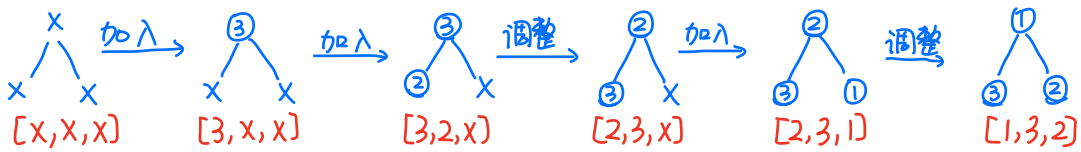
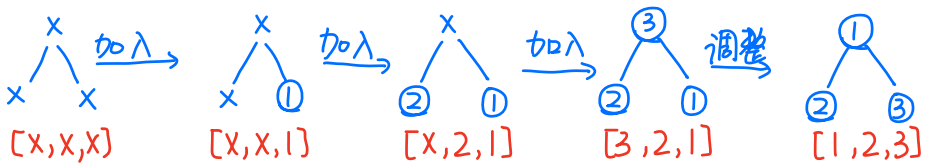
基础知识

nums = [3,2,1] # 原始数据

基于 sift up 的 heapify: O(nlogn)



基于 sift down 的 heapify: O(n)




```
>> from heapq import *
>> h = []
>> for x in [(1,'J'),(4,'N'),(3,'H'),(1,'A')]:
       heappush(h, item) # O(logk)
>> h = [(1,'J'),(4,'N'),(3,'H'),(1,'A')]
>> heapify(h) # heapify data inplace O(k)
>> while h:
       print(heappop(h)) #O(logk)
```

```
from random import randint
x = randint(LB,UB)
```

```
from functools import cmp_to_key
cmp = lambda x,y: -1 if str(x)+str(y) < str(y)+str(x) else 1
nums.sort(key=cmp_to_key(cmp))
```

isinstance(x, list)

(1) find first position of a num that's ≥ target ; return len(nums) if not found
```
idx = bisect_left(nums, target)
```
(2) find first position of a num that's > target ; return len(nums) if not found
```
idx = bisect_right(nums, target)
```
(3) 找 ≤ target 的 last position
```
idx = bisect_left(nums, target+1) -1
idx = bisect_right(nums, target) -1
```
(4) 找 ≥ target 的 first position
```
idx = bisect_left(nums, target)
idx = bisect_right(nums, target-1)
```

```
def dec(func):
    def inner():
        s = time.time()
        func()
        print(time.time()-s)
    return inner
```
↖ # datetime.now()

```
@dec
def func1():
    print('func1')
```
⟺
```
def func1():
    print('func1')
func1 = dec(func1)
```

* decorator 模式

Python: -3%4 → 1 -3//4 → -1 -(3//4) → 0
Java: -3%4 → -3 -3//4 → 0

a±b %mod = (a%mod ± b%mod)%mod
a*b%mod = a%mod * b%mod

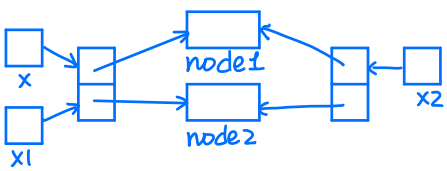
```
a=[1,2,3,1]
b=[1,3,1,2]
collections.Counter(a)==collections.Counter(b)
a+b # a[x]+b[x]
a-b # a[x]-b[x] (keeping only positive counts)
a&b # min(a[x],b[x])
a|b # max(a[x],b[x])
```

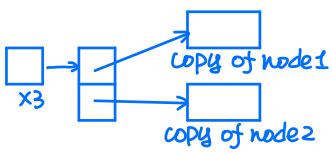
```
math.gcd      def gcd(a,b):
math.log          if b==0:
math.sqrt             return a
math.factorial    return gcd(b,a%b)
```

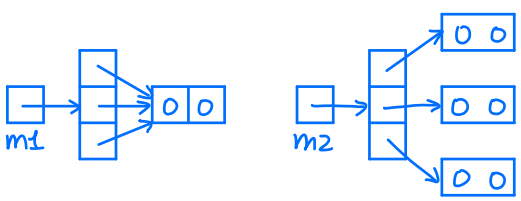
*singleton模式

```
def singleton(aClass):
    instance = None
    def onClass(*arg, **kwargs):
        nonlocal instance
        if instance == None:
            instance = aClass(*args, **kwargs)
        return instance
    return onCall

@singleton
class Spam:
    pass
```








```
m1=[[0]*2]*3 # shallow copy (wrong)
m2=[[0]*2 for _ in range(3)] #correct   or  m2=[[0 for _ in range(2)] for _ in range(3)]
```

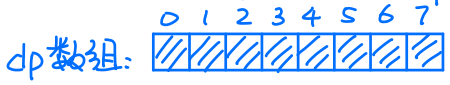
```
L=[1,2]
L.append(L)
```

如果数组长度为n，窗口大小为k
⇒总共有 n-(k-1)=n-k+1 个窗口，最后一个窗口的坐标为 n-k


```
[*num,1]  is equivalent to nums+[1]
sortedIdx=[x for (x,y) in sorted(enumerate(nums), key=lambda x:x[1])]
sorted(res, key=lambda x:x[0], reverse=True)
Node.__lt__ = lambda x,y: (x.val, x.index) < (y.val, y.index)
```

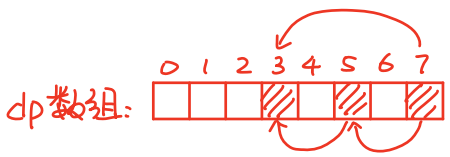
$-2^{31} \sim 2^{31}-1$
-2147483648~2147483647

Iterative DP (for loop)



为了计算 dp[7]，我们必须要先得到dp[0] ~ dp[6] ⇒ 不能剪枝

Recursive DP (DFS with memoization)



