解题报告

“腾讯杯”第二十二届中山大学程序设计比赛

I. Enlarge it

- 题意：输入一个字符矩阵，把它放大k倍输出。
- 模拟即可。

B. Triangle

- 题意：给你n根棍子，问是否有其中三根能组成三角形
- 如果写得不小心，C/C++/java使用int可能会溢出
- 对输入数据从小到大排序后得到数列$\{a_n\}$
- 输出YES，当且仅当存在i使得$a_i + a_{i+1} > a_{i+2}$
- 时间复杂度$O(n \log n)$，但是排序所有数据可能会超时，因此需要分情况处理
- 对于n较小的情况(例如$n < 100$)，按上面做法
- 对于n较大的情况，直接输出YES，后面将给出定理
- 时间复杂度$O(n)$

B. (Con. 1)

- 定理：给定一个非空的由正整数构成的多重集合A，若A中元素不能组成三角形三边，则A中的最大元素$\max A \geq F_{|A|}$，其中$F_{|A|}$是斐波那契数列的第n项
- 证明略
- F_n以指数级增加，例如$F_{100} = 354224848179261915075 > 2^{68}$
- F_{100}远远大于最长的棍子长度
- 因此，只要$|A| \geq 100$，必然有三根棍子可以组成三角形

E. Coding problem

- 题意：将字符串每个字符ASCII的二进制翻转码组成的字符串以每6个一组输出。
- 题目很简单，看题要注意数组不要开小了。
- #define p (i*8+j)/6
- int n=strlen(s);
- for(int i=0;i<n;i++)
- for(int j=0;j<8;j++,s[i]>>=1) a[p]=(a[p]<<1)+(s[i]&1);

H. Clumsy Keke

- 题意：给你一个立方块堆的三视图，求出它的体积(若有多个满足，则求最大的那个)。
- $m_x, m_y, m_z < 100$，立方块范围很小
- 可以直接枚举坐标(x, y, z)
- 根据三视图判断该坐标是否可能有立方块
- 若有则Ans加1
- 可以证明Ans为最大的立方体堆的体积
- 时间复杂度$O(n^3)$

D. Monitor

- 题意：在一个面积不超过n*m的矩形上，有p个矩形A，问之后的q个矩形B能否被之前的A全部覆盖（每个B的点都在至少一个A中）
- 由于n*m,p,q的范围过大，于是考虑O(n*m+p+q)的做法。
- 对于A类矩形(x1,y1,x2,y2)，我们只需要在(x1,y1),(x2+1,y2+1)处+1，在（x1,y2+1）(x2+1,y1)处-1
- 之后对整个面积求一个前缀和。则大于0的地方就是被A类矩形覆盖的点。
- 把值大于0的地方变成1，再一次求一次前缀和，处理好后即可在O(1)的时间算出一个矩形内被覆盖的点的数量。

A. Min-Max

- 题意：有一个长度为n的随机排列以及m个min、max操作。问最后一个操作的结果的期望 * n! 的结果。
- 样例：
- 3
- min a 2 a 3
- max b 1 a 1

- (1, 2, 3) -> 2
- (1, 3, 2) -> 2
- (2, 1, 3) -> 2
- (2, 3, 1) -> 2
- (3, 1, 2) -> 3
- (3, 2, 1) -> 3
- 所以结果为14

A. (Con. 1)
解释

- 枚举k，考虑计算结果 >= k 的排列有几个。
- 此时数字本质上只有两类，>= k 的以及 < k的。可以2^n的枚举每个位置的数是>=k还是<k，然后对于每一个状态模拟一遍所有m个操作，如果结果为1，则说明该状态是可行的。
- 接下来考虑每种状态对应几个排列，若该状态中有x个1，则对应的排列个数为 x! * (n-x)! 个。现在已经计算出结果 >= k 的排列有 f[k] 个，那么答案也就是所有 f[k] 的和。整体复杂度为 O((2^n) * m)

C. Reverse it

- 题意：涛涛有一个n行m列的卡牌阵，每个位置是0或1，0代表卡牌背面朝上，1代表卡牌正面朝下，涛涛每次可以选择一个小矩阵，把这个矩阵内所有卡牌翻转，问最后可以得到多少本质不同的卡牌阵。

