cheat sheet

一、几个模版

最小生成树

```
#最小生成树prim,输入用mat存编号为0~n-1的邻接表，表内元素为(x,d)x为下一个点d为权值，输出最小总权值
import heapq
def solve(mat):
    h=[(0,0)]
    n=len(mat)
    vis=[1 for i in range(n)]
    ans=0
    while h:
        (d,x)=heapq.heappop(h)
        if vis[x]:
            vis[x]=0
            ans+=d
            for (y,t) in mat[x]:
                if vis[y]:
                    heapq.heappush(h,(t,y))
    return ans
```

```
#最小生成树kruskal,输入同上
import heapq
p=[i for i in range(n)]
def find(x):
    if p[x]==x:
        return x
    p[x]=find(p[x])
    return p[x]
def union(x,y):
    u,v=find(x),find(y)
    p[u]=v
def solve(mat):
    h=[]
    ans=0
    for i in range(n):
        for (j,d) in mat[i]:
            heapq.heappush(h,(d,i,j))
    while h:
        (d,i,j)=heapq.heappop(h)
        if find(i)!=find(j):
            union(i,j)
            ans+=d
    return ans
```

最短路径

```
#dijkstra，输入同上,求出from和to之间的最短路径，需要保证输入无负权值，输出-1表示无法到达to
#若输入to则处理点对点问题，否则返回所有点的最短距离
import heapq
def solve(mat,f,to=-1):
    h=[(0,f)]
    n=len(mat)
    vis=[-1 for i in range(n)]
    while h:
        (d,x)=heapq.heappop(h)
        if x==to:
            return d
        if vis[x]==-1:
            vis[x]=d
            for (y,s) in mat[x]:
                if vis[y]==-1:
                    heapq.heappush(h,(d+s,y))
    return vis
```

拓扑排序

```
#输入为mat存邻接表，mat[i]为i指向的点的列表，输出-1表示有环，顺便做了输出字典序最小排序方式
import heapq
def solve(mat):
    n=len(mat)
    h=[]
    ru=[len(mat[i]) for i in range(n)]
    ans=[]
    for i in range(n):
        if ru[i]==0:
            heapq.heappush(h,i)
    while h:
        x=heapq.heappop(h)
        ans.append(x)
        for y in mat[x]:
            ru[y]-=1
            if ru[y]==0:
                heapq.heappush(h,y)
    if len(ans)<n:
        return -1
    else:
        return ans
```

判断无向图是否连通、有无回路

```
#输入同上，基于并查集和图论
p=[i for i in range(n)]
e=[0 for i in range(n)]
v=[1 for i in range(n)]
```

```
def find(x):
    if p[x]==x:
        return x
    p[x]=find(p[x])
    return p[x]
def union(x,y):
    s,t=find(x),find(y)
    if s==t:
        e[t]+=1
        return
    p[s]=t
    v[t]+=v[s]
    e[t]+=e[s]+1
def connect(mat):
    for i in range(n):
        for j in mat[i]:
            union(i,j)
    root=find(0)
    if v[root]==n:
        return True
    return False
def loop(mat):
    for i in range(n):
        for j in mat[i]:
            union(i,j)
    for i in range(n):
        r=find(i)
        if e[r]>=v[r]:
            return True
    return False
```

单调栈

```
#给定一个列表，输出每个元素之前小于它的最后一个元素的下标
def solve(lis):
    n=len(lis)
    stack=[]
    ans=[]
    for i in range(n):
        x=lis[i]
        while stack:
            (y,j)=stack[-1]
            if y>=x:
                stack.pop()
                continue
            break
        if not stack:
            stack.append((x,i))
            ans.append(-1)
        else:
            ans.append(stack[-1][1])
            stack.append((x,i))
    return ans
```

二、令人印象深刻的题目

02299: Ultra-QuickSort

http://cs101.openjudge.cn/practice/02299/

思路：

用归并排序计算逆序数

代码

```
#
def solve(lis):
    if len(lis)==1:
        #print(lis)
        return (lis,0)
    k=len(lis)//2
    l=len(lis)-k
    le=lis[:k]
    ri=lis[k:]
    u=solve(le)
    v=solve(ri)
    p=u[0];q=v[0]
    i=0;j=0
    ans=0
    num=[]
    while i<k or j<l:
        if j<l:
            if i<k and p[i]<=q[j]:
                num.append(p[i])
                ans+=j
                i+=1
            else:
                num.append(q[j])
                j+=1
        else:
            ans+=j
            num.append(p[i])
            i+=1
    #print(num)
    return (num,u[1]+v[1]+ans)
while True:
    n=int(input())
    if n==0:
        break
    lis=[]
    d=0
    for _ in range(n):
        temp=int(input())
        lis.append(temp)
    print(solve(lis)[1])
```

