循环以步长 n 扫描数组 B 的一列。因为 n 很大，每次对数组 B 的访问都会不命中，所以每次迭代总共会有 1.25 次不命中。

矩阵乘法版本（类）	每次迭代					
	加载次数	存储次数	A未命中次数	B未命中次数	C未命中次数	未命中总次数
ijk & *jik* (*AB*)	2	0	0.25	1.00	0.00	1.25
jki & *kji* (*AC*)	2	1	1.00	0.00	1.00	2.00
kij & *ikj* (*BC*)	2	1	0.00	0.25	0.25	0.50

图 6-45　矩阵乘法内循环的分析。6 个版本分为 3 个等价类，用内循环中访问的数组对来表示

类 AC 例程的内循环（图 6-44c 和图 6-44d）有一些问题。每次迭代执行两个加载和一个存储（相对于类 AB 例程，它们执行 2 个加载而没有存储）。内循环以步长 n 扫描 A 和 C 的列。结果是每次加载都会不命中，所以每次迭代总共有两个不命中。注意，与类 AB 例程相比，交换循环降低了空间局部性。

BC 例程（图 6-44e 和图 6-44f）展示了一个很有趣的折中：使用了两个加载和一个存储，它们比 AB 例程多需要一个内存操作。另一方面，因为内循环以步长为 1 的访问模式按行扫描 B 和 C，每次迭代每个数组上的不命中率只有 0.25 次不命中，所以每次迭代总共有 0.50 个不命中。

图 6-46 小结了一个 Core i7 系统上矩阵乘法各个版本的性能。这个图画出了测量出的每次内循环迭代所需的 CPU 周期数作为数组大小（n）的函数。

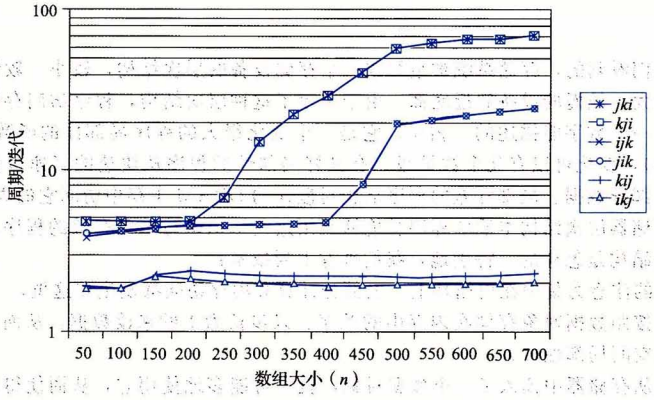

图 6-46　Core i7 矩阵乘法性能

对于这幅图有很多有意思的地方值得注意：

- 对于大的 n 值，即使每个版本都执行相同数量的浮点算术操作，最快的版本比最慢的版本运行得快几乎 40 倍。
- 每次迭代内存引用和不命中数量都相同的一对版本，有大致相同的测量性能。
- 内存行为最糟糕的两个版本，就每次迭代的访问数量和不命中数量而言，明显地比其他 4 个版本运行得慢，其他 4 个版本有较少的不命中次数或者较少的访问次数，或者兼而有之。
- 在这个情况中，与内存访问总数相比，不命中率是一个更好的性能预测指标。例