C. (Con. 1)

- 题解：
- 先计算出取一个矩形的方案是one= $C_n^2 * C_m^2$
- 再计算出取两个不同矩形的方案是two= C_{one}^2
- 再减去重复的方案
- 考虑4种会重复的形状，算出相应系数

有
空
有

有
空

空	有
有	有

空	有
有	空

K. Party

- 题意：有n个人，开m次派对，每次派对每对人会相互认识对方。对于每一次派对，我们要求出新认识的pair的数量。
- 本意：只允许O(n log(n)+mlog(n))通过
- 因为是算数量，所以可以把相互认识当作有向边，每个边当作从小连向大的有向边即可
- 对于每一个i，我们只用记录它在每次派对数量的数量认识的最大值f[i]即可。因为一旦i认识了f[i]，那么说明它认识了i到f[i]的全部人。
- 用线段树来维护一段区间上f值的max和sum即可
- 我们可以在线段树上二分来用logn的效率来计算不需要再连边的人。

F. Network

- 题意：给一棵边带权的树，定义两个点之间的距离为路径上边权的 max - min。一开始树上有一些黑点，其余的为白点，对于每个白点计算出距离它最远的黑点到它的距离 dis[u]，现在要求再添加一个黑点，使得dis数组的最小值最大。

F. (Con. 1)

- 首先需要计算出距离每个白点最远的黑点到它的距离max_dis。将所有点按照这个距离从小到大排序得到一个有序序列p[1..n]。接下来枚举一个将要变成黑点的白点，并且从小到大枚举答案Ans，并加入所有max_dis <= Ans的白点，查询当前点到所有已经加入白点的最小距离，若该值大于Ans 则说明Ans可以扩大。由于Ans 只会从1到n扩大一次，所以整体上的复杂度是线性的。
- 求最大距离和最小距离是经典问题，可以直接利用点分治解决。
- 复杂度：$O(n \log^2 n)$

G. Bring Bring big teacher brother

- 题意：求至多对给定串修改一个字符情况下所有的中点不重复的对称的回文的子串的平方的和最大是多少。
- 总体思路：计算出把第i个字符变成字符j的收益contr[i][j]。以及把第i个字符更改导致的损失del[i]。
- 字符串是对称的，每次倍增考虑f[i][j]，即第j字符是否是长为$2i-1$特殊串的中心。如果是则f[i][j]为1，不是为0.
- 现在分类讨论这个以j为中心，长度为2^i-1的字符串。
- 定义左串是其左边(2^(i-1)-1)的串，右串同理。

G. (Con. 1)

- 1.该串是特殊串
  - a.改任意一个非中心的字符会导致len^2的损失。对del打区间加标记。
  - b.改中心字符可能会导致损失，这取决于改成哪个字符，
    假设将中心字符改成字符k会导致重复，则contr[j][k]-=len^2
- 2.该串差一个字符是特殊串
  - a.左右串符合要求，中心字符因重复不符合要求
    - 直接算contr[j][k]
  - b.左右串存在某一位不相等，且左串或右串存在一个串是特殊串
    - 非特殊串的错字符和中心字符相等：纠正非特殊串的坏字符有贡献。
    - 特殊串的错字符和中心字符相等：纠正非特殊串坏字符无贡献。
    - 左右串的错字符均与中心字符不等：纠正非特殊串坏字符有贡献
  - 答案=原串不改动的贡献-del[i]+contr[i][j]的最大值