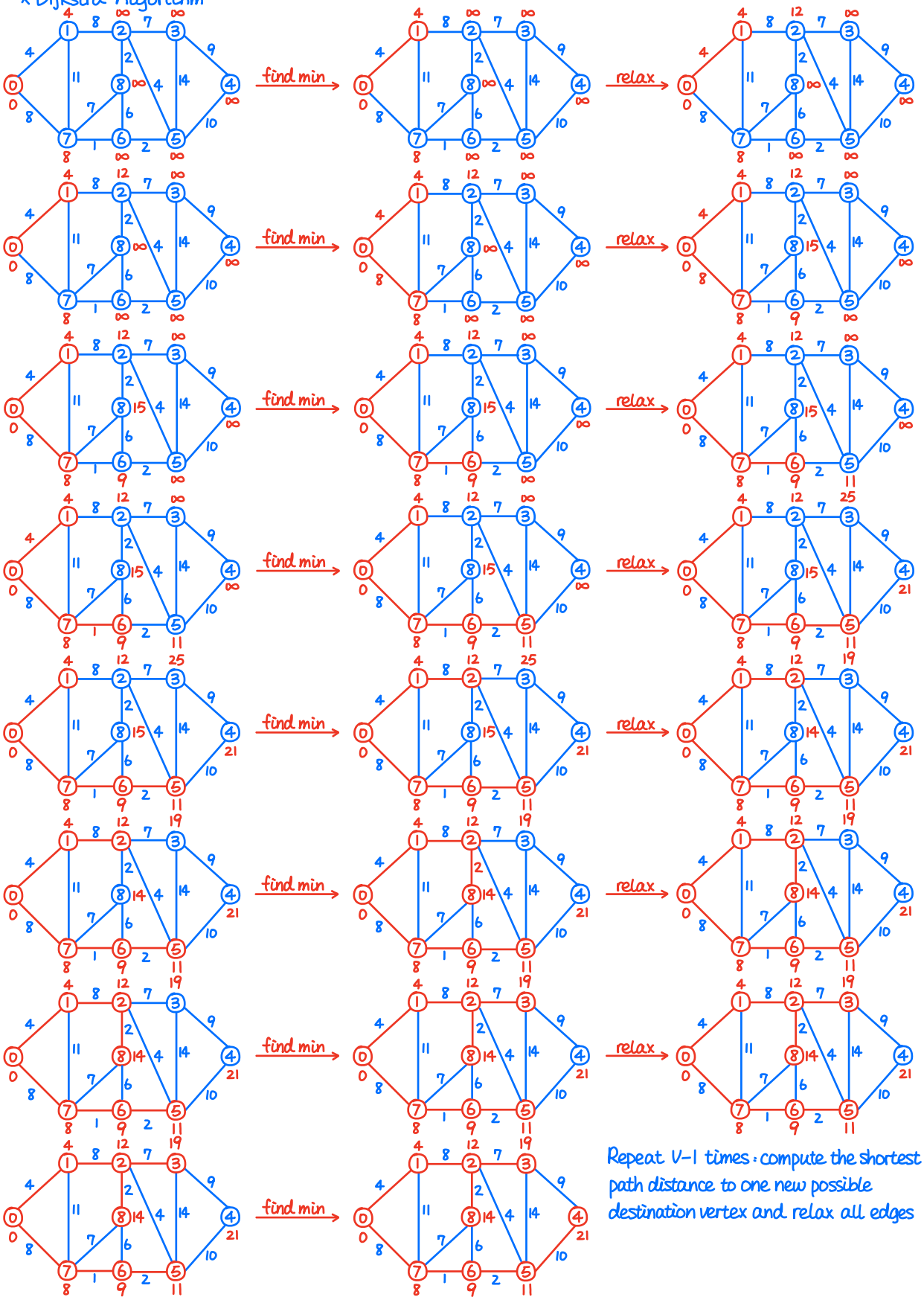
为了计算 dp[7]，我们可以只动态计算此它所依赖的先前状态 ⇒ 可以剪枝

(图里的箭头表示依赖关系：DP[7]依赖于DP[3]和DP[5]；DP[5]依赖于DP[3])

底层数据结构	Python	Java	C++
Dynamic Array	list	ArrayList	vector
Deque (array-based)	N/A	ArrayDeque	deque (比较少链表)
Deque (linked-list-based)	deque	LinkedList	list
Hash Table (key-value)	dict	HashMap	unordered_map
Hash Table (key-only)	set	HashSet	unordered_set
Balanced BST (key-value)	N/A	TreeMap	map
Balanced BST (key-only)	N/A	TreeSet	set
Heap	heapq	PriorityQueue	priority_queue
LinkedList + Hash table	OrderedDict	LinkedHashMap	N/A

Compute the shortest path distances from the source to "every" other vertex

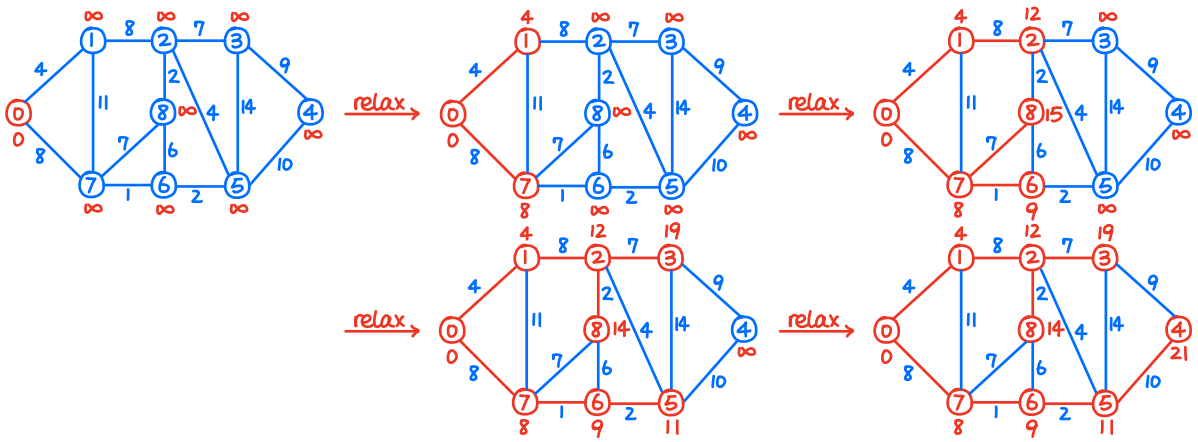
* Dijkstra Algorithm



Relaxation: if $dist(s \to u) + cost(u \to v) < dist(s \to v)$, update $dist(s \to v) = dist(s \to u) + cost(u \to v)$

With binary heap
$T(n) = O((E+V)\log V)$, $S(n) = O(V)$
↓ for connected graph
$T(n) = O(E \log V)$, $S(n) = O(V)$

* Bellman-Ford Algorithm: (restrict the number of vertexes to use ⇒ subproblems)
Either computes shortest paths from the source to all destinations, or, detects the negative cycle
Computing the shortest paths in the presence of a negative cycle is intractable (NP-hard)



1) for $k = 1, \dots, V-1$: (or stop early)
compute the shortest distances with $\le k$ edges
2) Run one extra iteration: that's reachable from s
if we see any improvement ⇒ negative cycle detected
(To reconstruct the negative cycle, simply remember the predecessors and run dfs to catch any repition)

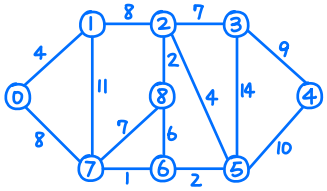
$T(n) = O(VE)$, $S(n) = O(V^2)$
↓ 滾動數組
$S(n) = O(V)$

Note: the longest possible path without a cycle can have $V-1$ edges

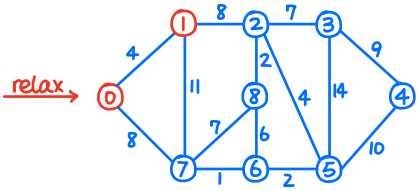
Undirected graph: $\sum degree(V) = 2|V|$
Directed graph: $\sum in\text{-}deg(V) = |V|$

* All-Pairs Shortest Paths (APSP) Floyd-Warshall Algorithm (restrict which vertexes to use ⇒ subproblems)

