Cheatsheet v1.0

Wrriten by Wangjie Su on 12/21/2023

02694: 波兰表达式

经典递归做法

```
m=0
def solve():
    global m
    a=l[m]
    m+=1
    if a=='+':
        return solve()+solve()
    elif a=='-':
        return solve()-solve()
    elif a=='*':
        return solve()*solve()
    elif a=='/':
        return solve()/solve()
    else:
        return float(a)

l=[i for i in input().split()]
n=solve()
print("%6f" % n)
```

02757: 最长上升子序列

dp，对每个点找到最长上升子序列

```
n=int(input())
s=[int(i) for i in input().split()]
r=[0]*(n+1)
for i in range(n):
    maxn=1
    for j in range(i):
        if s[j]<s[i] and r[j]+1>maxn:
            maxn=r[j]+1
    r[i]=maxn
print(max(r))
```

02773: 采药

0/1背包 模版题 反向更新 寻找约束条件下的最大or最小

```
t,m=map(int,input().split())
l=[[0]]
for i in range(m):
    l.append(list(map(int,input().split())))
ans=[0]*(t+1)
for i in range(1,m+1):
    for j in range(t,l[i][0]-1,-1):
        ans[j]=max(ans[j],ans[j-l[i][0]]+l[i][1])
print(ans[t])
```

02806: 公共子序列

dp，符合就整体往下，不符合就选一个最大的往下

```
while True:
    try:
        str1,str2=input().split()
        len1,len2=len(str1),len(str2)
        dp = [[0]*(len2+1) for _ in range(len1+1)]
        for i in range(len1):
            for j in range(len2):
                if str1[i]==str2[j]:
                    dp[i+1][j+1]=dp[i][j]+1
                else:
                    dp[i+1][j+1]=max(dp[i][j+1],dp[i+1][j])
        print(dp[len1][len2])
    except EOFError:
        break
```

03704: 括号匹配

stack结构，储存（位置在）处出队

```
while True:
    try:
        a=input()
        ans=[" "]*len(a)
        stack,ps=[""]*len(a),0
        for i in range(len(a)):
            if a[i]=="(":
                stack[ps]=i
                ps+=1
            elif a[i]==")":
                if (ps):
                    ps-=1
                else:
                    ans[i]="?"
        for _ in range(ps):
            ans[stack[_]]="$"
        print(a)
        print("".join(ans))
    except EOFError:
        break
```

04138: 质数的和与积

欧拉筛法

```
n=int(input())
if n%2==1:
    print(2*(n-2))
else:
    from math import sqrt
    ls,x,y=[True]*(n+1),2,int(sqrt(n))+1
    while x<y:
        if ls[x]==True:
            for i in range(x*2,n+1,x):
                ls[i]=False
        x+=1
    ls=[i for i in range(2,n+1) if ls[i]==True]
    i=0
    while True:
       if n//2-i in ls and n//2+i in ls:
           print((n//2-i)*(n//2+i))
           break
       i+=1
```

19757: Saruman's Army

其实也可以用线段覆盖做

```
while True:
        r, n = map(int, input().split())
        if r == -1 and n == -1:
            return
        res = 0
        troop = list(map(int, input().split()))
        troop.sort()
        i = 0
        while i < n:
            far = troop[i]
            while i < n and troop[i] - far <= r:
                i += 1
            res += 1
            far = troop[i - 1]
            while i < n and troop[i] - far <= r:
                i += 1
        print(res)
```

24834: 通配符匹配

典型的栈储存位置，然后回溯

```
def isMatch(s,p):
    slen=len(s)
    plen=len(p)
    s_idx=0
```

```
    p_idx=0
    p_idx2=-1
    s_idx2=-1
    while s_idx<slen:
        if p_idx<plen and (p[p_idx]=='?' or p[p_idx]==s[s_idx]):
            s_idx+=1
            p_idx+=1
        elif p_idx<plen and p[p_idx]=='*':
            p_idx2=p_idx
            s_idx2=s_idx
            p_idx+=1
        elif p_idx2==-1:
            return False
        else:
            p_idx=p_idx2+1
            s_idx=s_idx2+1
            s_idx2+=1
    while p_idx<plen:
        if p[p_idx]!='*':
            return False
        p_idx+=1
    return True

n=int(input())
for i in range(n):
    s=input()
    p=input()
    result=isMatch(s,p)
    if result:
        print("yes")
    else:
        print("no")
```