24591: 中序表达式转后序表达式

http://cs101.openjudge.cn/practice/24591/

思路：

先处理输入，然后依次遍历输入列表，如果是数，直接输出；如果是左括号，先入栈；如果是运算符，就把此前能确定先运算的运算符出栈输出，即输出所有比目前元素高级的运算符直到上一个括号或栈底；如果是右括号，就输出至上一个左括号

代码

```
#
n=int(input())
for _ in range(n):
    s=input()
    lis=[];temp=''
    for x in s:
        if x in '+-*/()':
            if temp!='':
                lis.append(temp)
                temp=''
            lis.append(x)
        else:
            temp+=x
    if temp!='':
        lis.append(temp)
    os=[]
    for x in lis:
        if x=='(':
            os.append(x)
        if x not in '+-*/()':
            print(x,end=' ')
        if x in '+-*/':
            if os:
                y=os[-1]
                if x in '+-':
                    while os and os[-1] not in '()':
                        y=os.pop()
                        print(y,end=' ')
                else:
                    while os and os[-1] not in '+-()':
                        y=os.pop()
                        print(y,end=' ')
            os.append(x)
        if x==')':
            y=os.pop()
            while y!='(':
                print(y,end=' ')
                y=os.pop()
    while os:
        print(os.pop(),end=' ')
    print()
```

晴问9.5: 平衡二叉树的建立

https://sunnywhy.com/sfbj/9/5/359

思路:

照猫画虎，好难

代码

```
#
class node:
    def __init__(self,key):
        self.key=key
        self.p=None
        self.l=None
        self.r=None
        self.h=0
def hi(x):
    if x==None:
        return -1
    else:
        return x.h
def lr(rt):
    global root
    x=rt.r;y=x.l;z=rt.p
    x.p=z
    if z:
        if rt==z.l:
            z.l=x
        else:
            z.r=x
    rt.p=x
    x.l=rt
    rt.r=y
    if y:
        y.p=rt
    rt.h=1+max(hi(rt.l),hi(rt.r))
    x.h=1+max(hi(x.l),hi(x.r))
    if z:
        z.h=1+max(hi(z.l),hi(z.r))
    y=rt
    while y.p:
        y=y.p
    root=y
def rr(rt):
    global root
    x=rt.l;y=x.r;z=rt.p
    x.p=z
    if z:
        if rt==z.l:
            z.l=x
        else:
            z.r=x
    rt.p=x
    x.r=rt
```

```
    rt.l=y
    if y:
        y.p=rt
    rt.h=1+max(hi(rt.l),hi(rt.r))
    x.h=1+max(hi(x.l),hi(x.r))
    if z:
        z.h=1+max(hi(z.l),hi(z.r))
    y=rt
    while y.p:
        y=y.p
    root=y
n=int(input())
lis=list(map(int,input().split()))
root=node(lis[0])
for i in lis[1:]:
    x=node(i)
    y=root;z=root
    while y:
        z=y
        if y.key<=i:
            y=y.r
        else:
            y=y.l
    if z.key<=i:
        z.r=x
    else:
        z.l=x
    x.p=z
    t=x
    while t:
        t.h=1+max(hi(t.l),hi(t.r))
        t=t.p
    t=x
    while True:
        if not t:
            break
        t.h=1+max(hi(t.l),hi(t.r))
        if abs(hi(t.l)-hi(t.r))<=1:
            t=t.p
        else:
            if hi(t.l)>hi(t.r):
                x=t.l
                y=x.l;z=x.r
                if hi(y)<hi(z):
                    lr(x)
                rr(t)
            else:
                x=t.r
                y=x.r;z=x.l
                if hi(y)<hi(z):
                    rr(x)
                lr(t)
            t=t.p
ans=[]
def solve(rt):
    if not rt:
        return
    ans.append(rt.key)
```

```
    solve(rt.l)
    solve(rt.r)
solve(root)
print(' '.join(map(str,ans)))
```

