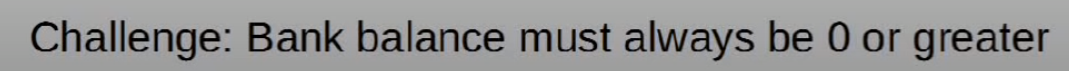
Accounting method

我们给每一个进入前的operation赋一个初始值

代表着收入（这个初始值是相当于平均的，与实际值不同）

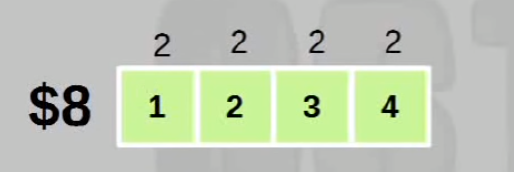
而其他操作每次付1，

我们需要足够的收入以免破产

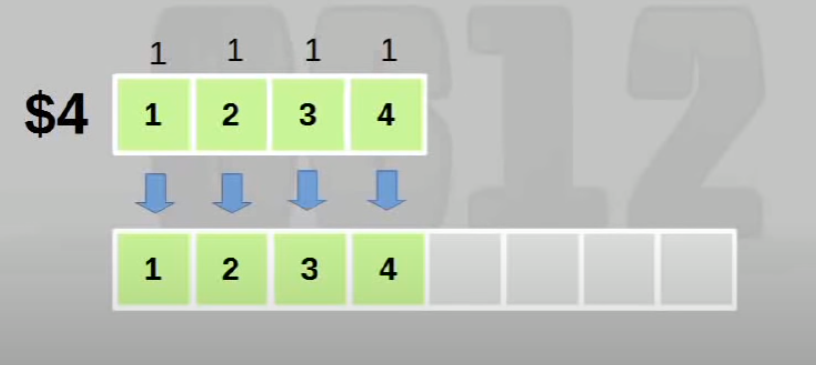


Dynamic array:给每个进入operation算3块钱收入

进入的一瞬间需要花费一块钱

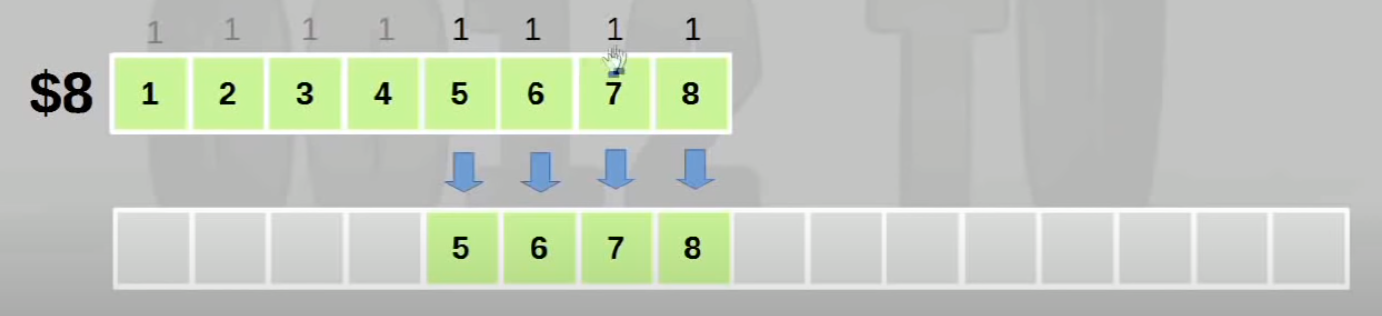


