3 个算法题, 3AC, 职位是 JAVA 开发工程师
第一题
leetcode 53 Maximum subarray problem , 算法导论第四章
我就贴我笔记里的, 我自己交的没存
algo
time O(n) space O(1)
n 表示数组长度
DP(具有最优化子结构), 可用数学归纳法证明 , 考虑以 num[i],0<=i<n 结尾的所有子串
code
复制代码

```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        cur=nums[0]
        ans=cur
        for i in range(1,len(nums)):#  每次只考虑 nums 的前 i+1 个数
            # 这一行, cur 存着以 nums[i-1]结尾的子串的和的最大值
            if cur>0:
                cur+=nums[i]
            else:
                cur=nums[i]
            ans=max(ans,cur)
        return ans
```

第二题
描述
给了一个无向图, 无向图的边还有两种名字, 需要手动创建这个无向图,
再给一个起点和终点, 找出起点到终点的最短路径,
code
复制代码

```
import queue

# bfs with path
mp={}

dirname_mp = {1:"north",-1:"south",2:"east",-2:"west"}
# d
# north 1
# south -1
# east 2
# west -2

# d of s1 is s2
def union(s1,s2,d):
    con(s1,s2,d)
    con(s2,s1,-d)
```

```
3
1 def con(s1,s2,d):
4     if s1 not in mp :
1         mp[s1]={d:s2}
5     else:
1         mp[s1][d]=s2
6
1 union("DAOTIAN1", "TULU1",-1)
7 union("DAOTIAN", "TULU1",1)
1 union("TULU2", "TULU1",-2)
8 union("DAOTIAN", "TULU",-1)
1 union("CUNKOU", "TULU",1)
9 union("CUNKOU", "NONGSHE",2)
2 # TULU2 right part
0 union("TULU2", "JIEDAO",2)
2 union("LIUJIABUDI", "JIEDAO",-1)
1 union("JIEDAO", "TIEPU",-1)
2
2 union("JIEDAO1", "JIEDAO",-2)
2 union("JIEDAO1", "XIAOJIUGUAN",-1)
3 union("JIEDAO1", "GAOJIADAYUAN",2)
2
4 union("JIEDAO2", "GAOJIADAYUAN",-2)
2 union("TULU3", "JIEDAO2",-2)
5 union("QINGSHILU", "TULU3",-2)
2
6 union("PIANFANG", "ZHENGYUAN",-2)
2 union("FANTING", "ZHENGTING",-2)
7 union("XIYIFANG", "HOUYUAN",-2)
2
8 union("HOUYUAN", "GUIGE",-2)
2 union("ZHENGTING", "PIANTING",-2)
9 union("ZHENGYUAN", "ZHANGFANG",-2)
3
0 union("ZHENGYUAN", "GAOJIADAYUAN",-1)
3 union("ZHENGYUAN", "ZHENGTING",1)
1 union("ZHENGTING", "HOUYUAN",1)
3 union("HOUYUAN","HUAYUAN",1)
2 union("YASHI","GUIGE",-1)
3
3 q = queue.Queue()
3 sta,end= input().split(",")
4
3 class Step:
```

```
5    def __init__(self, name, prev=None):
3        self.name = name
6        self.prev = prev
3
7 q.put(Step(sta))
3 def popAll(q):
8    while not q.empty():
3        yield q.get()
9
4 def back_print(step):
0    buf = []
4    while step.prev!=None:
1        for d in [1,-1,2,-2]:
4            if d in mp[step.prev.name] and mp[step.prev.name][d]==step.name:
2                break
4        buf.append(dirname_mp[d])
3        step = step.prev
4    print(",".join(reversed(buf)))
4
4 covered = {sta}
5 while not q.empty():
4    for cur_step in list(popAll(q)):
6        for nxt_name in mp[cur_step.name].values():
4            if nxt_name==end:
7                back_print(Step(nxt_name, cur_step))
4                break
8            if nxt_name not in covered:
4                q.put(Step(nxt_name, cur_step))
9                covered.add(nxt_name)
```

第三题

描述

给一个数组，例如[1,4,1,1,2,0,5]，记作 arr,

一开始在位置为 0 的商店上，开着 arr[0]油量的车子

每个元素的索引代表一个商店的位置，每到一个位置为 i 的商店可以选择 换一个油量为 arr[i]车子，之前那个车子舍弃，或者保持原来的车子

数组长度 记作 n

输出 从位置为 0 的 商店到位置为 n-1 的商店 使用车子数量的最小值，如果到达不了输出-1

algo

time:O(n*n) space:O(n)

本质上还是一个二维 dp，只不过可以节省成一维 dp

我用 dp2[cnt][j]表示用 最多 cnt 辆车，到达 j 位置商店，最多能有多少油

code

复制代码

```
def main():
    arr = list(map(lambda x:int(x), input().split(',')))
    n = len(arr)
    if n<=1: return 0
    dp = [0]*n
    cnt=1
    dp[0]=arr[0]
    for i in range(1,n):
        dp[i] = dp[i-1] - 1
    while cnt<=n:
        for i in range(cnt-1, n):
            # dp[i] 未赋新值前存着 dp2[cnt][i]
            if i==n-1 and dp[i]>=0:
                print(cnt)
                return

            # 只靠 cnt 辆车能否到达位置为 i 的商店, 并换乘到这辆车
            if dp[i]>=0:
                dp[i]=max(arr[i],dp[i])
            else:
                pass

            # 这个地方逻辑比较复杂, 可以按 dp[i]是否>=0 分类讨论
            if i!=cnt-1:
                # dp[i-1] 存着 dp2[cnt+1][i-1]
                dp[i]=max(dp[i],dp[i-1]-1)
        cnt+=1
    print(-1)
main()
```