不管准备的怎么样了，已经是最后一天了，虽然还有海洋大数据的实验3没有完成，但是必须开始复习了。

1. 读取文件

```
with open("fuck.txt","r") as file:
    for line in file:
        print(line.strip())
```

2. 背包问题

```
N,A,B=map(int,input().split())
w=[0]+list(map(int,input().split()))
dp=[[0 for j in range(A+1)] for k in range(B+1)]
ans = 0
for i in range(1,N+1):
    for j in range(B,-1,-1):
        for k in range(A,-1,-1):
            x1,x2=0,0
            if j >= w[i]:
                x1 = max(dp[j][k],dp[j-w[i]][k]+w[i])
            if k >= w[i]:
                x2 = max(dp[j][k],dp[j][k-w[i]]+w[i])
            dp[j][k] = max(dp[j][k],x1,x2)
            ans = max(ans,dp[j][k])
print(ans)
```

3. 快速幂+矩阵快速幂

```
inline void qpow (int T) {
    initans ();//这里是将ans初始化为单位矩阵。

    while (T) {
        if (T & 1) {
            Mul (ans, ans, a);
            //ans=ans*a。
        }

        Mul (a, a, a);
        //a=a*a。

        T >>= 1;
    }
}
```

4. 状压DP

```
for s in range(1<<n):
    for i in range(n):
        if s & (1<<i):
            for j in range(n):
                if s & (1<<j) and g[j][i] == 0:
                    dp[s][i] += dp[s^(1<<i)][j]
```

`dp[s][i]`表示"当前在i城市，已访问城市集合为s"的最短长度。

5. DFS爆搜

- 玩具蛇

  ```
  import os
  import sys

  # 请在此输入您的代码
  ans = 0
  def dfs(nownum, posx, posy, vis, ans, idx):
      if nownum == 16:
          ans[idx] += 1
          return
      for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
          new_x, new_y = posx + dx, posy + dy
          if 0 <= new_x < 4 and 0 <= new_y < 4 and vis[new_x][new_y] == 0:
              vis[new_x][new_y] = 1
              dfs(nownum + 1, new_x, new_y, vis, ans, idx)
              vis[new_x][new_y] = 0  # 回溯


  vis = [[0] * 4 for i in range(4)]
  # 开始搜索
  cnt = 0
  ans = [0 for i in range(17)]  # 用列表存储结果以便传递引用
  for i in range(4):
      for j in range(4):
          vis[i][j] = 1
          dfs(1, i, j, vis, ans,i+1)
          vis[i][j] = 0
  print(sum(ans))
  ```

6. 单调队列

```
from collections import deque

class MonotonicQueue:
    def __init__(self,compare):
        self.compare = compare
        self.mp = deque()
    def push(self,value):
        while self.mp and self.compare(self.mp[-1],value):
            self.mp.pop()
        self.mp.append(value)
    def front(self):
```

```
        return self.mp[0]
    def pop(self,value):
        if self.mp and self.mp[0] == value:
            self.mp.popleft()

n,k = map(int,input().split())
nums = list(map(int,input().split()))
def solve(compare):
    mq = MonotonicQueue(compare)
    ret = []
    for i in range(len(nums)):
        if i>=k:
            mq.pop(nums[i-k])
        mq.push(nums[i])
        if i >= k-1:
            ret.append(mq.front())
    return ret
print(*solve(lambda x,y:x>y))
print(*solve(lambda x,y:x<y))
```

7. 单调栈

```
n=int(input())
arr=[0]+list(map(int,input().split()))
ans=[0]*(n+1)
stack=[]
for i in range(1,n+1):
    while len(stack) != 0 and arr[stack[-1]]<arr[i]:
        ans[stack.pop()]=i
    stack.append(i)
while len(stack) != 0:
    ans[stack.pop()]=0
print(*ans[1:])
```

8. 回文串

```
class Solution:
    def longestPalindrome(self, s: str) -> str:
        def expand(s,left,right):
            while left >=0 and right < len(s) and s[left] == s[right]:
                left -=1
                right += 1
            return left+1,right-1
        a,b=0,0
        for i in range(len(s)):
            l1,r1 = expand(s,i,i)
            l2,r2 = expand(s,i,i+1)
            if r1-l1 > b-a:
                a,b=l1,r1
            if r2-l2 > b-a:
                a,b=l2,r2
        return s[a:b+1]
```

10. 欧拉函数

11. 欧拉筛

```
prims = []
n=1000
vis = [False]*(n+1)
for i in range(2,n+1):
    if not vis[i]:
        prims.append(i)
    j=0
    while i*prims[j] <= n:
        vis[i*prims[j]]=True
        if i % prims[j] == 0:break
        j+=1
```

12. Dijkstra算法

```
def dijkstra(n, src):
    dist = [inf]*n
    vis = [False]*n
    dist[src] = 0

    # 找最短的加入到S中:
    for _ in range(n):
        # 待加入的点为u
        u, min_dist = min([(i, dist[i]) for i in range(n) if not vis[i]],
key=lambda x: x[1])
        vis[u] = True
        for v in range(n):
            if not vis[v]:
                dist[v] = min(dist[v],dist[u] + g.get((u,v),inf))
    return dist
```

13. Floyd算法

```
import copy
n,m=map(int,input().split())
f = [[float('inf')]*(n) for i in range(n)]
# 设置自身到自身的距离为0
for i in range(n):
    f[i][i] = 0
for _ in range(m):
    u,v,w=map(int,input().split())
    f[u-1][v-1] = f[v-1][u-1] = min(w,f[v-1][u-1])
for k in range(n):
    for i in range(n):
        for j in range(n):
            f[i][j] = min(f[i][k]+f[k][j],f[i][j])
for i in range(n):print(*f[i])
```

14. Nim游戏

- 如果先手遇到的奇数台阶上的石子 $a_{2k-1} \oplus a_{2k-3} \oplus a_{2k-5} \ldots\ldots \oplus a_1 = x \neq 0$ 的话先手必赢，否则先手必输。
- 如果先手遇到石子 $a_k \oplus a_{k-1} \oplus a_{k-2} \ldots\ldots \oplus a_1 = x \neq 0$ 的话先手必赢，否则先手必输。

16. 前缀和+差分

$$前缀和 = 上区 + 左区 - 左上区 + 当前格$$

$$S_{i,j} = S_{i-1,j} + S_{i,j-1} - S_{i-1,j-1} + a_{i,j}$$

$$区间和 = 全区 - 上区 - 左区 + 左上区$$

$$S_{ab,ij} = S_{i,j} - S_{a-1,j} - S_{i,b-1} + S_{a-1,b-1}$$

对于一个给定的矩阵 A，它的差分矩阵 B 中 $B_{i,j}$ 可用如下公式计算：

> $$B_{i,j} = A_{i,j} - A_{i-1,j} - A_{i,j-1} + A_{i-1,j-1}, (1 \leq i \leq n)$$

对 A 矩阵中该区块的每个元素都加上 d 的操作等价于进行下面四个操作：

> $$B[x_1, y_1] += d$$
>
> $$B[x_2 + 1, y_1] -= d$$
>
> $$B[x_1, y_2 + 1] -= d$$
>
> $$B[x_2 + 1, y_2 + 1] += d$$