01088: 滑雪

记忆化搜索，储存位置信息dfs

```
def dfs(i,j):
    if dp[i][j]>0:
        return dp[i][j]
    else:
        for k in range(4):
            if 0<=i+d[k][0]<r and 0<=j+d[k][1]<c and maze[i][j]>maze[i+d[k][0]]
[j+d[k][1]]:
                dp[i][j]=max(dp[i][j],dfs(i+d[k][0],j+d[k][1])+1)
    return dp[i][j]

r,c=map(int,input().split())
maze=[]
for i in range(r):
    l=list(map(int,input().split()))
    maze.append(l)
dp=[[0]*c for _ in range(r)]
d=[[-1,0],[1,0],[0,1],[0,-1]]
ans=0
```

```
for i in range(r):
    for j in range(c):
        ans=max(ans,dfs(i,j))
print(ans+1)
```

18211:军备竞赛

双指针典范

```
p=int(input())
weapon=list(map(int,input().split()))
weapon.sort()
l,r=0,len(weapon)-1
ans=tmp=0
while l<=r:
    if p>=weapon[l]:
        p-=weapon[l]
        l+=1
        tmp+=1
    elif p<weapon[l] and tmp>0:
        tmp-=1
        p+=weapon[r]
        r-=1
    else:
        break
    ans=max(ans,tmp)
print(ans)
```

04116:拯救行动

bfs+heapq模版 所有bfs问题通解

```
import heapq
class position:
    def __init__(self,x,y,t):
        self.x=x
        self.y=y
        self.t=t
    def __lt__(self,other):
        return self.t<other.t
t=int(input())
results=[]
for _ in range(t):
    n,m=map(int,input().strip().split())
    maze=[[0]*m for _ in range(n)]
    visited=[[0]*m for _ in range(n)]
    start,end=None,None
    for i in range(n):
        line=list(input().strip())
        for j in range(m):
            if line[j]=='r':
                start=position(i,j,0)
                visited[i][j]=1
            elif line[j]=='a':
```

```
                end=(i,j)
            elif line[j]=='x':
                maze[i][j]=1
            elif line[j]=='#':
                maze[i][j]=-1
    queue=[]
    heapq.heappush(queue,start)
    dx=[1,0,-1,0]
    dy=[0,1,0,-1]
    while queue:
        x,y,time=queue[0].x,queue[0].y,queue[0].t
        if (x,y)==end:
            break
        for k in range(4):
            nx,ny=x+dx[k],y+dy[k]
            if 0<=nx<n and 0<=ny<m and visited[nx][ny]==0 and maze[nx][ny]!=-1:
                visited[nx][ny]=1
                if maze[nx][ny]==0:
                    new_time=time+1
                else:
                    new_time=time+2
                heapq.heappush(queue,position(nx,ny,new_time))
        heapq.heappop(queue)
    if not queue:
        results.append('Impossible')
    else:
        results.append(time)
for res in results:
    print(res)
```

21458:健身房 (dp)

恰好型dp，初始化为INF

```
t,n=map(int,input().split())
l=[[0]]
for i in range(n):
    l.append(list(map(int,input().split())))
ans=[0]+[-float("inf")]*t
for i in range(1,n+1):
    for j in range(t,l[i][0]-1,-1):
        ans[j]=max(ans[j],ans[j-l[i][0]]+l[i][1])
print(ans[t] if ans[t]>=0 else -1)
```

22007：N皇后问题

dfs典范

```
def isvalid(former,row,col):
    for i in range(row):
        if former[i]==col or abs(i-row)==abs(former[i]-col):
            return False
    return True
```

```
def queen(former=[],row=0):
    if row==n:
        result.append(former[:])
        return
    for col in range(n):
        if isvalid(former,row,col):
            former.append(col)
            queen(former,row+1)
            former.pop()

n=int(input())
result=[]
queen()
if result:
    for i in result:
        print(*i)
else:
    print("NO ANSWER")
```