G. (Con. 2)

- 答案=原串不改动的贡献-del[i]+contr[i][j]的最大值

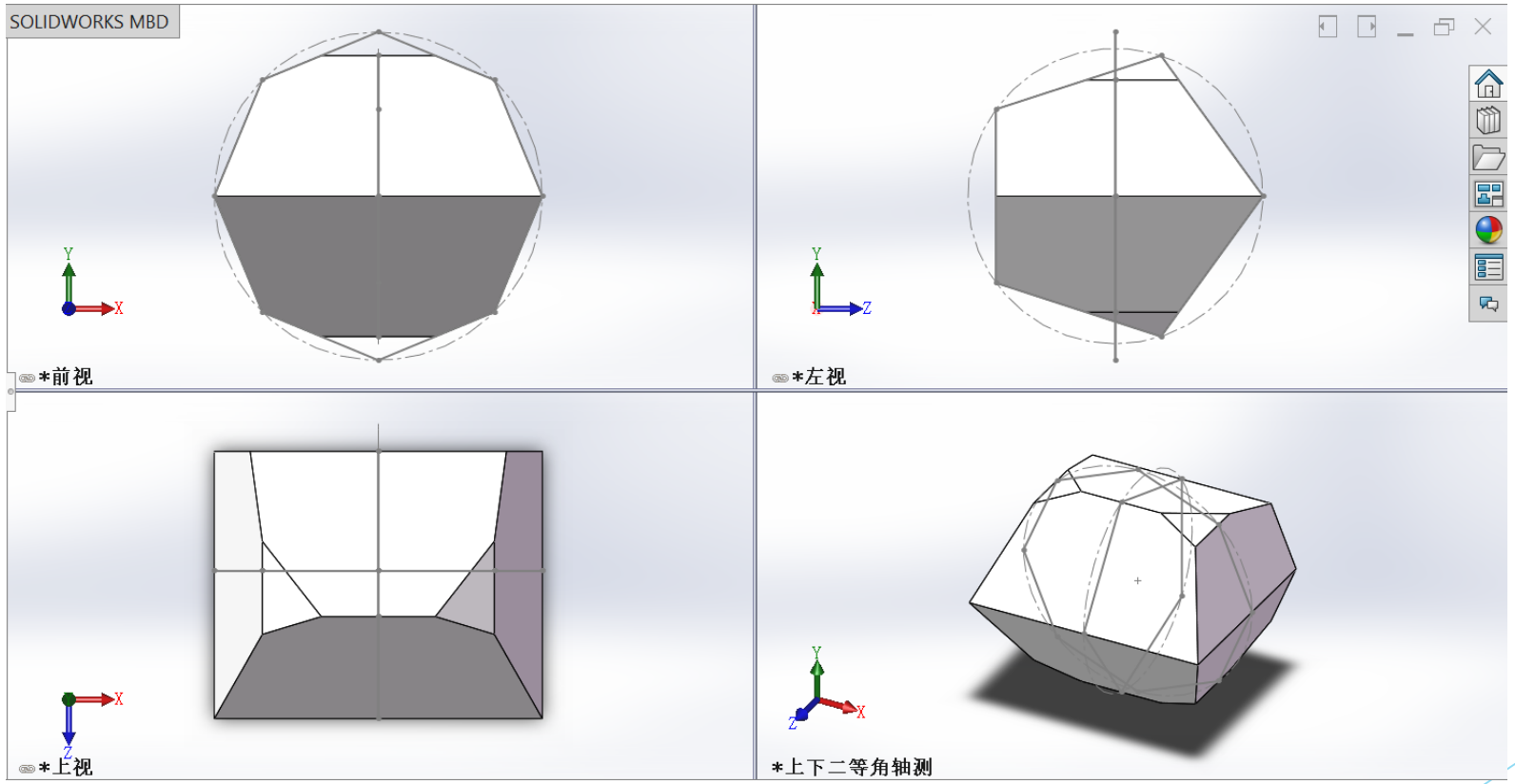
J. Silly Keke

- 题意：给你一个零件，它是由两个无限长的凸直棱柱A和B相交部分组成的，A的母线与B的母线垂直，求该零件的体积
- 注意：
  - 输入的边的斜率(的绝对值)有可能非常大，甚至为无穷
  - 输入的边的斜率(的绝对值)有可能非常小，甚至等于零
  - 此题精度要求非常高，绝对误差10^{-3} ≈相对误差10^{-10}，即10位有效数字
- 因为输入的点的坐标按顺序给出，因此不用排序，总时间复杂度$O(n)$
- 但是为了降低难度，$O(n \log n)$也能通过
- 计算方法将在后面描述