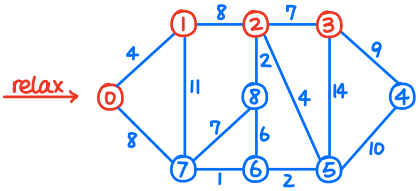
↳ some prefix of the vertexes



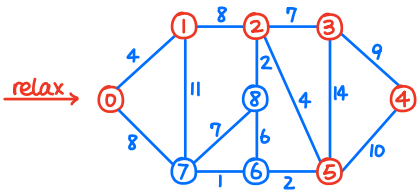
relax →



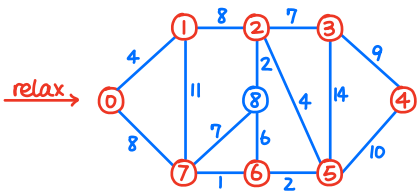
relax →

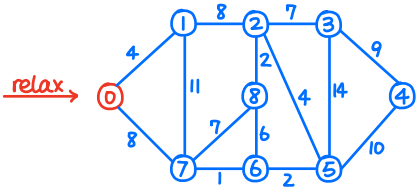


relax →

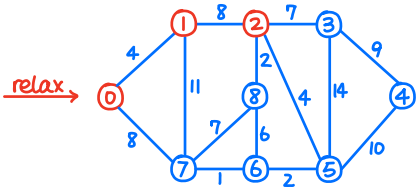


relax →

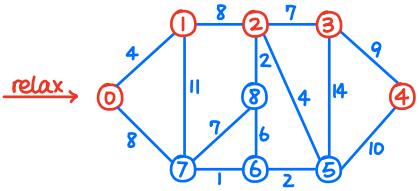




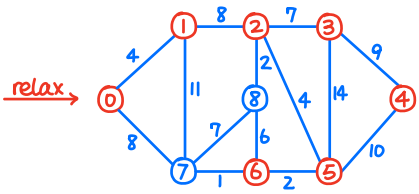
relax →



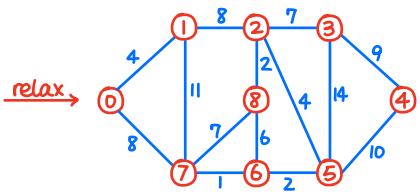
relax →



relax →



relax →



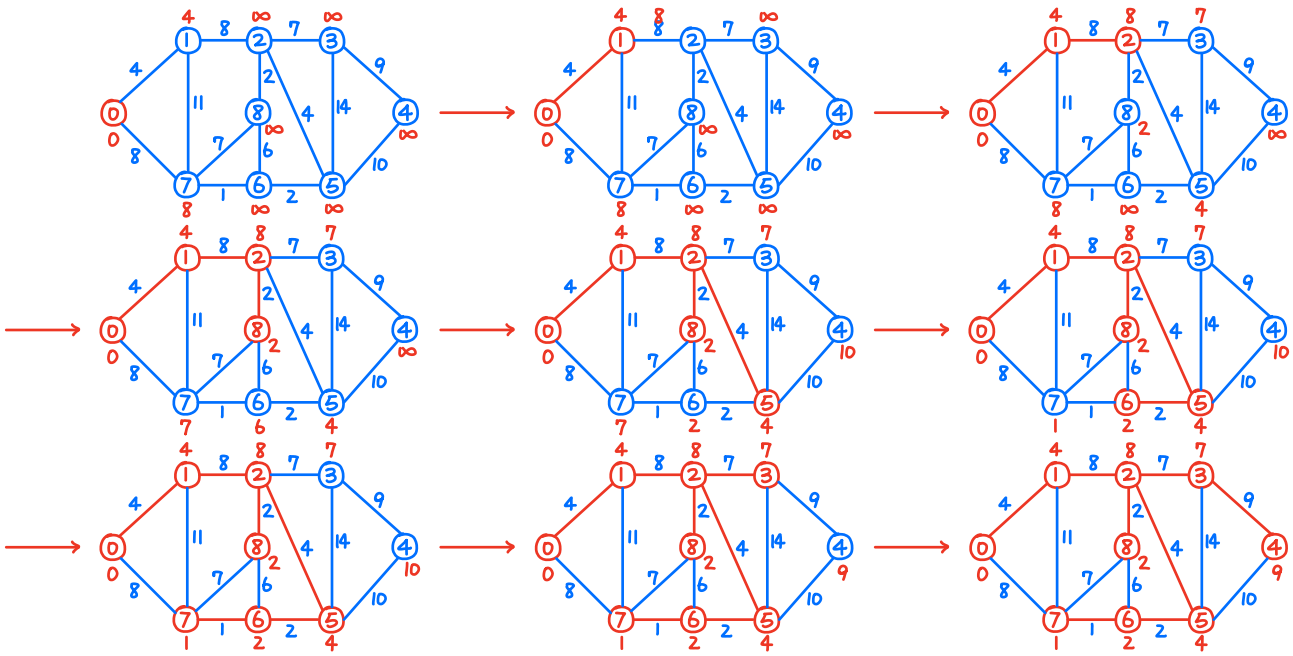
1) for $k = 1, \dots, V$:
   compute the shortest path with vertexes $1, \dots, k$ as internal nodes for all source-destination pairs (i, j)
2) if we see any negative diagonal element
   ⇒ negative cycle detected

$T(n) = O(V^3)$, $S(n) = O(V^3)$

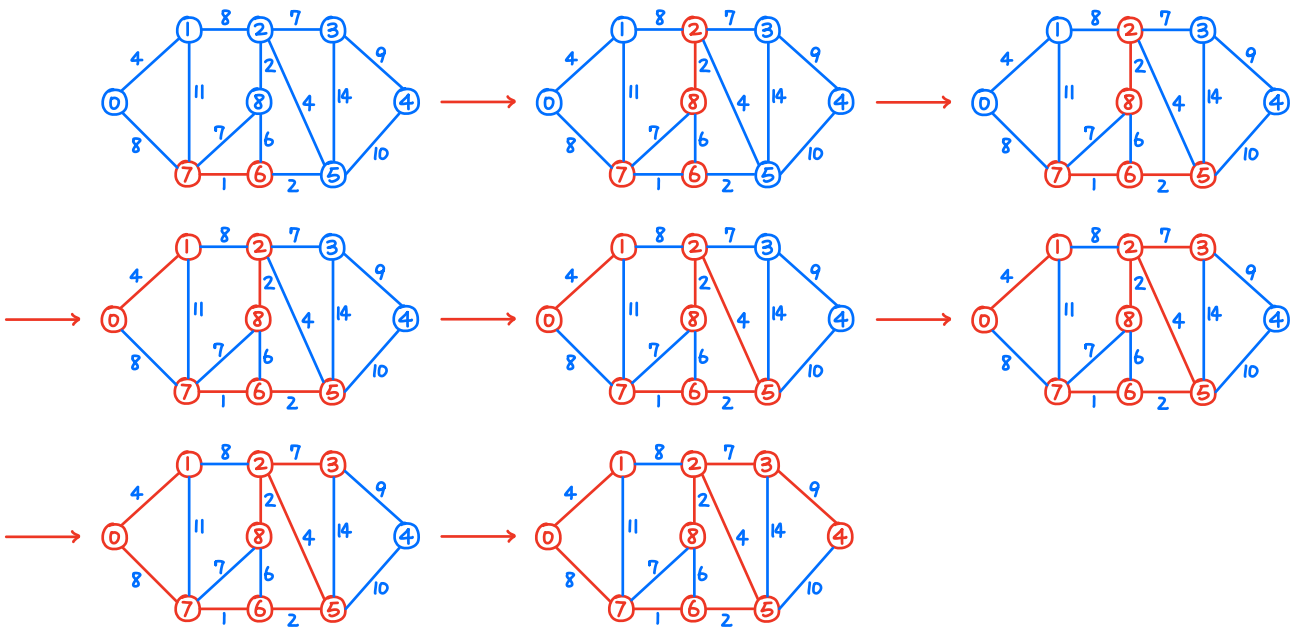
↓ 滾動數組

$S(n) = O(V^2)$

* Prim's Algorithm (MST): expand the tree by adding the minimum weight edge.
With binary heap: $T(n) = O(E \log V)$



* Kruskal's Algorithm (MST): include the minimum weight edge if it does not form a cycle
With union find: $T(n) = O(E \log V)$



