习题 1.1

```
for (my_i = my_first_i my_i < my_last_i; ++my_i)
{
    my_x = Compute_next_value(...);
    my_sum += my_x;
}
```

显然，除了最后一个核以外，其余核区间长度为$\lceil \frac{n}{p} \rceil$，所以计算元素的分配情况如下：

对每个核，有：my_first_i=(i-1)* $\lceil \frac{n}{p} \rceil$.

除了最后一个核，有：my_last_i=i*$\lceil \frac{n}{p} \rceil$.

对最后一个核，有：my_last_i=n-1.

习题 1.6

A．0 号核接受其余 p−1 个核的结果并把它们相加，因此执行了 p−1 次接受操作和 p−1 次加法操作。

B．0 号核一直和相邻的结果相加，因为树形结构有$\lceil \log_2 N+1 \rceil$层，所以执行了$\lceil \log_2 N \rceil$次接受操作和加法操作。

C．如下表所示

核数量	A 算法	B 算法
2	1	1
4	3	2
8	7	3
16	15	4
32	31	5
64	63	6
128	127	7
256	255	8
512	511	9
1024	1023	10

习题 2.2

队列的特点是队尾插入，队首删除。可以只将队首和队尾元素放进缓存，仅在插入或删除时更新缓存。这样就可以尽量避免频繁操作主存，达到提高写直达高速缓存的性能的目的。

习题 2.3

```
double A[MAX][MAX], x[MAX], y[MAX];
/* First pair of loops */
for (i = 0; i < MAX; ++i)
    for (j = 0; j < MAX; ++j)
        y[i] += A[i][j] * x[j];
/* Second pair of loops */
for (j = 0; j < MAX; ++j)
```

```
    for (i = 0; i < MAX; ++i)
        y[i] += A[i][j] * x[j];
```

两对循环的代码如上。

显然，缓存变大会增加两对循环的性能，其中第一对循环的性能提升要比第二对循环的性能提升更加明显：因为第一对循环对 A 的操作的内存地址是相邻的，而第二对则是跳跃的；所以缓存变大之后第一对循环从主存中读取数据的次数变少了，性能提升也就更优。

同理可得，更大的矩阵会显著降低第一对循环的性能，而对第二对循环的性能影响较小（与前者形成相对关系）。

MAX=8 时，第一个循环一行有两次缺失的发生，所以 2*8=16 次；第二个循环一列有八次缺失的发生，所以 8*8=64 次

习题 2.16

A．

```
#include<stdio.h>
int main()
{
    int n=0;
    scanf("%d",&n);
    int mat_a[n][n],mat_b[n][n];
    int i=0,j=0;
    for(;i<n;i++)
    {
        for(;j<n;j++)
        {
            mat_a[i][j]+=mat_b[i][j];
        }
    }
    system("pause");
    return 0;
}
```

如上面代码所示，上述程序就是符合题意运行时间的程序。加速比为$T_{串行}/T_{并行}=1/p+n^2/\log_2(p)$

当 p 增加、n 保持恒定时，加速比先增加后降低，效率降低；

当 p 保持恒定、n 增加时，加速比增加，效率增高。

B．

效率 $E=\frac{T_{串行}}{pT_{并行}}=\frac{1}{1+p\frac{T_{开销}}{T_{串行}}}$；因为 p 固定，所以 E 的大小由$\frac{T_{开销}}{T_{串行}}$决定。因此，

·如果$T_{开销}$比$T_{串行}$增长得慢，随着问题规模的增加，并行效率将增加。

·如果$T_{开销}$比$T_{串行}$增长得快，随着问题规模的增加，并行效率将降低。

习题 2.17

在程序进行并行运行的时候，某些资源可以通过线程或者进程进行共享处理，而当这些资源在串行运行时无法进行共享，就有可能会被程序进行多次加载从而导致资源浪费。因此并行运行该部分资源时就可以克服资源限制，并获得大于 p 的加速比。

习题 2.19

效率 $E=\frac{T_{串行}}{pT_{并行}}=\frac{1}{1+p\frac{T_{开销}}{T_{串行}}}=\frac{1}{1+p\frac{\log 2p}{n}}$，要让效率 E 恒定，那么$p\frac{\log 2p}{n}$的值应该为常数 c。当 p 变成 kp 时，n 变成 n'，则有$p\frac{\log 2p}{n}=kp\frac{\log 2kp}{n'}$，解得 $n'=n*k*\frac{\log 2kp}{\log 2p}$，所以，n 的增长速率为：$k*\frac{\log 2kp}{\log 2p}$。

当 p 的值从 8→16 时，k=2，代入公式得，n 得增长速率为：$\frac{8}{3}$.

同时，该并行程序是可扩展的，因为对于任意的 n，都能找到对应的 p。

习题 2.20

是强可扩展的。理由：可以获得线性加速比的程序的问题规模可以认为是常数 c。因此我们只需要响应地增加线程或者进程地数量, 就能让线性加速后地程序通过相同的效率解决问题而不用同时增加问题的大小。