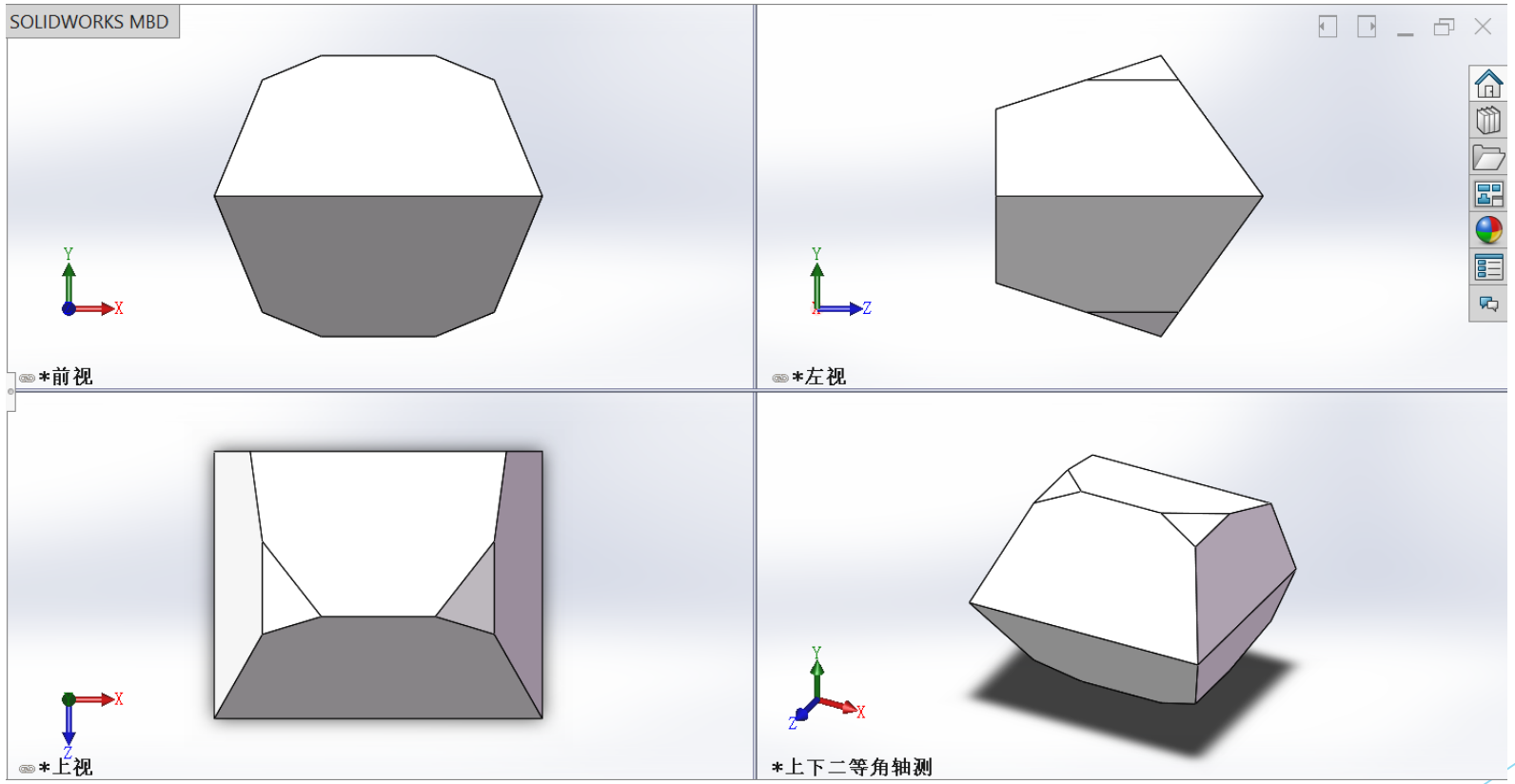
J. (Con. 1)