模

板

```
#avg case: O(nlogn) worst case: O(n^2)
def quickSort(A, start, end):
    if start >= end:
        return
    idx = partition(A, start, end)
    quickSort(A, start, idx-1)
    quickSort(A, idx+1, end)

def partition(A, start, end)
    mid = start + (end-start)//2
    pivot = A[mid]
    A[start], A[mid] = A[mid], A[start]
    left, right = start+1, end
    while left <= right:
        while left<=right and A[left] <pivot:
            left += 1
        while left<=right and A[right]>pivot:
            right -= 1
        if left <= right:
            A[left], A[right] = A[right], A[left]
            left += 1
            right -= 1
    A[start], A[right] = A[right], A[start]
    return right

def sortColors2(colors, k):
    return quickSort(colors, 1, k, 0, len(colors)-1)

def quickSort(colors, colorStart, colorEnd, start, end):
    if colorStart >= colorEnd or start >= end:
        return
    colorPivot = (colorStart + colorEnd)//2
    left, i, right = start, start, end
    while i <= right:
        if colors[i] <= colorPivot:
            colors[left], colors[i] = colors[i], colors[left]
            i += 1
            left += 1
        elif colors[i] == colorPivot:
            i += 1
        else:
            colors[right], colors[i] = colors[i], colors[right]
            j -= 1
    quickSort(colors, colorStart, colorPivot-1, start, left-1)
    quickSort(colors, colorPivot+1, colorEnd, right+1, end)
```

```
#avg case: O(n) worst case: O(n^2)
def quickSelect(A, start, end, k):
    if start == end:
        return A[k]
    left, right = start, end
    pivot = A[start + (end-start)//2]
    while left <= right:
        while left<=right and A[left] <pivot:
            left += 1
        while left<=right and A[right]>pivot:
            right -= 1
        if left <= right:
            A[left], A[right] = A[right], A[left]
            left += 1
            right -= 1
    if k <= right:
        return quickSelect(A, start, right, k)
    if k >= left:
        return quickSelect(A, left, end, k)
return nums[k]
```

Quick sort 两种常见写法：

1. 交换左右指针，注意：

分成： 

✓

而不是： 

x

反例：[1 1 1 1 1]

[ ] [1 1 1 1] X 死循环！

2. 柳老师法：把pivot提出来，然后分别和左右指针交换，就没上面这个问题：

[1 1 1 1 1]

[1] [1 1 1 1]

✓ 问题规模变小！Ok！

```
from collections import deque
def bfs(root):
    queue = deque([root])
    visited = set([root])
    while queue:
        for i in range(len(queue)): #分层遍历
            node = queue.popleft()
            for neighbor in node.neighbors:
                if neighbor not in visited:
                    queue.append(neighbor)
                    visited.add(neighbor)
```

```
def binarySearch(A, target):
    start, end = 0, len(A) - 1
    while start + 1 < end: #
        mid = start + (end - start) // 2 #
        if nums[mid] < target:
            start = mid #
        elif nums[mid] == target:
            end = mid #
        else:
            end = mid #
    if A[start] == target:
        return start
    if A[end] == target:
        return end
    return -1
```

```
def topSort(graph):
    from collections import deque
    indegrees = {node: 0 for node in graph}
    for node in graph:
        for neighbor in node.neighbors:
            indegrees[neighbor] += 1
    res = []
    queue = deque(node for node in graph if indegrees[node] == 0)
    while queue:
        node = queue.popleft()
        res.append(node)
        for neighbor in node.neighbors:
            indegrees[neighbor] -= 1
            if indegrees[neighbor] == 0:
                queue.append(neighbor)
    if len(res) == len(graph):
        return res
    return -1
```

```
def kthLargest(nums, k):
    from heapq import *
    h = []
    for num in nums:
        heapush(h, num)
        if len(heap) > k:
            heappop(h)
    return h[0]
```

```
def combinationSum(nums, target):
    nums.sort()
    res = []
    dfs(nums, target, 0, [], res)
    return res

def dfs(nums, target, start, state, res):
    if target == 0:
        res.append(state[:]) #deep copy
        return
    if target < 0:
        return
    for i in range(start, len(nums)):
        if i > start and nums[i] == nums[i-1]:
            continue
        state.append(nums[i])
        self.dfs(nums, target - nums[i], i+1, state, res)
        state.pop() #backtrack
```

```
def middleNode(head):
    slowPtr, fastPtr = head, head
    while (fastPtr is not None) and (fastPtr.next is not None):
        slowPtr = slowPtr.next
        fastPtr = fastPtr.next.next
    return slowPtr
```

```
def reverseList(head):
    cur = head
    prev = None
    while cur is not None:
        temp = cur.next
        cur.next = prev
        prev = cur
        cur = temp
    return prev
```

```
class Trie:
    def __init__(self):
        self.children={}
        self.isWord=False

    def insert(self, word):
        p=self
        for c in word:
            if c not in p.children:
                p.children[c]=Trie()
            p=p.children[c]
        p.isWord=True

    def _startsWith(self, prefix):
        p=self
        for c in prefix:
            if c not in p.children:
                return None
            p=p.children[c]
        return p

    def startsWith(self, prefix):
        return self._startsWith(prefix) is not None

    def search(self, word):
        p=self._startsWith(word)
        return p is not None and p.isWord
```

没children就造children也要往下走

```
class UnionFind:
    def __init__(self):
        self.parents={}
        self.ranks={}

    def add(self, x):
        if x in self.parents:
            return
        self.parents[x]=x
        self.ranks[x]=1

    def find(self, x):
        if self.parents[x]!=x:
            self.parents[x]=self.find(self.parents[x])
        return self.parents[x]

    def union(self, x, y):
        p=self.find(x)
        q=self.find(y)
        if p==q:
            return False
        if self.ranks[p]<self.ranks[q]:
            self.parents[p]=q
        elif self.ranks[p]>self.ranks[q]:
            self.parents[q]=p
        else:
            self.parents[q]=p
            self.ranks[p]+=1
        return True
```

path compression

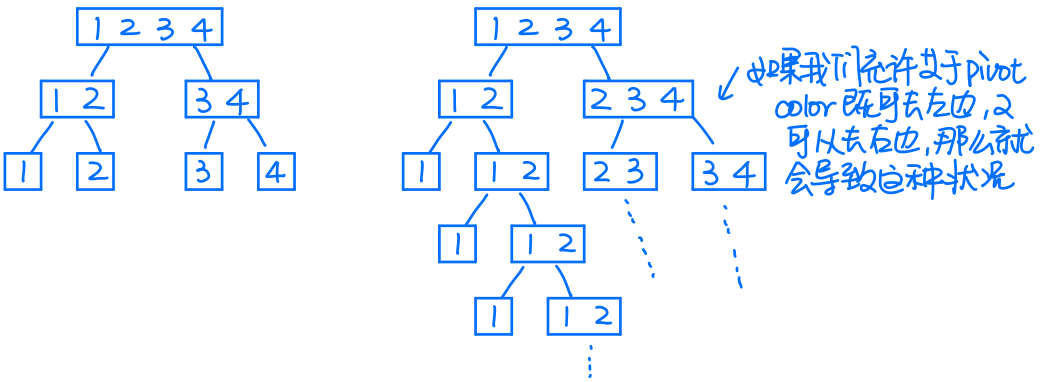
union by rank

題

Lintcode 143 Sort Colors II

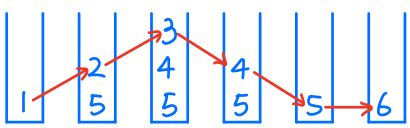
1. 分颜色 (这一步很像merge sort)

