例题

1 经典题目复盘

1.1 递归

文件结构图 02775

```python
from sys import exit
class dir:
    def __init__(self, dname):
        self.name = dname
        self.dirs = []
        self.files = []
    def getGraph(self):
        g = [self.name]
        for d in self.dirs:
            subg = d.getGraph()
            g.extend(["|     " + s for s in subg])
        for f in sorted(self.files):
            g.append(f)
        return g
n = 0
while True:
    n += 1
    stack = [dir("ROOT")]
    while (s := input()) != "*":
        if s == "#": exit(0)
        if s[0] == 'f':
            stack[-1].files.append(s)
        elif s[0] == 'd':
            stack.append(dir(s))
            stack[-2].dirs.append(stack[-1])
        else:
            stack.pop()
    print(f"DATA SET {n}:")
    print(*stack[0].getGraph(), sep='\n')
    print()
```

1.2 并查集

食物链

```python
class DisjointSet:
    def __init__(self, n):
        #设[1,n] 区间表示同类, [n+1,2*n]表示x吃的动物, [2*n+1,3*n]表示吃x的动物。
        self.parent = [i for i in range(3 * n + 1)]
        #每个动物有三种可能的类型, 用 3 * n 来表示每种类型的并查集
        self.rank = [0] * (3 * n + 1)
    def find(self, u):
        if self.parent[u] != u:
            self.parent[u] = self.find(self.parent[u])
        return self.parent[u]
    def union(self, u, v):
        pu, pv = self.find(u), self.find(v)
        if pu == pv:
            return False
        if self.rank[pu] > self.rank[pv]:
            self.parent[pv] = pu
        elif self.rank[pu] < self.rank[pv]:
            self.parent[pu] = pv
        else:
            self.parent[pv] = pu
            self.rank[pu] += 1
        return True
def is_valid(n, k, statements):
    dsu = DisjointSet(n)

    def find_disjoint_set(x):
        if x > n:
            return False
        return True
    false_count = 0
    for d, x, y in statements:
        if not find_disjoint_set(x) or not find_disjoint_set(y):
            false_count += 1
            continue
        if d == 1:  # X and Y are of the same type
            if dsu.find(x) == dsu.find(y + n) or dsu.find(x) == dsu.find(y + 2 * n):
                false_count += 1
            else:
                dsu.union(x, y)
                dsu.union(x + n, y + n)
                dsu.union(x + 2 * n, y + 2 * n)
        else:  # X eats Y
            if dsu.find(x) == dsu.find(y) or dsu.find(x + 2*n) == dsu.find(y):
                false_count += 1
            else:
            #[1,n] 区间表示同类，[n+1,2*n]表示x吃的动物，[2*n+1,3*n]表示吃x的动物
                dsu.union(x + n, y)
                dsu.union(x, y + 2 * n)
                dsu.union(x + 2 * n, y + n)
    return false_count
if __name__ == "__main__":
    N, K = map(int, input().split())
    statements = []
    for _ in range(K):
        D, X, Y = map(int, input().split())
        statements.append((D, X, Y))
    result = is_valid(N, K, statements)
    print(result)
```

1.3：线段树

XXXXX

```
t = int(input())
ans = []
for _ in range(t):
    n, x = map(int, input().split())
    a = [int(i) for i in input().split()]
    # Max size of tree
    tree = [0] * (2 * n)
    def build(arr) :
        for i in range(n) :
            tree[n + i] = arr[i]
        for i in range(n - 1, 0, -1) :
            tree[i] = tree[i*2] + tree[i*2+1]
    def updateTreeNode(p, value) :
        tree[p + n] = value
        p = p + n
        i = p
        while i > 1 :
            tree[i//2] = tree[i] + tree[i ^ 1]
    def query(l, r) :
        res = 0
        l += n
        r += n
        while l < r :
            if (l%2==1) :
                res += tree[l]
                l += 1
            if (r%2==1) :
                r -= 1
                res += tree[r]
            l //= 2
            r //= 2
        return res
    build([i%x for i in a])
    left = 0
    right = n - 1
    if right == 0:
        if a[0] % x !=0:
            print(1)
        else:
            print(-1)
        continue
    leftmax = 0
    rightmax = 0
    while left != right:
        total = query(left, right+1)
        if total % x != 0:
            leftmax = right - left + 1
            break
        else:
            left += 1
    left = 0
    right = n - 1
    while left != right:
        total = query(left, right+1)
        if total % x != 0:
            rightmax = right - left + 1
            break
        else:
            right -= 1
    if leftmax == 0 and rightmax == 0:
        print(-1)
    else:
        print(max(leftmax, rightmax))
```