- 计算分为三步：
- 可以证明零件纵截面总是矩形(后面将举例说明)，因此可以将零件分层计算
- 零件的每层是一个梯台(侧视图和正视图都是梯形，上下底面都是矩形)
- 梯台体积：简单的积分，或者祖氏原理+割补法
  - $V_{梯台} = \frac{1}{6}h\left(a_{上}b_{上} + \left(a_{上} + a_{下}\right)\left(b_{上} + b_{下}\right) + a_{下}b_{下}\right)$
    其中a_i, b_i是底面i的长和宽，h是梯台的高
- 事实上，等价于辛普森公式
  - $V_{梯台} = \frac{1}{6}h\left(S_{上} + 4S_{中} + S_{下}\right)$
  - 其中S_i是截面i的面积

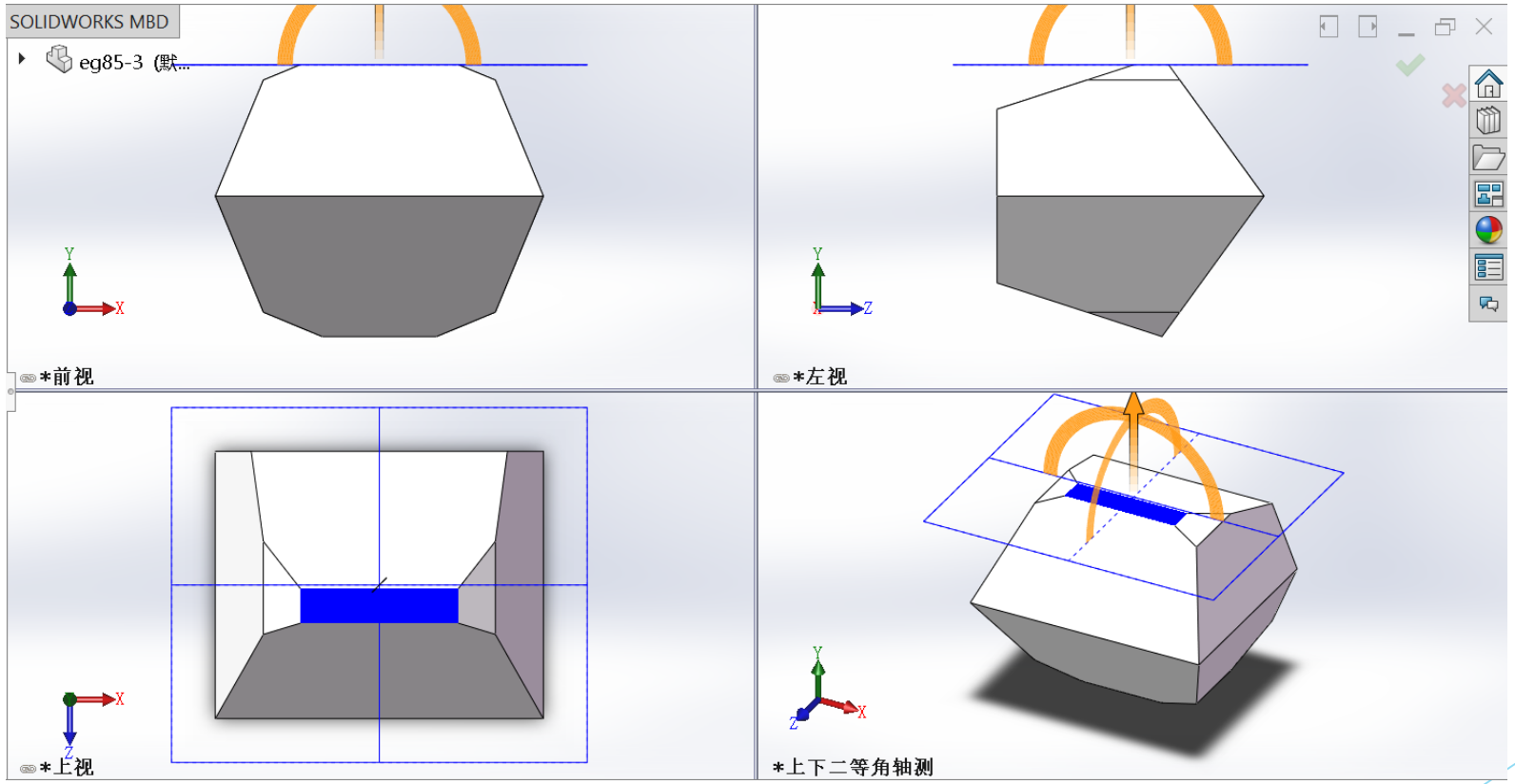
J. (Con. 2)
例子：正八边形+正五边形



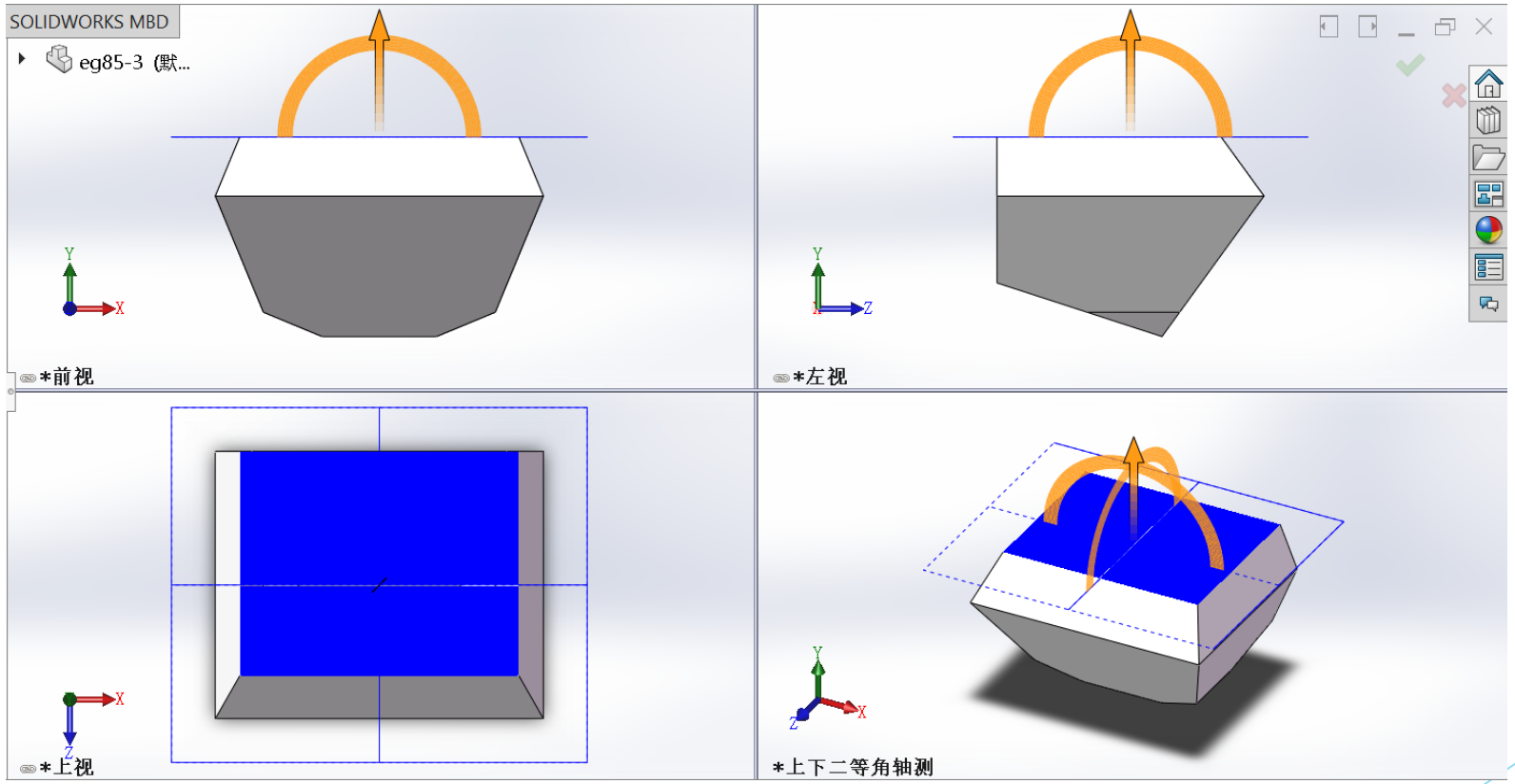
J. (Con. 3)
例子：正八边形+正五边形



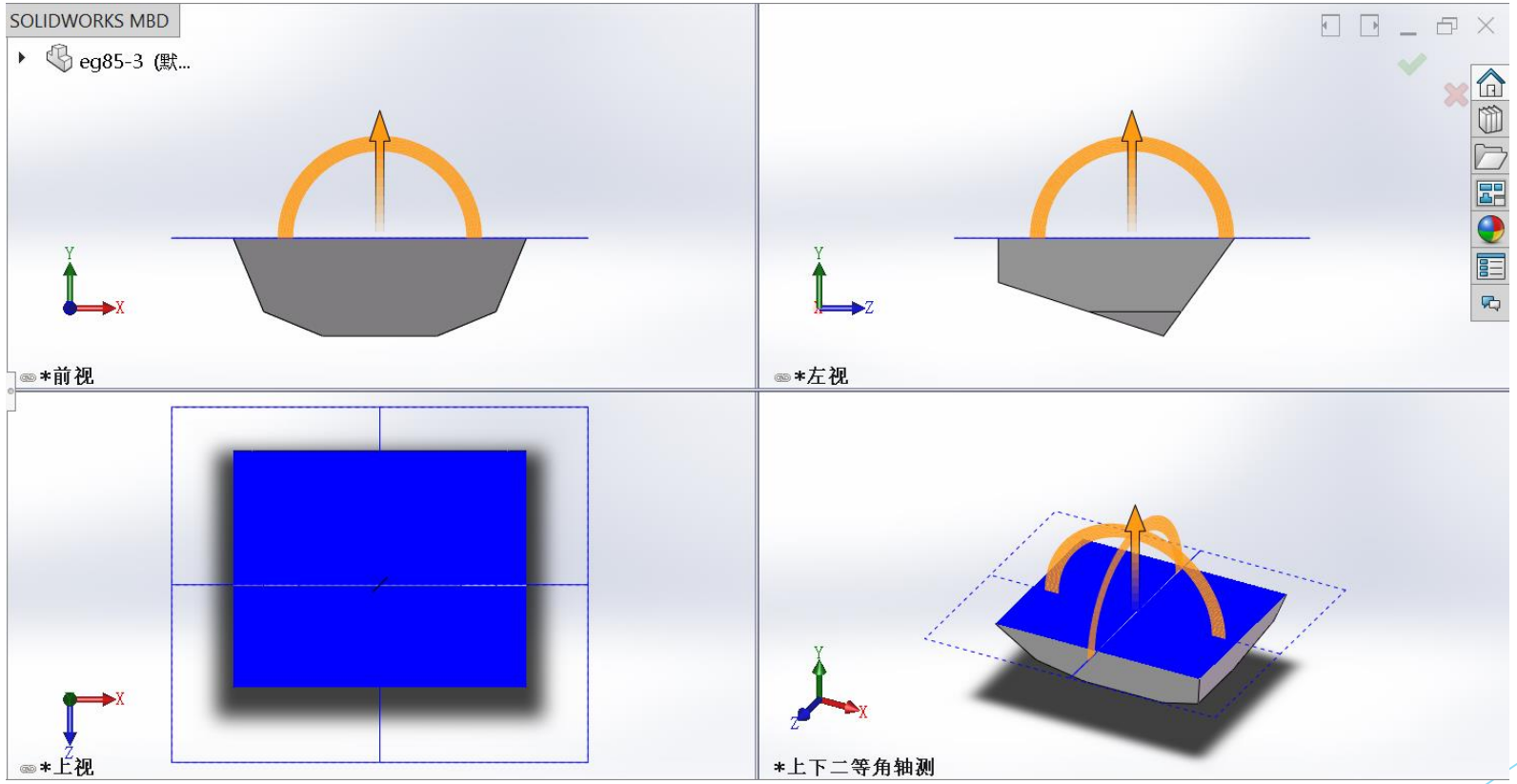
