**PartB**

PartB要求优化矩阵转置过程中的Cache访问，Cache结构是固定的，需要根据不同形状的矩阵来设计方案，尽可能地减少未命中地次数，基本思路是分块处理。

* 32\*32

Cache有32组，每组能存放8个int，按行存储的情况下这个Cache最多能存数组的前八行，因此按8\*8分块较为合适。直接采用8\*8分块即可使未命中次数降低到300+，距离满分还有一定的距离。

仔细考虑，在进行对角元素的转置时，A和B访问的行数相同，两者对应的缓存位置也相同，访问A的行之后，访问B的行时会将A的行移除，之后再次访问A又会将A移入，从而产生两次未命中，因此可以将A的行中的八个元素一次读出，之后访问B的行产生一次未命中，此时将读出的八个元素依次写入即可。这样可以将在对对角元素转置时产生的两次未命中减少为一次，理论上可以减少32次。最终未命中次数低于300达到了满分标准。

* 64\*64

64\*64的矩阵在Cache中只能存4行，采用8\*8的分块，后四行分块不能存到Cache中，实践可知效率很低。采用4\*4的分块，一个分块并不能占满Cache，没能充分利用Cache的空间，未命中次数可以降低到1700，离满分1300还有很远的距离。参考别人的方法，可以先采用8\*8的分块，在8\*8内部再做处理减少未命中，步骤如下：

首先对A的上半部分进行访问，上半部分可以完全装入Cache，因此不会发生冲突。之后将A的左半部分照常读入B的左上部分。如果此时将A的右上部分也正常读入B的左下部分，则B的读入会产生冲突，因为缓存中同样只能装入B的上半部分，所以B的上半部分和下半部分同样冲突。此时，我们可以先将A右半部分暂存在B的右半部分中，避免冲突。

下一步把要B的右上部分移到左下部分，把A的左下部分移到右上部分。细分来说，先将B右上第一行读出到临时变量中，再将A的左下第一列读出，之后将A左下第一列放到B的右上第一行中，将B的右上第一行读入到B的左下第一行中，因为是先读了B的右上之后又向其中写入，在写入的过程中必然会命中，从而减少了未命中。

最后之间完成右下角的转置，即可完成一个8\*8分块的转置，最终的未命中次数低于1300，达到了满分标准。

* 61\*67

61\*67并非规则矩阵，较难找到规律从而减少Cache的未命中，因此直接采用分块即可，分块为17的时候可将未命中次数降到2000以下，达到了满分标准。