23558:有界的深度优先搜索

dfs典范

```
def dfs(x,d):
    if d<=l-1:
        for i in tree[x]:
            if check[i]:
                check[i]=False
                ans.append(i)
                dfs(i,d+1)

n,m,l=map(int,input().split())
tree=[[] for _ in range(n)]
for i in range(m):
    a,b=map(int,input().split())
    tree[a].append(b)
    tree[b].append(a)
for i in tree:
    i.sort()
start=int(input())
check,ans=[True]*n,[start]
check[start]=False
dfs(start,0)
print(*ans)
```

24755:有多少种二叉树

卡特兰数，与入栈次数、括号匹配数、二叉搜索树的数量、凸多边形的三角划分等一样

```
n=int(input())
def f(n):
    num=0
    if n==0 or n==1:
        return 1
    else:
        for i in range(n):
            num+=f(i)*f(n-1-i)
        return num
print(f(n))
```

08210:河中跳房子

二分法典范

```
def check(x):
    t,num=0,0
    for i in range(1,n+1):
        if a[i]-t<x:
            num+=1
        else:
            t=a[i]
    if l-t<x:
        num+=1
    return num<=m

l,n,m=map(int,input().split())
a=[0]*(n+1)
for i in range(1,n+1):
    a[i]=int(input())
le,ri=0,l
while le+1<ri:
    mid=(le+ri)//2
    if check(mid):
        le=mid
    else:
        ri=mid
print(le)
```

04119:复杂的整数划分问题

N划分成K个正整数之和的划分数目、N划分成若干个不同正整数之和的划分数目、N划分成若干个奇正整数之和的划分数目

```
def test1(n,k):
    dp1=[[0]*51 for _ in range(51)]
    dp1[0][0]=1
    for i in range(1,n+1):
        for j in range(1,i+1):
            if i==j:
                dp1[i][j]=1
            else:
                dp1[i][j]=dp1[i-1][j-1]+dp1[i-j][j]
```

```
    print(dp1[n][k])

def test2(n):
    dp2=[[0]*51 for _ in range(51)]
    dp2[0][0]=1
    for i in range(1,n+1):
        for j in range(1,n+1):
            if i==j:
                dp2[i][j]=dp2[i][j-1]+1
            elif j>i:
                dp2[i][j]=dp2[i][i]
            else:
                dp2[i][j]=dp2[i-j][j-1]+dp2[i][j-1]
    print(dp2[n][n])

def test3(n):
    dp3=[[0]*51 for _ in range(51)]
    dp3[0][0]=1
    for i in range(1,n+1):
        for j in range(1,n+1):
            if j % 2==0:
                dp3[i][j]=dp3[i][j-1]
            else:
                if i<j:
                    dp3[i][j]=dp3[i][i]
                elif i==j:
                    dp3[i][j]=dp3[i][j-1]+1
                else:
                    dp3[i][j]=dp3[i-j][j]+dp3[i][j-1]
    print(dp3[n][n])
```

27205:护林员盖房子

前缀0的数目

```
m,n=map(int,input().split())
martix=[[1]*(n+2)]
martix.extend([[1]+list(map(int, input().split()))+[1] for _ in range(m)])
maze=[[0]*(n+1) for _ in range(m+1)]
for i in range(1,m+1):
    for j in range(1,n+1):
        if not martix[i][j]:
            maze[i][j]=maze[i][j-1]+1
        else:
            maze[i][j]=0
smax=0
for i in range(1,m+1):
    for j in range(1,n+1):
        if maze[i][j]!=0:
            width=1
            length=maze[i][j]
            area=width*length
            smax=max(smax,area)
            for k in range(i-1,0,-1):
                if maze[k][j]:
```

```
                    width+=1
                    length=min(length,maze[k][j])
                    smax=max(smax,width*length)
                else:
                    break
print(smax)
```

18155:组合乘积

dfs模版

```
def dfs(n):
    for num in s:
        if num>n or n%num!=0 or num in vis:
            continue
        elif num==n:
            return True
        else:
            vis.add(num)
            if dfs(n//num):
                return True
            vis.discard(num)
    return False

t=int(input())
s=set(map(int,input().split()))
vis=set()
flag=dfs(t)
print("YES" if flag else "NO")
```