04089: 电话号码

trie, http://cs101.openjudge.cn/practice/04089/

Trie 数据结构可能需要自学下。

思路：

用字典嵌套表示trie在其中进行插入和查询即可

代码

```
#
for ____ in range(int(input())):
    trie={}
    n=int(input())
    lis=[]
    for _ in range(n):
        lis.append(input())
    lis.sort(key=len,reverse=True)
    flag=1
    for w in lis:
        temp=trie;l=len(w);f=1
        for x in w:
            if x in temp:
                temp=temp[x]
            else:
                f=0
                temp[x]={}
                temp=temp[x]
        if f:
            flag=0
    if flag:
        print("YES")
    else:
        print("NO")
```

03441: 4 Values whose Sum is 0

data structure/binary search, http://cs101.openjudge.cn/practice/03441

思路：

用一个字典存储AB相加后每个和的个数，再对CD遍历计算

代码

```
#
A=[];B=[];C=[];D=[]
n=int(input())
```

```
for _ in range(n):
    a,b,c,d=map(int,input().split())
    A.append(a);B.append(b);C.append(c);D.append(d)
dic={}
for i in range(n):
    for j in range(n):
        x=A[i]+B[j]
        if x in dic:
            dic[x]+=1
        else:
            dic[x]=1
ans=0
for i in range(n):
    for j in range(n):
        x=-(C[i]+D[j])
        if x in dic:
            ans+=dic[x]
print(ans)
```

22067: 快速堆猪

http://cs101.openjudge.cn/practice/22067/

思路:

用两个栈分别存猪和这个猪进去之后的最小值

代码

```
#
pigstack=[]
minstack=[]
mi=-1
while True:
    try:
        s=input().split()
        if s[0]=='push':
            n=int(s[1])
            pigstack.append(n)
            if mi==-1:
                minstack.append(n)
                mi=n
            elif mi>=n:
                minstack.append(n)
                mi=n
        if s[0]=='pop':
            if not pigstack:
                continue
            temp=pigstack.pop()
            if temp==mi:
                minstack.pop()
                if minstack:
                    mi=minstack[-1]
                else:
```

```
                    mi=-1
        if s[0]=='min':
            if mi>=0:
                print(mi)
    except EOFError:
        break
```

27947: 动态中位数

http://cs101.openjudge.cn/practice/27947/

思路:

用两个栈维护左半边与右半边

代码

```
#
import heapq

for _____ in range(int(input())):
    lis=list(map(int,input().split()))
    n=len(lis)
    if True:
        print((n+1)//2)
        h1=[-lis[0]];h2=[lis[0]]
        print(h2[0],end=' ')
        for i in range(1,1+(n-1)//2):
            x,y=lis[2*i-1],lis[2*i]
            if x>y:
                x,y=y,x
            mid=h2[0]
            if x<=mid and y>=mid:
                heapq.heappush(h1,-x)
                heapq.heappush(h2,y)
            if x>mid:
                heapq.heappop(h2)
                heapq.heappush(h2,x)
                heapq.heappush(h2,y)
                heapq.heappush(h1,-h2[0])
                mid=h2[0]
            if y<mid:
                heapq.heappop(h1)
                heapq.heappush(h1,-x)
                heapq.heappush(h1,-y)
                heapq.heappush(h2,-h1[0])
                mid=h2[0]
            print(mid,end=' ')
        print()
```

04135: 月度开销

http://cs101.openjudge.cn/practice/04135/

思路:

注意到题目数据数据总和上界不大，采用对目标值二分查找

代码

```
#
def check(lis,k,m):
    c=0;i=0;temp=0
    for x in lis:
        if x>k:
            return False
        if temp+x<=k:
            temp+=x
        else:
            c+=1
            temp=x
        if c>=m:
            return False
    return True
n,m=map(int,input().split())
lis=[int(input()) for i in range(n)]
l=sum(lis)//m+1;r=sum(lis)
while l<r:
    k=(l+r)//2
    if check(lis,k,m):
        r=k
    else:
        l=k+1
print(r)
```

01182: 食物链

http://cs101.openjudge.cn/practice/01182/

思路:

把一个点、该点所吃的、被该点吃的共3n个点做成并查集

代码

```
#

def find(x):
    if p[x]==x:
        return x
    p[x]=find(p[x])
    return p[x]
def union(x,y):
```

```
    u,v=find(x),find(y)
    p[u]=v
n,k=map(int,input().split())
p=[i for i in range(3*n+1)]
ans=0
for _ in range(k):
    d,x,y=map(int,input().split())
    if x>n or y>n:
        ans+=1
        continue
    if d==1:
        if find(x)==find(y+n) or find(x+n)==find(y):
            ans+=1
            continue
        union(x,y)
        union(x+n,y+n)
        union(x+2*n,y+2*n)
    if d==2:
        if find(x)==find(y) or find(x)==find(y+n):
            ans+=1
            continue
        union(x+n,y)
        union(x,y+2*n)
        union(x+2*n,y+n)
print(ans)
```