分治时候不能有重合，比如左边这个OK，右边这个不行，会stack overflow



2. Partition (这一步是类似Quick Sort)

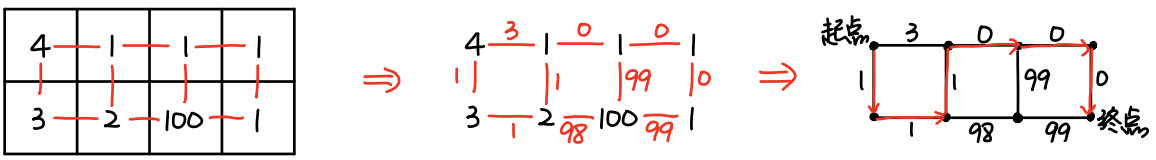
Leetcode 114 Flatten Binary Tree to Linked List



Lintcode 92 01背包+最多装多满
Lintcode 125 01背包+最大价值
Lintcode 440 完全背包+最大价值
Lintcode 562 完全背包+刚好填满问方案数
Lintcode 563 01背包+刚好填满问方案数
Lintcode 564 完全背包+刚好填满问方案数(permutation)
Lintcode 798 多重背包+最大价值
Lintcode 799 多重背包+可以达到问方案数
Lintcode 800 01背包+最大概率
Lintcode 801 完全背包+最多装多满

		算法1 (recursive DP)	算法2 (iterative DP)
Lintcode 107	Word Break (Any sol)	x 栈溢出	✓
Lintcode 582	Word Break II (#sols)	✓	x MLE
Lintcode 683	Word Break III (All sols)	✓	✓

Lintcode 258 Map Jump

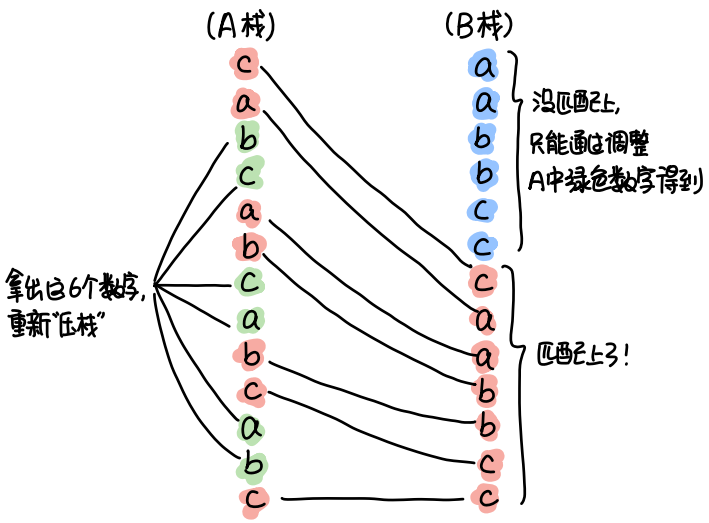


即转化为3-个在weighted graph上求最小cost路径问题
注意：这里为计算每条路径的cost，我们经过的所有边的权值，取max(而不是sum)
换句话说，我们目的是bottleneck，而不是路径和。dijkstra或bellman ford(不负责)求解即可

Lintcode 307 Transform String

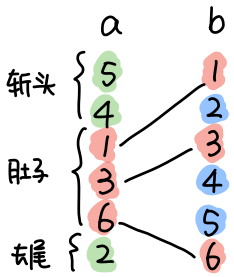
输入："cabcabcabcabc"　　正确答案：6
"aabbcccaabbcc"

1. 它个放入头部的动作有点像"压栈"
2. 那么不妨把A和B想象成两个栈
3. 问题即转换成如何把A栈调整成B栈
4. 统计那些没匹配上的数即可



Lintcode 313 Minimum insertion

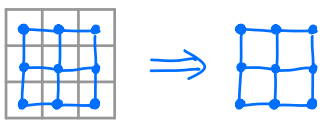
假设b是已经排好序的，我们要找到"肚子"最长有多长



Lintcode 611 Knight Shortest Path

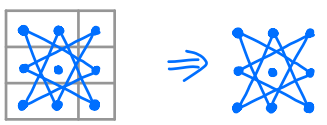
1. 考虑一个3×3的棋盘

2. 上下左右移动



对应图
(implicit graph)

3. 马字移动



对应图
(implicit graph)

Lintcode 1514 Robot Room Cleaner

```python
DIR = ((-1, 0), (0, 1), (1, 0), (0, -1))
class Solution:
    def cleanRoom(self, robot):
        self.dfs(0, 0, robot, 0, set())

    def goBack(self, robot):
        robot.turnLeft()
        robot.turnLeft()
        robot.move()
        robot.turnRight()
        robot.turnRight()

    def dfs(self, x, y, robot, d, visited):
        visited.add((x, y))
        robot.clean()
        for i in range(4):
            nd = (d + i) % 4
            dx, dy = DIR[nd]
            nx, ny = x + dx, y + dy
            if (nx, ny) not in visited and robot.move():
                self.dfs(nx, ny, robot, nd, visited)
                self.goBack(robot)
            robot.turnRight()
```

写法1

①把当前点加进去

```python
DIR = ((-1, 0), (0, 1), (1, 0), (0, -1))
class Solution:
    def cleanRoom(self, robot):
        visited = set()
        visited.add((0, 0))
        self.dfs(0, 0, robot, 0, visited)

    def goBack(self, robot):
        robot.turnLeft()
        robot.turnLeft()
        robot.move()
        robot.turnRight()
        robot.turnRight()

    def dfs(self, x, y, robot, d, visited):
        robot.clean()
        for i in range(4):
            nd = (d + i) % 4
            dx, dy = DIR[nd]
            nx, ny = x + dx, y + dy
            if (nx, ny) not in visited and robot.move():
                visited.add((nx, ny))
                self.dfs(nx, ny, robot, nd, visited)
                self.goBack(robot)
            robot.turnRight()
```

写法2

②把下一个点加进去

Lintcode 1834 Grouping Options

1. 本质上是球同，盒同(不为空)的分组问题.
   R不过这里用排序的方式来选代表，比如[1,2]和[2,1]是我们认，我们选前者为代表
   dp[i][j]= i个球，j个盒子的方案数
   = sum(dp[i-j][k] for k in range(1, j+1))
   (先把j个工具人到盒子里去占位，然后把剩下的人分为1到j组)
   (简化后) = dp[i-1][j-1] + dp[i-j][j]

① 最小组为1的方案数 ② 最小组>1的方案数，既然每组都>1，
总数就减j，但还是分成非空的j组(不然会产生矛盾)

2. 球同，盒同(可为空)
   dp[i][j] = dp[i][j-1] + dp[i-j][j]

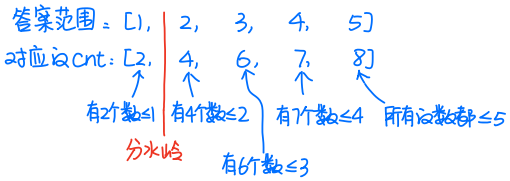
① 最小组为0的方案数 ② 最小组>0的方案数

Leetcode 4 Median of two sorted arrays

如果我们统计 ≤ 每个候选答案(假设已排序)的数的个数 cnt，这个 cnt 数组一定是(非严格)递增的。

而我们要找的是 the first position where cnt ≥ K

举个例子：A=[1,2,3,4], B=[1,2,3,5], K=3 (肉眼可见答案应该是 2)