01321:棋盘问题

dfs模版

```
ans=num=0
def dfs(a):
    global ans,num
    if num==k:
        ans+=1
        return
    for i in range(a+1,n):
        for j in range(n):
            if martix[i][j]=="#" and i not in setx and j not in sety:
                setx.add(i)
                sety.add(j)
                num+=1
                dfs(i)
                num-=1
                setx.remove(i)
                sety.remove(j)

while True:
    n,k=map(int,input().split())
    if n==-1 and k==-1:
```

```
        break
    martix=[list(input()) for i in range(n)]
    ans,setx,sety=0,set(),set()
    dfs(-1)
    print(ans)
```

01384:Piggy-Bank

完全背包模版 从头到尾遍历

```
for _ in range(int(input())):
    e,f=map(int,input().split())
    tw=f-e
    n=int(input())
    p,w=[],[]
    for i in range(n):
        pi,wi=map(int,input().split())
        p.append(pi)
        w.append(wi)
    dp=[0]+[float("inf")]*tw
    for i in range(n):
        for j in range(w[i],tw+1):
            dp[j]=min(dp[j],dp[j-w[i]]+p[i])
    print(f"The minimum amount of money in the piggy-bank is {dp[tw]}."if
dp[tw]!=float("inf") else "This is impossible.")
```

01328:Radar Installation

线段覆盖问题通解，按右端点排序

```
class island:
    def __init__(self,left,right):
        self.left=left
        self.right=right
from math import sqrt
NO,p=1,[]
while True:
    n,d=map(int,input().split())
    if n==0 and d==0:
        break
    l,num,flag,i=[],1,True,0
    while i<n and flag:
        x,y=map(float,input().split())
        if y>d:
            flag=False
        else:
            left=x-sqrt(d**2-y**2)
            right=x+sqrt(d**2-y**2)
            l.append(island(left,right))
        i+=1
    for j in range(n-i+1):
        input()
    if not flag:
        p.append(f"Case {NO}: -1")
```

```
    else:
        l.sort(key=lambda x:x.left)
        left=l[0].left
        right=l[0].right
        for i in range(1,n):
            if l[i].left<=right:
                right=min(right,l[i].right)
            else:
                num+=1
                right=l[i].right
        p.append(f"Case {NO}: {num}")
    NO+=1
for i in p:
    print(i)
```

26971:分发糖果

|两遍扫实现

```
n=int(input())
l=list(map(int,input().split()))
num,numlist=0,[1]*n
for i in range(1,n):
    if l[i]>l[i-1]:
        numlist[i]=numlist[i-1]+1
    elif l[i]<l[i-1]:
        numlist[i-1]=numlist[i]+1
for i in range(n-2,-1,-1):
    if l[i]>l[i+1]:
        numlist[i]=max(numlist[i+1]+1,numlist[i])
    elif l[i]<l[i+1]:
        numlist[i+1]=max(numlist[i]+1,numlist[i+1])
for i in range(n):
    num+=numlist[i]
print(num)
```

01742:Coins

多重背包问题，用二进制优化，将物品分为1、2、4.....倍的0/1背包问题

```
import math
def sum_2(x):
    s=0
    while x>0:
        s+=(x&1)
        x=x>>1
    return s

while True:
    n,m=map(int,input().split())
    if n==0 and m==0:
        break
    ls=list(map(int,input().split()))
    w=(1<<(m+1))-1
```

```
    result=1
    for i in range(n):
        number=ls[i+n]+1
        limit=int(math.log(number,2))
        rest=number-(1<<limit)
        for j in range(limit):
            result=(result|(result<<(ls[i]*(1<<j))))&w
        if rest>0:
            result=(result|(result<<(ls[i]*rest)))&w
    print(bin(result).count('1')-1)
```