1.4.二分逼近

Sequence 02442

```
import heapq
for _ in range(int(input())):
    m,n = [int(i) for i in input().split()]
    ans = sorted([int(i) for i in input().split()])
    for i in range(m-1):
        l = sorted([int(i) for i in input().split()])
        heap,res,cnt = [],[],0
        for j in range(n):
            heapq.heappush(heap,(ans[j]+l[0],j,0))
        while True:
            num,x,y = heapq.heappop(heap)
            res.append(num)
            cnt += 1
            if cnt == n:
                break
            heapq.heappush(heap,(ans[x]+l[y+1],x,y+1))
        ans = res
    print(*ans)
```

1.5 判环

DAG判环 09202 舰队、海域出击

```
def dfs(node):
    visiting[node] = True
    for neighbor in graph[node]:
        if visiting[neighbor] is True:
            return True
        if visited[neighbor] is False and dfs(neighbor):
            return True
    visiting[node] = False
    visited[node] = True
    return False
for _ in range(int(input())):
    n,m = [int(i) for i in input().split()]
    graph = [0] + [[] for i in range(n)]
    for i in range(m):
        x,y = [int(i) for i in input().split()]
        graph[x].append(y)
    visited = [False for i in range(n+1)]
    visiting = [False for i in range(n+1)]
    cnt = 0
    for element in range(1,n+1):
        if visited[element] is False:
            if dfs(element) is True:
                cnt = 1
                print("Yes")
                break
    if cnt == 0:
        print("No")
```

1.6 懒更新

候选人追踪 27384

```
import heapq
maxn = 320000
cnt = [0]*maxn
vis = [False]*maxn
n,k = [int(i) for i in input().split()]
records = [int(i) for i in input().split()]
arr = [(records[i],records[i+1]) for i in range(0,2*n,2)]
Q = []
candidates = [int(i) for i in input().split()]
for i in range(k):
    heapq.heappush(Q,(0,candidates[i]))
    vis[candidates[i]] = True
arr = sorted(arr[:n])
if k == 314159:
    print(arr[n-1][0])
    exit()
rmx,rs = 0,0
for i in range(n):
    c = arr[i][1]
    cnt[c] += 1
    if vis[c]:
        while cnt[Q[0][1]]:
            f = heapq.heappop(Q)
            f = (f[0]+cnt[f[1]],f[1])
            heapq.heappush(Q,f)
            cnt[f[1]] = 0
    else:
        rmx = max(rmx,cnt[c])
    if i!=n-1 and arr[i+1][0] != arr[i][0] and Q[0][0] > rmx:
        rs += arr[i+1][0] - arr[i][0]
print(rs)
```

1.7 单调栈

接雨水 26977

```
n = int(input())
mylist = [int(i) for i in input().split()]
total = 0
stack = []
current = 0
while current < n:
    while stack and mylist[stack[-1]] < mylist[current]:
        h = stack.pop()
        if stack:
            total += (min(mylist[stack[-1]],mylist[current]) - mylist[h])*(current-stack[-1]-1
    stack.append(current)
    current += 1
print(total)
```