我们要找啥？最小的一个候选答案 such that 有3个数(或更多)≤这个候选答案

⟺ 第一个候选答案 such that 有3个数(或更多)≤这个候选答案

回到我们的图上，那条分水岭就是我们要的！

1) 在分水岭之前，≤候选答案的数不足3个

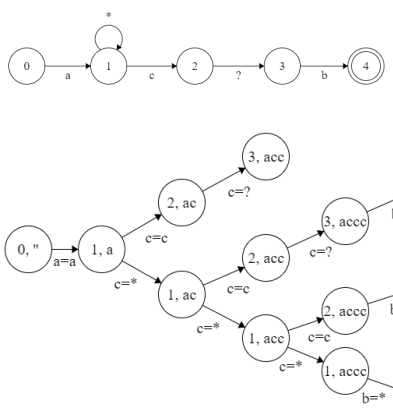
2) 在分水岭之后，≤候选答案的数到达3个以上

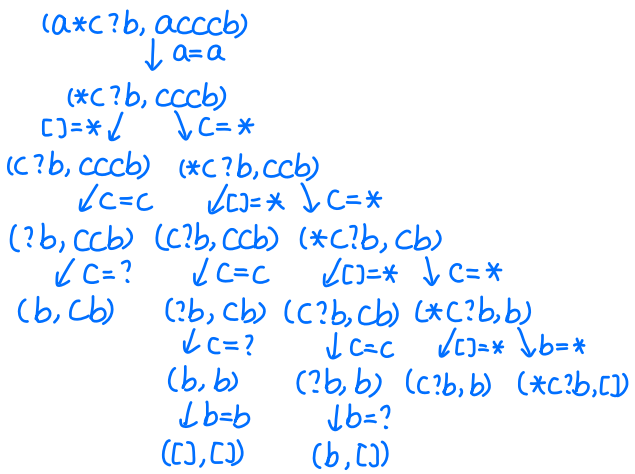
如果分水岭存在，那么就是一个典型的圈圈叉叉问题，二分法一定可以给到我们答案。

现在的问题是，有没有可能这个分水岭不存在？或者说这个 OOXX 的 pattern 不存在？

我认为是不可能的，因为 cnt 数组一定是(非严格)递增的，自然会有那么一条分水岭。

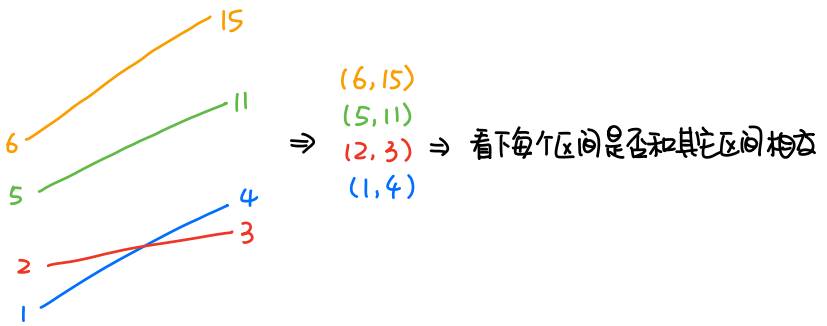
Leetcode 44 Wildcard matching



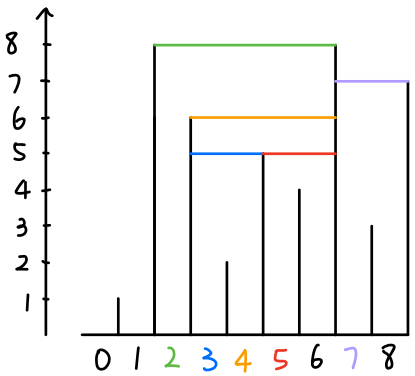


[团建题] 相交的直线。给出 n 条不同的直线，每条直线可以用两个参数(k,b)表示，即 y=kx+b

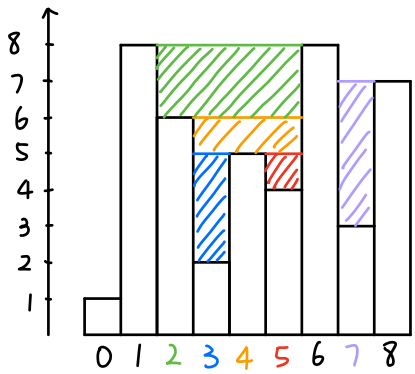
同时规定一个 x 轴上的区间范围 [lo, hi]，问在这个范围内，有多少条直线至少与其他一条直线相交。