02287:Tian Ji -- The Horse Racing

大抵大，小抵小，小抵大

```
def check(a,b):
    ans=0
    l1,r1=0,len(a)-1
    l2,r2=0,len(b)-1
    while l1<=r1 and l2<=r2:
        if b[r2]>a[r1]:
            r2-=1
            r1-=1
            ans+=200
        elif b[l2]>a[l1]:
            l1+=1
            l2+=1
            ans+=200
        elif b[l2]==a[r1]:
            l2+=1
            r1-=1
            ans+=0
        else:
            r1-=1
            l2+=1
            ans-=200
    return ans

while True:
    n=int(input())
    if n==0:
        break
    tian=list(map(int,input().split()))
    king=list(map(int,input().split()))
    tian.sort()
    king.sort()
    maxi=check(king,tian)
    print(maxi)
```

02385:Apple Catching

二维dp，在不同地点的同时间dp，也可与移动办公一样开两个数组

```
t,w=map(int,input().split())
```

```
l=[0]*(t+1)
for i in range(1,t+1):
    l[i]=int(input())
dp=[[0]*(w+1) for i in range(t+1)]
for i in range(1,t+1):
    for j in range(0,w+1):
        if j==0:
            dp[i][j]=dp[i-1][j]
        else:
            dp[i][j]=max(dp[i-1][j],dp[i-1][j-1])
        if l[i]-j&1==1:
            dp[i][j]+=1
print(max(dp[t]))

t,m=map(int,input().split())
p,n=[0]*(t+1),[0]*(t+1)
for i in range(1,t+1):
    p[i],n[i]=map(int,input().split())
dpp,dpn=[0]*(t+1),[0]*(t+1)
dpp[1],dpn[1]=p[1],n[1]
for i in range(2,t+1):
    dpp[i]=max(dpp[i-1]+p[i],dpn[i-1]+p[i]-m)
    dpn[i]=max(dpn[i-1]+n[i],dpp[i-1]+n[i]-m)
print(max(dpp[t],dpn[t]))
```

20127:寻宝3.0 v0.2

bfs+heap+条件 模版

```
import heapq
def bfs(x,y):
    d=[[-1,0],[1,0],[0,1],[0,-1]]
    queue=[]
    heapq.heappush(queue,[0,x,y])
    check=set()
    check.add((x,y))
    while queue:
        step,x,y=map(int,heapq.heappop(queue))
        if martix[x][y]==1:
            return step
        for i in range(4):
            dx,dy=x+d[i][0],y+d[i][1]
            if martix[dx][dy]!=2 and (dx,dy) not in check:
                heapq.heappush(queue,[step+(martix[dx][dy]!=3),dx,dy])
                check.add((dx,dy))
    return "NO"

m,n=map(int,input().split())
martix=[[2]*(n+2)]+[[2]+list(map(int,input().split()))+[2] for i in range(m)]+
[[2]*(n+2)]
print(bfs(1,1))
```

03263:新数字三角形

从下开始往上dp

```
while True:
    n=int(input())
    if n==0:
        break
    maxsum=[[0]*120 for _ in range(120)]
    a=[[0]*120 for _ in range(120)]
    for i in range(1,n+1):
        a[i][1:i+1]=map(int,input().split())
    x,y=map(int,input().split())
    for i in range(n,0,-1):
        for j in range(1,i+1):
            max_low=max(maxsum[i+1][j],maxsum[i+1][j+1])
            maxsum[i][j]=max(max_low,a[i][j])
    print(maxsum[x][y])
```

26977:接雨水

两端搜索维护最大值

```
n=int(input())
l=[int(i) for i in input().split()]
l_max,r_max,num=[0]*(n+1),[0]*(n+1),0
for i in range(n):
    l_max[i+1]=max(l_max[i],l[i])
for i in range(n-1,-1,-1):
    r_max[i]=max(r_max[i+1],l[i])
for i in range(n):
    num+=min(l_max[i+1],r_max[i])-l[i]
print(num)
```

26976:摆动序列

改变最长上升子序列的判定条件罢了

```
n=int(input())
s=[int(i) for i in input().split()]
r=[0]*n
d=[0]*n
for i in range(n):
    maxn=1
    for j in range(i):
        if r[j]==1 and s[i]!=s[j]:
            maxn=max(2,maxn)
            d[i]=s[i]-s[j]
        elif (s[i]-s[j])*d[j]<0 and r[j]+1>maxn:
            maxn=r[j]+1
            d[i]=s[i]-s[j]
    r[i]=maxn
print(max(r))
```