J. (Con. 4)
零件纵截面总是矩形



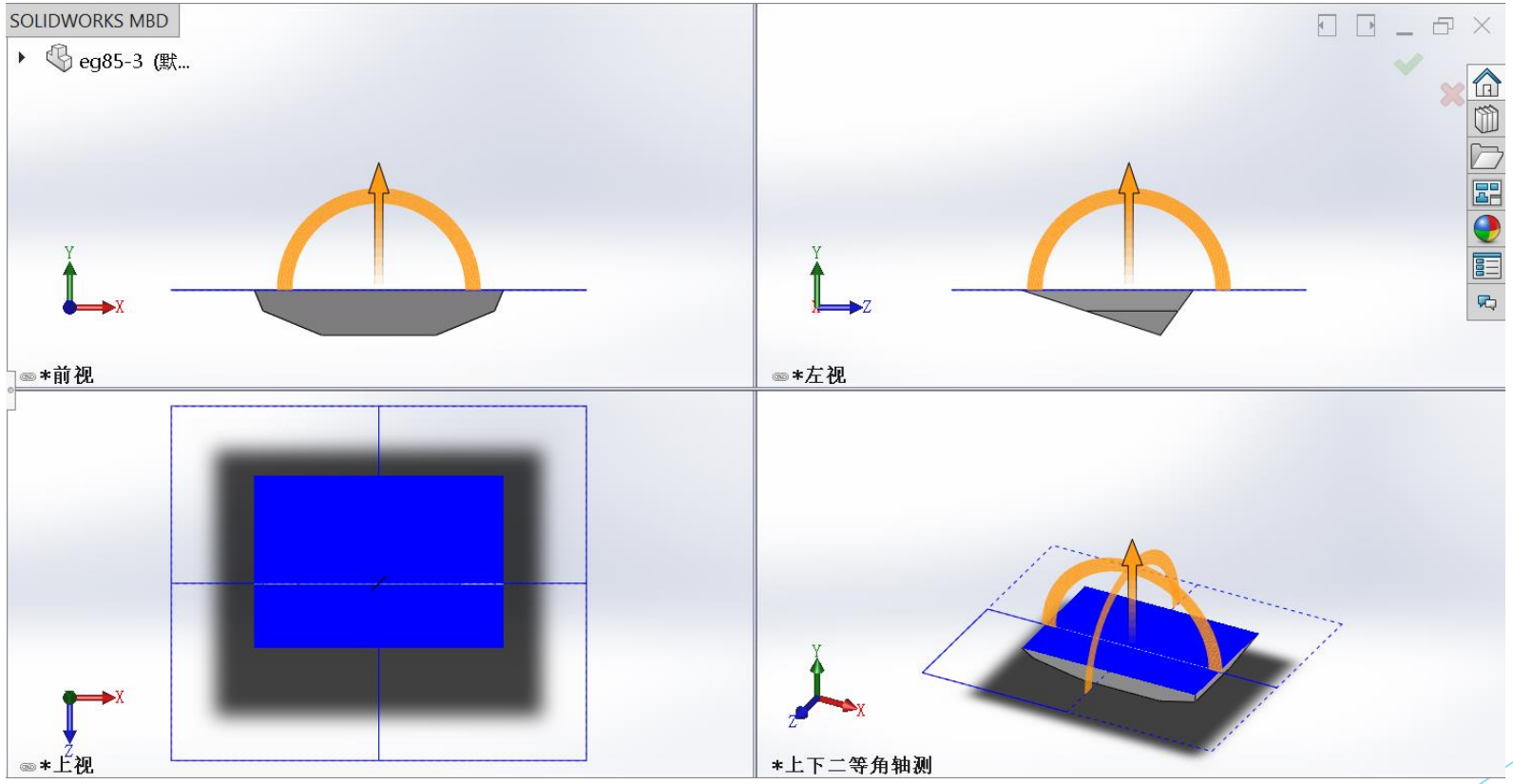
J. (Con. 5)
零件纵截面总是矩形



J. (Con. 6)
零件纵截面总是矩形



J. (Con. 7)
零件纵截面总是矩形



J. (Con. 8)
将零件分层计算