28046: 词梯

bfs, http://cs101.openjudge.cn/practice/28046/

思路:

先利用字典存每种可能的词桶，便于建图，bfs

代码

```
#
class word:
    def __init__(self,k):
        self.k=k
        self.n=[]
n=int(input())
dic1={};dic2={};dic={}
for ___ in range(n):
    w=input()
    for i in range(0,4):
        ww=w[:i]+'_'+w[i+1:]
        if w not in dic1:
            dic1[w]=[ww]
        else:
            dic1[w].append(ww)
        if ww not in dic2:
            dic2[ww]=[w]
        else:
```

```
            dic2[ww].append(w)
for w in dic1:
    dic[w]=word(w)
for w in dic1:
    x=dic[w]
    for ww in dic1[w]:
        for v in dic2[ww]:
            if v!=w:
                x.n.append(dic[v])
    #print(list(map(lambda x:x.k,dic[w].n)))
from collections import deque
f,t=map(lambda x:dic[x],input().split())
q=deque([f]);p={f:None}
while q:
    x=q.popleft()
    for y in x.n:
        if y not in p:
            q.append(y)
            p[y]=x
if t not in p:
    print("NO")
else:
    ans=[]
    while t:
        ans.append(t.k)
        t=p[t]
    print(" ".join(reversed(ans)))
```

28050: 骑士周游

dfs, http://cs101.openjudge.cn/practice/28050/

思路：

神奇的优化：dfs的时候走下一步可行路径最少的，这样可以避免一头扎进节点海里出不来

代码

```
#
n=int(input())
x,y=map(int,input().split())
vis=[[0 for i in range(n)] for j in range(n)]
vis[x][y]=1
def check(x,y,n):
    if x<0 or x>n-1 or y<0 or y>n-1:
        return False
    return True
def nei(i,j):
    ans=0
    for k in range(8):
        ii=i+dx[k];jj=j+dy[k]
        if check(ii,jj,n) and vis[ii][jj]==0:
            ans+=1
    return ans
dx=[-2,-1,1,2,2,1,-1,-2]
dy=[1,2,2,1,-1,-2,-2,-1]
```

```python
def dfs(x,y,n,c):
    if c==n*n:
        return 1
    vis[x][y]=1
    lis=[(x+dx[i],y+dy[i]) for i in range(8)]
    lis.sort(key=lambda x:nei(x[0],x[1]))
    for (xx,yy) in lis:
        if check(xx,yy,n) and vis[xx][yy]==0:
            if dfs(xx,yy,n,c+1):
                return True
    vis[x][y]=0
    return False
if dfs(x,y,n,1):
    print("success")
else:
    print("fail")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44673788提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
n=int(input())
x,y=map(int,input().split())
vis=[[0 for i in range(n)] for j in range(n)]
vis[x][y]=1
def check(x,y,n):
    if x<0 or x>n-1 or y<0 or y>n-1:
        return False
    return True
def nei(i,j):
    ans=0
    for k in range(8):
        ii=i+dx[k];jj=j+dy[k]
        if check(ii,jj,n) and vis[ii][jj]==0:
            ans+=1
    return ans
dx=[-2,-1,1,2,2,1,-1,-2]
dy=[1,2,2,1,-1,-2,-2,-1]
def dfs(x,y,n,c):
    if c==n*n:
        return 1
    vis[x][y]=1
    lis=[(x+dx[i],y+dy[i]) for i in range(8)]
    lis.sort(key=lambda x:nei(x[0],x[1]))
    for (xx,yy) in lis:
        if check(xx,yy,n) and vis[xx][yy]==0:
            if dfs(xx,yy,n,c+1):
                return True
    vis[x][y]=0
    return False
if dfs(x,y,n,1):
    print("success")
else:
    print("fail")
```

基本信息

#:	44673788
题目:	28050
提交人:	23n2300010772(玩原玩的)
内存:	4128kB
时间:	74ms
语言:	Python3
提交时间:	2024-04-16 16:02:53

English 帮助 关于