21462:加密的称赞 v0.2 (matrices)

逆时针矩阵旋转

```
n=int(input())
martix=[list(map(int,input().split())) for i in range(n)]
cnt_x,cnt_y,dx,dy=0,0,0,1
cnt_n,cnt_s,cnt_e,cnt_w=0,n-1,n-1,0
ans=[]
for i in range(1,n*n+1):
    if martix[cnt_y][cnt_x]==0:
        break
    ans.append(chr(martix[cnt_y][cnt_x]))
    cnt_x+=dx
    cnt_y+=dy
    if dx==1 and dy==0 and cnt_x==cnt_e and cnt_y==cnt_s:
        dx,dy=0,-1
        cnt_s-=1
    elif dx==0 and dy==1 and cnt_x==cnt_w and cnt_y==cnt_s:
        dx,dy=1,0
        cnt_w+=1
    elif dx==-1 and dy==0 and cnt_x==cnt_w and cnt_y==cnt_n:
        dx,dy=0,1
        cnt_n+=1
    elif dx==0 and dy==-1 and cnt_x==cnt_e and cnt_y==cnt_n:
        dx,dy=-1,0
        cnt_e-=1
print("".join(ans))
```

20121:解梦人kk

顺时针矩阵旋转

```
n=int(input())
martix=[list(map(int,input().split())) for i in range(n)]
cnt_x,cnt_y,dx,dy=0,0,1,0
cnt_n,cnt_s,cnt_e,cnt_w=0,n-1,n-1,0
ans=[]
for i in range(1,n*n+1):
    ans.append(str(martix[cnt_y][cnt_x]))
    cnt_x+=dx
    cnt_y+=dy
    if dx==1 and dy==0 and cnt_x==cnt_e and cnt_y==cnt_n:
        dx,dy=0,1
        cnt_n+=1
    elif dx==0 and dy==1 and cnt_x==cnt_e and cnt_y==cnt_s:
        dx,dy=-1,0
        cnt_e-=1
    elif dx==-1 and dy==0 and cnt_x==cnt_w and cnt_y==cnt_s:
        dx,dy=0,-1
        cnt_s-=1
    elif dx==0 and dy==-1 and cnt_x==cnt_w and cnt_y==cnt_n:
        dx,dy=1,0
        cnt_w+=1
```

```
print("".join(ans))
```

25570:洋葱

矩阵分层向内运算

```
n=int(input())
l=[]
for _ in range(n):
    l.append(list(map(int,input().split())))
indexn,indexs,indexw,indexe=0,n-1,0,n
ans=0
for i in range((n+1)//2):
    if indexn!=indexs:
        tmp=sum(l[indexn][indexw:indexe])+sum(l[indexs][indexw:indexe])
        for j in range(indexn+1,indexs):
            tmp+=l[j][indexw]+l[j][indexe-1]
    else:
        tmp=sum(l[indexn][indexw:indexe])
    ans=max(ans,tmp)
    indexn+=1
    indexs-=1
    indexw+=1
    indexe-=1
print(ans)
```

09267:核电站

m-1个随便，减去开头不能放的情况

```
n,m=map(int,input().split())
f=[0]*(n+1)
f[0]=1
for i in range(1,m):
    f[i]=f[i-1]*2
f[m]=f[m-1]*2-1
for i in range(m+1,n+1):
    f[i]=f[i-1]*2-f[i-m-1]
print(f[n])
```

02431:Expedition

加油问题，在路上的加油站全压入heap，然后在没油时加最多的

```
import heapq
n=int(input())
stops,fuel,line=[0],[0],[]
for i in range(n):
    line.append(list(map(int,input().split())))
line.sort()
for i in line:
    x,num=i
    stops.append(x)
```

```
    fuel.append(num)
l,p=map(int,input().split())
queue,ans=[],0
for i in range(n,-1,-1):
    d=l-stops[i]
    while d>p:
        if not queue:
            print(-1)
            quit()
        p+=-1*heapq.heappop(queue)
        ans+=1
    heapq.heappush(queue,-fuel[i])
    p-=d
    l=stops[i]
print(ans)
```