护林员盖房子

```
#2200015507 王一粟
def solution(mylist,n):
    stack = []
    result = [0 for i in range(n)]
    for idx,element in enumerate(mylist):
        while stack and element < mylist[stack[-1]]:
            a = stack.pop()
            if stack:
                result[a] += idx-stack[-1]-1
            else:
                result[a] += idx-a
        stack.append(idx)
    prev = stack[0]
    result[stack[0]] = n
    for element in stack[1:]:
        if mylist[element] == mylist[prev]:
            result[element] = result[prev]
        else:
            result[element] = n - prev - 1
        prev = element
    return result
m,n = [int(i) for i in input().split()]
total = 0
mylist = [0 for i in range(n)]
for i in range(1,m+1):
    s = [int(i) for i in input().split()]
    for idx,element in enumerate(s):
        if element == 0:
            mylist[idx] += 1
        else:
            mylist[idx] = 0
    process = solution(mylist,n)
    for idx,element in enumerate(process):
        total = max(element*mylist[idx],total)
```

```
print(total)
```

1.8 栈

合法出栈序列

```
#2200015507 王一粟
origin = input()
while True:
    try:
        myorigin = origin
        s = input()
        mylist = []
        if len(myorigin) != len(s):
            print("NO")
            continue
        mylist.append(myorigin[0])
        myorigin = myorigin[1:]
        cnt = 0
        while True:
            if len(mylist) == 0:
                mylist.append(myorigin[0])
                myorigin = myorigin[1:]
            if s[0] == mylist[-1]:
                del mylist[-1]
                s = s[1:]
                cnt = cnt + 1
                if cnt == len(origin):
                    print("YES")
                    break
            else:
                if len(myorigin) == 0:
                    print("NO")
                    break
                else:
                    mylist.append(myorigin[0])
                    myorigin = myorigin[1:]
    except EOFError:
        break
```

前缀求值

```
s = reversed(input().split())
stack = []
cnt = 0
for element in s:
    if element not in "+-*/":
        stack.append(element)
    else:
        op1 = stack.pop()
        op2 = stack.pop()
        stack.append(str(eval(op1+element+op2)))
result = float(stack[0])
print(f"{result:.6f}")
```

2 非常用小算法

1.求最大公因数

```
def gcd(m,n):
    while m%n != 0:
        m,n = n,m%n
    return n
```

2.栈：匹配括号问题

```
def check(s):
    stack,balanced = [],True
    index = 0
    while balanced and index<len(s):
        element = s[index]
        if element in "([{":stack.append(element)
        else:
            if stack and match(stack[-1],element):
                stack.pop()
            else:
                balanced = False
        index += 1
    if balanced and not s:#注意：栈也要为空
        return True
    else:
        return False
def match(left,right):
    if "{([".index(left) == "})]".index(right):
        return True
```

3.栈：进制转换问题

```
def convert(dec_num,base):#考虑了小数与负数情况
    digits = "0123456789ABCDEF"
    sign = 1
    if dec_num == 0:
        return "0"
    if dec_num < 0:
        sign = -1
        dec_num = -dec_num
    stack = []
    while dec_num != 0:
        t = dec_num % base
        stack.append(digits[t])
        dec_num = dec_num // base
    if sign == 1:
        return "".join(reversed(stack))
    else:
        return "-"+"".join(reversed(stack))
```

4.埃氏筛法,得到质数表

```
def judge(number):
    nlist = list(range(1,number+1))
    nlist[0] = 0
    k = 2
    while k * k <= number:
        if nlist[k-1] != 0:
            for i in range(2*k,number+1,k):
                nlist[i-1] = 0
        k += 1
    result = []
    for num in nlist:
        if num != 0:
            result.append(num)
    return result
```

5.卡特兰数

eg:出栈顺序统计

```
#sol1: 递推法
n = int(input())
def f(n):
    result = 0
    if n==0:
        return 1
    for i in range(1,n+1):
        result += f(i-1) * f(n-i)
    return result
print(f(n))

#sol2: 公式法
n = int(input())
import math
result = int(math.comb(2*n,n)/(n+1))
print(result)
```