T1

60分做法：

先记录每个数能分解出的 2 和 5 的数目，然后弄个01背包即可 .

用 $dp[i][j][l]$ 表示从前 i 个数中选出 j 个数其中 5 的个数为 l 个的情况下 2 的数目最多是多少 .

初始状态为 $dp[0][0][0]=0$, 推到终态 $dp[n][k][x]$ 即可 . 注意要存在上一种状态才能到达下一种状态 .

100分做法：

开三维会 mle, 可以优化一下空间，去掉 i 这维变成一个二维背包问题

关键代码：

```
void solve(){

    cin >> n >> k;
```

```
    for(int i = 0; i < N; i++){
        for(int j = 0; j < NMAX; j++){
            dp[i][j] = -INF64;
        }
    }

    dp[0][0] = 0;

    for(int l = 1, x; l <= n; l++){
        cin >> x;

        int five = 0, two = 0, m = x;
        while(m % 5 == 0)five++, m /= 5;
        while(x % 2 == 0)two++, x /= 2;

        s += five;

        for(int i = min(k,l); i > 0; i--){
            for(int j = s; j > 0; j--){
                dp[j][i] = max(dp[j][i], dp[j -
five][i - 1] + two);
            }
        }
    }

    int ans = 0;
```

```
    for(int i = 1; i <= s; i++)ans = max(ans,
min(i, dp[i][k]));

    cout << ans << '\n';
}
```

T2

我们让哨卡主动去找宝石，因此可以反向建边，对于i位置如果可以到达i +/- a[i],建立i +/- a[i]指向i 的边；

然后分别从所有的奇数点和偶数点开始广搜两次

关键代码

```
void calc(int x)
{
  queue<int> q;
  for(int i=0;i<n;i++)
  {
    if(arr[i]%2==x) {vis[x][i]=1; q.push(i);}
  }
  while(!q.empty())
```

```
  {
    int c=q.front(); q.pop();
    for(int i:adj[c])//访问c的邻接点，C++11用法，
noip不可用
    {
      if(vis[x][i]) continue;
      vis[x][i]=1;
      dis[x][i]=dis[x][c]+1;
      q.push(i);
    }
  }
}

int main()
{
  cin >> n;
  for(int i=0;i<n;i++)
  {
    cin >> arr[i];
    if(i-arr[i]>=0) adj[i-arr[i]].push_back(i);
    if(i+arr[i]<n) adj[i+arr[i]].push_back(i);
  }
  calc(0); calc(1);
  for(int i=0;i<n;i++)
    cout << (vis[1-arr[i]%2][i]?dis[1-arr[i]%2]
[i]:-1) << " \n"[i==n-1];
}
```

T3

令$dp[i]$ 表示当前gcd 为i,期望几次之后变成1

$$dp[i] = \frac{1}{m}\sum_{j=1}^{m} dp[gcd(i,j)] + 1$$

直接大力递推可以拿到50分。

满分做法：转移的时候原先是枚举所有的j去求最大公约数，现在可以直接枚举新的最大公约数是谁（比如g），然后算一下有多少个j和i的最大公约数为g，对于同一个最大公约数一起求和。

于是我们只需要求$[1,m]$ 中有几个数与i的gcd为g，这个是一个容斥原理的入门题，相当于求$[1,m/g]$里面有多少个数与i/g 互质