T1

1.对于 50% 的数据：$2 \le n \le 10^5$

暴力枚举a，然后求一个最小lcm 的(a,b)

2.对于 100% 的数据：$2 \le n \le 10^9$

结论：答案是 k 和 $n-k$，k 为 n 的最大真因子。

证明：假设$a \le b$,那么最小公倍数取到b是最小的，假设$b = k \times a$, 那么$a(k+1) = n$,所以a是n的一个因子，a越大，b越小

T2

操作可以理解为某个字符往前跳到任意位置

这题关键是要算出有多少的字符可以不用动，令$dp[i][j]$表示s串的前i个字符与t串的前j个字符最长的公共子序列且满足可以不用动，转移类似于LCS的转移，假如$s[i] = t[j]$,那么只要i后面的每一种字符的数量都大于等于j后面的每一种字符的数量，这个就是可以转移的，预处理即可

为什么呢？ 一旦有一个字符少了，后面肯定就无法做到相等，反之，如果都大于等于下面的，肯定可以找到一种方案操作往后让s串等于t串，可以在纸上试试看。

T3

$dp[i][j][k]$表示构造的序列匹配了第一个串的前i个字符跟第二个串的前j个字符,并且左括号的数量比右括号的数量多k的最短长度

转移的时候，枚举放左括号还是右括号，就可以计算出i和j要不要加1，注意时刻都不能让右括号比左括号多。

```
#define ll long long
const int inf=1e9;
const int mod=1e9+7;
const int maxn=2e2+10;
char s[maxn],t[maxn];
int n,m;
int dp[maxn][maxn][2*maxn];
struct ppo{
  int x,y,k;
  char c;
}pre[maxn][maxn][2*maxn];
int main(){
  scanf("%s%s",s,t);
  n=strlen(s);m=strlen(t);
```

```
  for(int i=0;i<=n;i++)
    for(int j=0;j<=m;j++)
      for(int k=0;k<2*maxn;k++)
        dp[i][j][k]=inf;
  dp[0][0][0]=0;
  for(int i=0;i<=n;i++){
    for(int j=0;j<=m;j++){
      for(int k=0;k<2*maxn;k++){
        if(dp[i][j][k]==inf) continue;
        int x=i+(i<n&&s[i]=='(');
        int y=j+(j<m&&t[j]=='(');
        if(k+1<2*maxn&&dp[i][j][k]+1<dp[x][y]
[k+1]){
          dp[x][y][k+1]=dp[i][j][k]+1;
          pre[x][y][k+1]=ppo{i,j,k,'('};
        }
        x=i+(i<n&&s[i]==')');
        y=j+(j<m&&t[j]==')');
        if(k>0&&dp[i][j][k]+1<dp[x][y][k-1]){
          dp[x][y][k-1]=dp[i][j][k]+1;
          pre[x][y][k-1]=ppo{i,j,k,')'};
        }
      }
    }
  }
  string str;
  int x=n,y=m,k=0,p=0;
  for(int i=0;i<2*maxn;i++){
```

```
    if(dp[n][m][p]+p>dp[n][m][i]+i) p=i;
  }
  for(int i=0;i<p;i++) str.pb(')');
  k=p;
  for(int i=0;i<dp[n][m][p];i++){
    ppo p=pre[x][y][k];
    str.pb(p.c);
    x=p.x;y=p.y;k=p.k;
  }
  reverse(str.begin(), str.end());
  cout<<str<<endl;
  return 0;
}
```