Leetcode 11 Container with Most Water

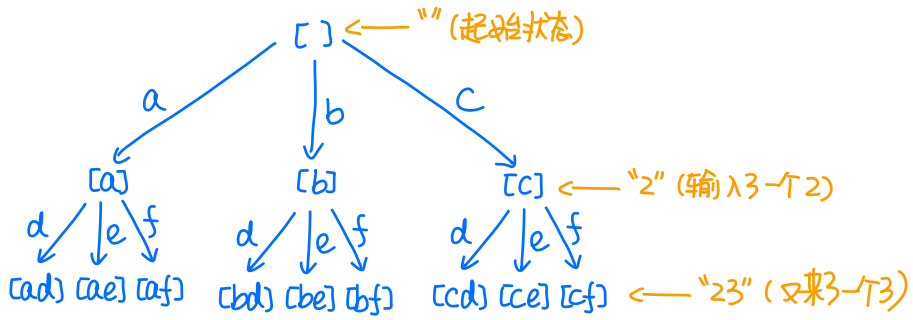


Leetcode 42 Trapping Rain Water

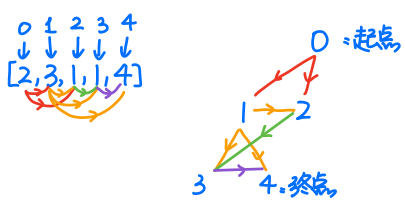


Leetcode 17 Letter Combinations of a Phone Number

这道题里面，图的构成是取决于输入的。假如输入为"23"：

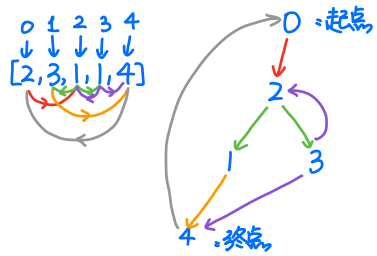


Leetcode 55 Jump Game & Leetcode 45 Jump Game II



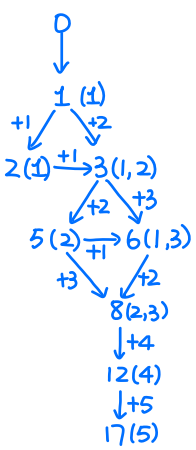
转换成3个在简单图上求最短路径的问题 ⇒ 直接BFS求解即可

Leetcode 1306 Jump Game III



Leetcode 403 Frog Jump

BFS
1) 按层遍历
2) 不需要记录 seen



Leetcode 287 Find the duplicate number

1. 两种标记方法

0	1	2	3	4
2	1	4	2	3

1) 用下标来标记

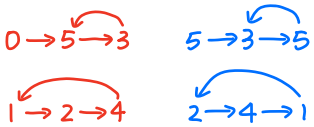
环的起点就是重复值

2) 用值来标记

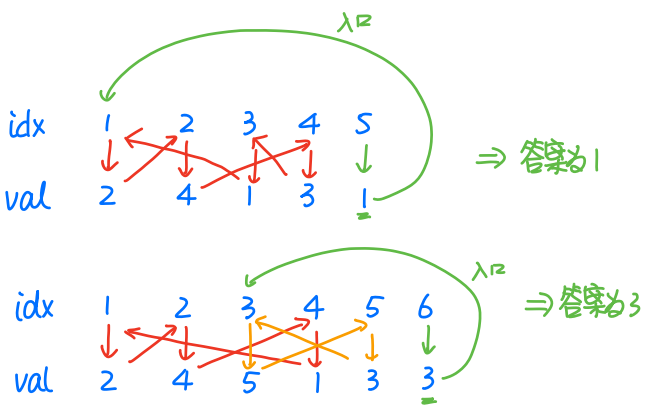
指向环的点就是重复值

2) 注意图中可能会存在多个环

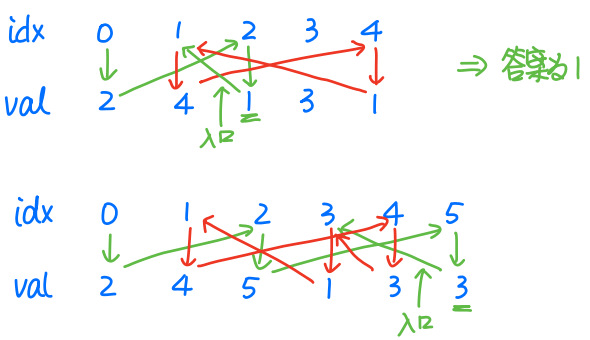
0	1	2	3	4	5
5	2	4	5	1	3

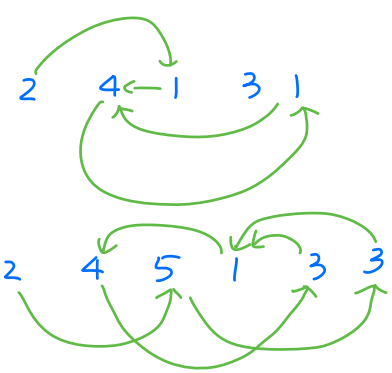


① 287 形式1做法 (1-index)



① 287 普通做法 (0-index)





Leetcode 300 LIS

dp[i]表示找长度为i+1的递增序列，末尾最小是多少，假如当前处理的数为x，如果前面有i-1个比x小的，x又比dp[i]小，那x就是更小的符合条件的dp[i]

0,	8,	4,	12,	2,	10,	6,	14,	1,	9,	5,	13,	3,	11,	7,	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	4	4	2	2	2	2	1	1	1	1	1	1	1	1
			12	12	10	6	6	6	6	5	5	3	3	3	3
							14	14	9	9	9	9	9	7	7
											13	13	11	11	11
															15

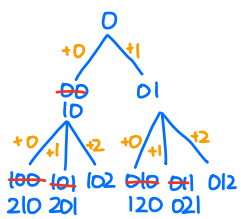
LIS

Leetcode 903 Valid Permutation for DI sequence

permutation还可以这样生成：

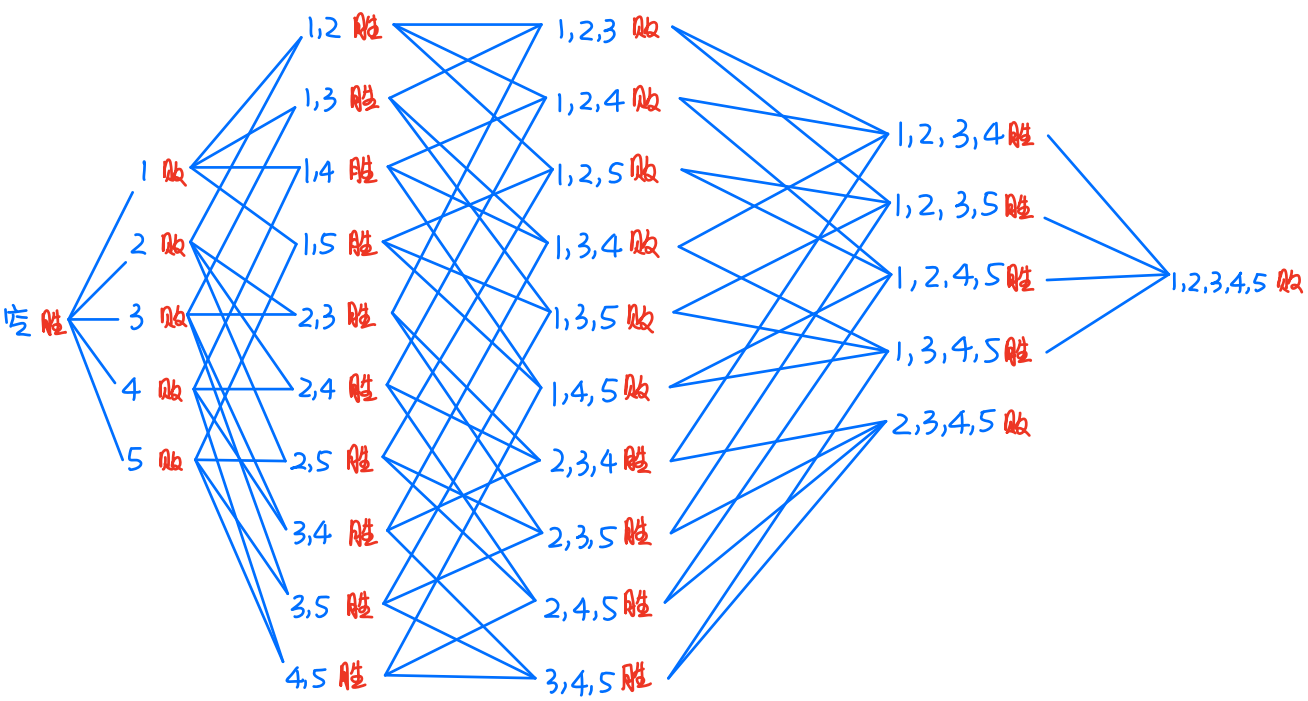
1）生成0～i的全排列，比如012, 021, 102, 120, 201, 210

2）在每一个上一轮生成的全排列后面贴一个0～i+1的数，并把之前位上大于等于当前数的进行提升



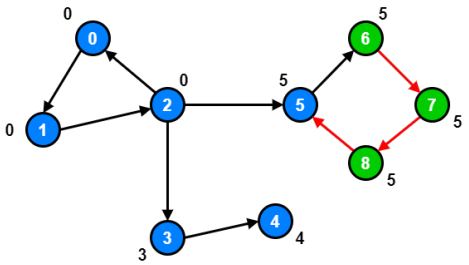
Leetcode 464 Can I win

n=5, target=14

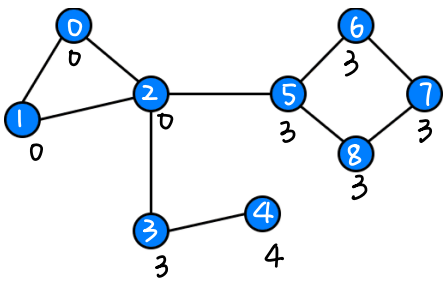


Leetcode 1192 Critical Connection in a Network

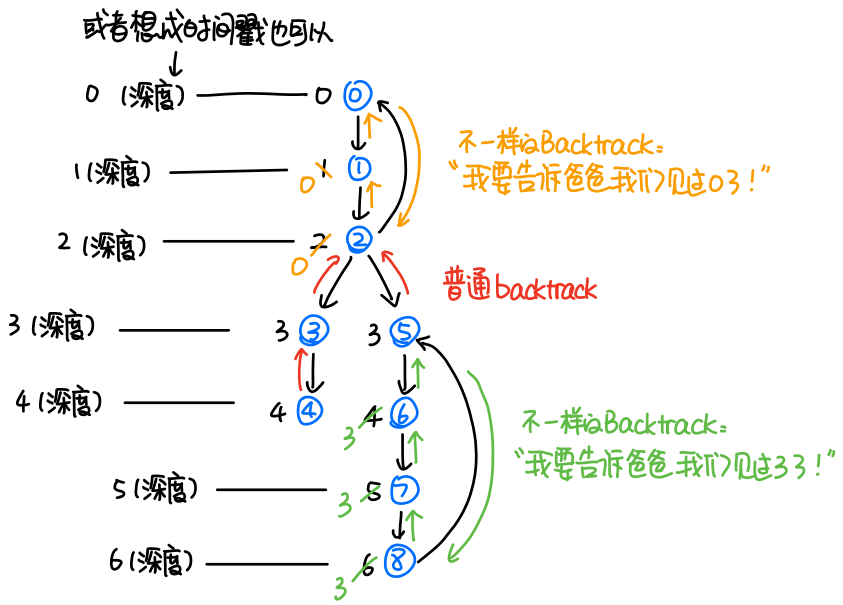
解法1：Tarjan Algorithm，维护3个全局dfs时间戳，2称low link
所有 u→v dfs边中，只要 u dfs时间戳小于 v dfs时间戳 ⇒ u——v是关键连接，2称bridge
⇒ 2→5，2→3，3→4