扩展array



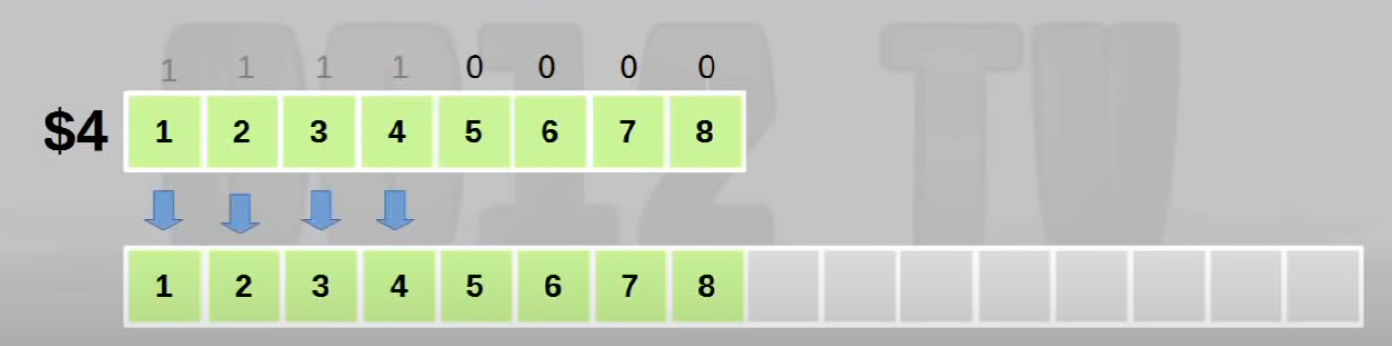
复制到大array里，扩充的时间可以不用算，因为和单个元素无关

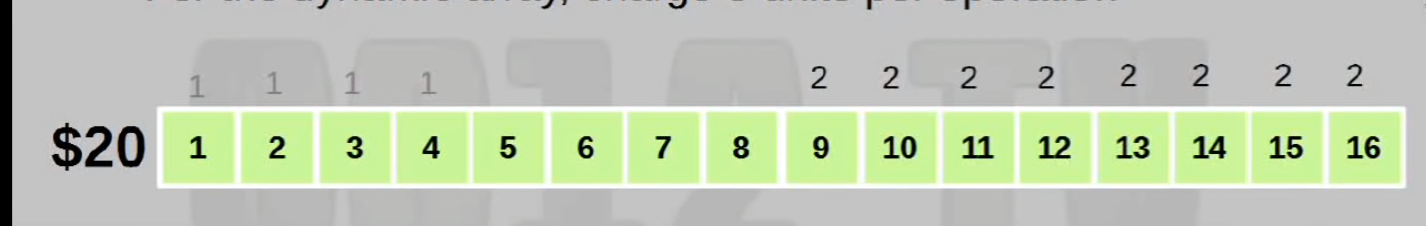


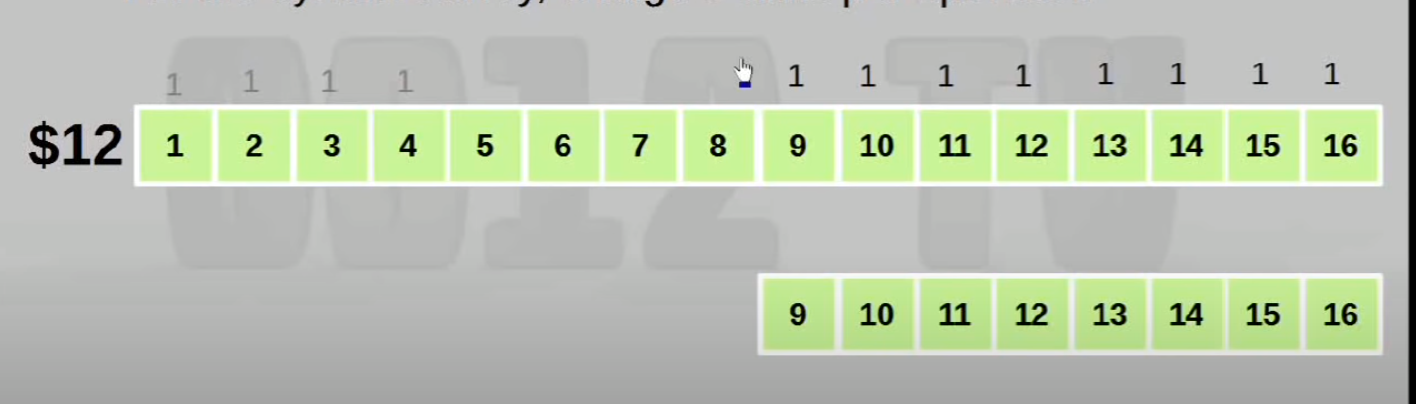


我们发现5678还剩1dollar，

都是我们的钱，我们可以用这个钱来移动1234



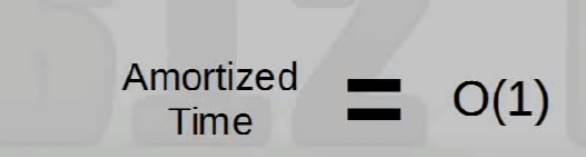




永远够付前面一半

也就是说每个operation实际花费是3

3N=O(n)

