A 题题解

算法：贪心+优先队列

对于题目，多举几个例子就不难发现：只要每次从集合里面取出最短的两根香肠，并且把长度为两根香肠之和放入集合即可，最后把所有的答案进行累加即可。如果纯粹的试用贪心算法，时间复杂度是 $O(N^2)$。这里可以配合使用**优先队列（priority_queue）**进行高效实现：按照小顶堆的顺序存放香肠的长度，每次从队列的队首取出两个元素相加得到 sum，并弹出两个队首元素，将 sum 压入队列，并将 sum 加给 ans，最后队列里面没有元素的时候，输出 ans。时间复杂度可以为 O(NlogN)。

B 题题解

算法：区间 DP

dp[i][j]表示前 i**(1<=i<=n)**个数分为 j**(1<=j<=m)**部分的和的乘积的最大值。样例中(5+3)*(3+5) =64。状态转移方程为：

dp[i][j]=max(dp[i][j],dp[k][j-1]*(sum[i]-sum[k]))

其中，sum[i]-sum[k]表示区间(k,i]的累加和!

核心代码：

```
for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
          for(int k=j-1;k<i;k++)
          {
             dp[i][j]=max(dp[i][j],dp[k][j-1]*(sum[i]-sum[k]));
          }
```

C 题题解

算法：”二分“字符串

第一步：判断前 k 个字母中有多少个字母没有包含在原字符串 s 中，把不包含的字母按照字典序由大到小的顺序放在一个字符数组 rst 里面，并记录个数 cnt；

第二步：从 s 中开始“二分”放字母。找到 s 的中间位置，左边的下标给 L，右边的下标给 R。

（1）如果 in[l]!='?' && in[r]!='?' && in[l]!=in[r]，那么说明 s 必不对称，输出"IMPOSSIBLE"；

（2）如果 in[l]=='?' && in[r]=='?'，就往里面按字典序从大到小放 rst[t]（t 初始值为 0）---in[l]=in[r]=rst[t++];如果放满了 cnt 个字母，则放字母 a;

（3）除了上面两种情况，如果 in[l]=='?'，则 in[l]=in[r]；

（4）除了上面三种情况，in[r]=in[l]。

（5）每处理一次，l--,r++，直到 l=0。

第三步：如果最后 cnt 个字母没有放完，输出"IMPOSSIBLE"；反之，输出填满后的字符串 s。