解法2：伪Tarjan(?)，利用DFS dfs递归深度来维护一个局部dfs时间戳



具体过程如下：



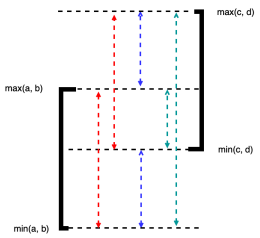
Leetcode 1330 Reverse Subarray to Maximum Array Value

$\cdots, a, [b, \cdots, c], d, \cdots$

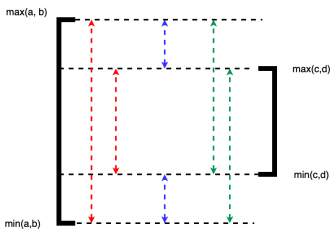
↓

$\cdots, a, [c, \cdots, b], d, \cdots$

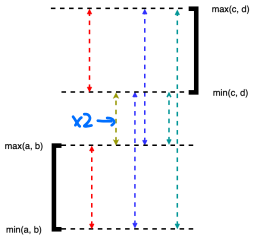
$diff = |a-c| + |b+d| - |a-b| - |c-d|$



(1a)



(1b)



(2)

分情况讨论：(1) 如果 $[\min(a,b), \max(a,b)]$ 和 $[\min(c,d), \max(c,d)]$ 这两个区间有重合，那么若果不会变好

(2) 如果 $[\min(a,b), \max(a,b)]$ 和 $[\min(c,d), \max(c,d)]$ 这两个区间无重合，则增值如图中黄线所示 (x2)

Leetcode 1439 Find the kth smallest sum of a matrix with sorted rows

$$\begin{bmatrix} a_1, a_2, a_3, a_4, a_5, a_6, a_7, \cdots, a_n \\ b_1, b_2, b_3, b_4, b_5, b_6, b_7, \cdots, b_n \\ c_1, c_2, c_3, c_4, c_5, c_6, c_7, \cdots, c_n \end{bmatrix} \qquad K=4$$

Claim: $a_5 + b_i + c_j$ 不可能为第4小的sum, $\forall i, j$

反证法：

假如 $a_5 + b_i + c_j$ 为第4小的sum, $\exists i, j$

⇒ 有且仅有3个sum比 $a_5 + b_i + b_j$ 小 ①

又有：

$$\begin{aligned} a_5 + b_i + c_j &\geq a_5 + b_1 + c_1 \\ &> a_4 + b_1 + c_1 \\ &> a_3 + b_1 + c_1 \\ &> a_2 + b_1 + c_1 \\ &> a_1 + b_1 + c_1 \end{aligned}$$

⇒ 我们找到了4个sum比 $a_5 + b_i + b_j$ 小 ②

①, ② 产生矛盾，证毕

Leetcode 1954 Minimum Garden Perimeter to Collect Enough Apples

8	7	6	5	4	5	6	7	8
7	6	5	4	3	4	5	6	7
6	5	4	3	2	3	4	5	6
5	4	3	2	1	2	3	4	5
4	3	2	1	0	1	2	3	4
5	4	3	2	1	2	3	4	3
6	5	4	3	2	3	4	5	2
7	6	5	4	3	4	5	6	1
8	7	6	5	4	5	6	7	8

$i=1: d(1) = 8\times1 + 8\times2 - 4\times1 - 4\times2$

$i=2: d(2) = 8\times2 + 8\times3 + 8\times4 - 4\times2 - 4\times4$

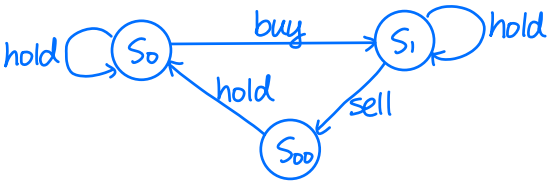
$i=3: d(3) = 8\times3 + 8\times4 + 8\times5 + 8\times6 - 4\times3 - 4\times6$

$i=4: d(4) = 8\times4 + 8\times5 + 8\times6 + 8\times7 + 8\times8 - 4\times4 - 4\times8$

$$\begin{aligned} \Rightarrow d(i) &= 8 \times (i+2i)(i+1)/2 - 4i - 4\times2i \\ &= 4\times3i\,(i+1) - 4i - 8i \\ &= 12i^2 + 12i - 12i \\ &= 12i^2 \end{aligned}$$

$$\begin{aligned} \Rightarrow \sum_{i=1}^{n} d(i) &= 12\times1^2 + 12\times2^2 + 12\times3^2 + 12\times4^2 + \cdots 12\times n^2 \\ &= 12 \times n(n+1)(2n+1)/6 \\ &= 2n(n+1)(2n+1) \end{aligned}$$

Leetcode 309 Best time to buy and sell stock with cool down

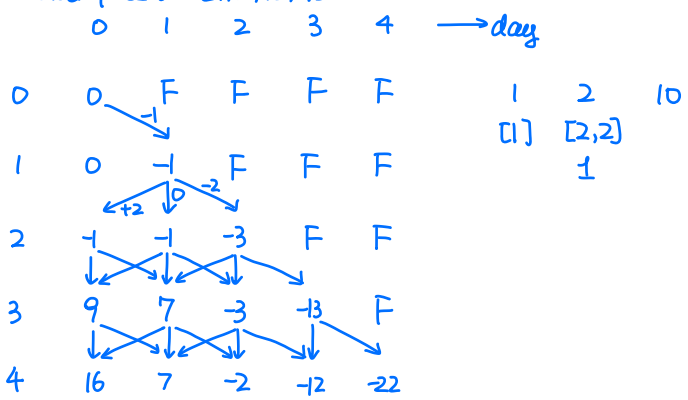


S0: 手上没股票
S1: 手上有股票
S00: 手上没股票，并且被冻住

Lintcode 1691. Best time to buy & sell stock V

dp[i][j] - profit of holding j stocks on day i

stock price a = [1,2,10,9]



1 2 10
[1] [2,2]
1

[讨论题目] We are writing a tool to help users manage their calendars at a corporation. Given a list of meeting times and sizes and a list of the available meeting room sizes, find a way to fit all of the meetings in a big enough room

room = [3,4,5] # 会议室大小

meeting = [(7,9,2),(8,12,3),(9,11,5),(9,11,4)] #起始时间，结束时间，需求

↓ 先预处理一下，找到能满足需求的最小会议室大小

meeting = [(7,9,3),(8,12,3),(9,11,5),(9,11,4)]

↓ 先按会议室大小，再按开始时间排序

meeting = [(9,11,5)①, (9,11,4)②, (7,9,3)③, (8,12,3)④]

然后，贪心地进行会议安排即可，如果安排不了，就返回 Impossible