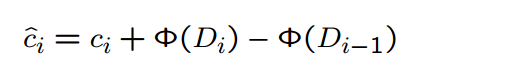


Potential method

和accounting method差不多，但是accounting追求的是从一个基础上一步一步推 ，发现其中规律

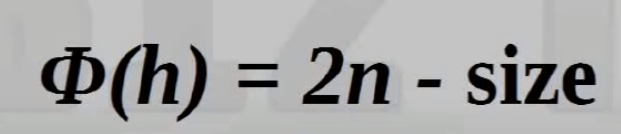
而这个用一个function,直接计算ith操作时Di的势能，

而一个操作的amortized cost等于当前势能减前一个操作的势能



ci等于这个操作本身actual cost

例如Dynamic array

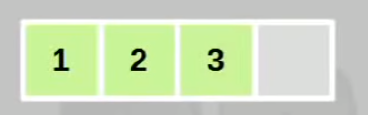


n就是第几个操作，size是总array大小

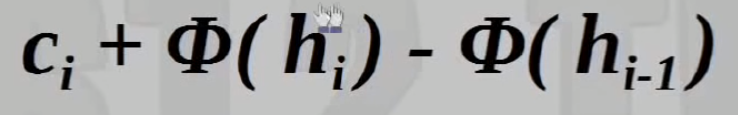
要翻倍的类型

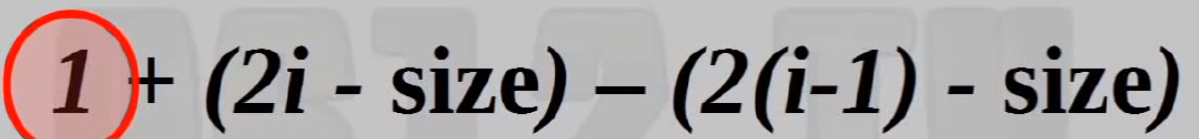


**常规型**

常规型一个operation的用量

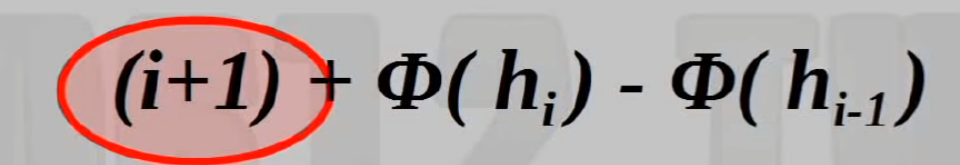


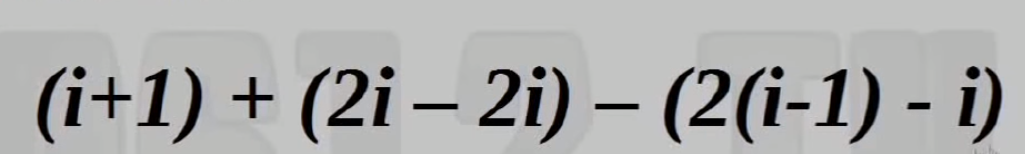


因为是normal case,这步操作只是单纯的1，插入

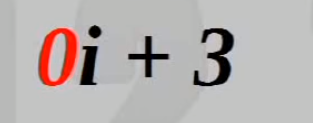
最终等于3

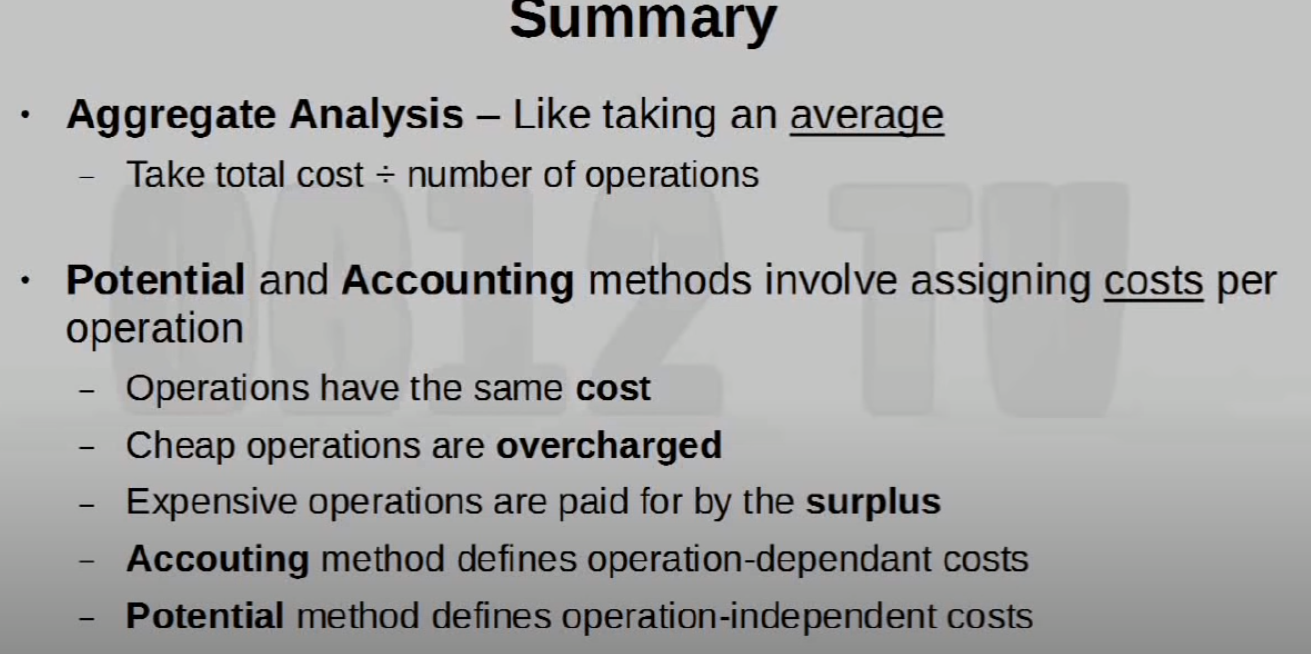
翻倍型一个operation insertion需要1，移动所有的需要i





要注意size改了·，我们现在的size是已经翻过倍的

最终等于3



aggregate analysis取平均值

Potential and accounting 给每个操作附一个cost