习题1.1

用非负整数0，1，…，p-1来标识各个核，对k号核

若n能被p整除，平均每个核处理n/p个计算元素，故

```
my_first_i=(n/p)*k;
my_last_i =(n/p)*(k+1);
```

若n不能被p整除，则前p-1个核每个核处理$\lceil n/p \rceil$个元素，最后一个核处理剩下的元素

```
my_first_i=⌈n/p⌉*k;
if(k<p-1)
    my_last_i =⌈n/p⌉*(k+1)；
else
    my_last_i =n；
```

第二种情况的解法仍适用于第一种情况。

但上述算法会使第0号核和第p-1号核处理元素至多有n/p的数目差异，改进方法如下：

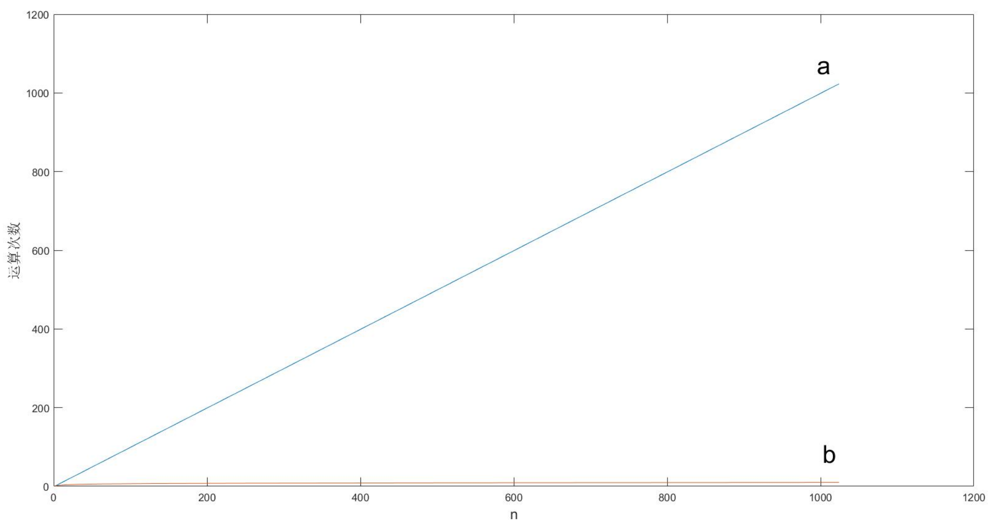
```
rem=n%p;
my_first_i=⌈n/p⌉*k;
if(k<rem)
    my_last_i=⌈n/p⌉*(k+1)；
else
    my_last_i=⌈n/p⌉*(k+1)-1；
```

习题1.6

a. $n-1$

b. $\log_2 n$

表如下所示，可以直观的看出b的运算次数远比a的运算次数少



习题2.2

写直达有一个特性就是，当CPU想Cache写数据时，高速缓存行会立即写入主存中，而将访问主存所花费的时间是远大于CPU的每条指令的运行时间的，因为高速缓存的速度要比主存块很多，若每次都在此阻塞将会导致极大的时间上的花销，在CPU硬件里设置一个队列，将要写入主存的数据入队，随着时间推移，队列中的数据会依次被写入主存中，且只要队列不满，CPU不会因此而阻塞，从而提高了写直达高速缓存的性能。

习题2.3

当矩阵更大时，两个嵌套循环的缺失次数都会增加，但第一个嵌套循环的缺失次数以线性的速度增加，第二个嵌套循环以二次方的速度增加。

当缓存增大时，第一个嵌套循环的缺失次数会减少，但在该Cache的写入规则不变的情况下（即每次缺失后将一行写入），第二个嵌套循环的缺失次数并不会减少。

假设一个高速缓存行可以存放4个元素，第一个嵌套循环每行有2次缺失，共缺失$2 \times 8 = 16$次，第二个嵌套循环每行有8次缺失，共缺失$8 \times 8 = 64$次。

习题2.16

a.

这里n是问题规模，而p则是核数，根据公式

$$S = \frac{T_{串行}}{T_{并行}}$$

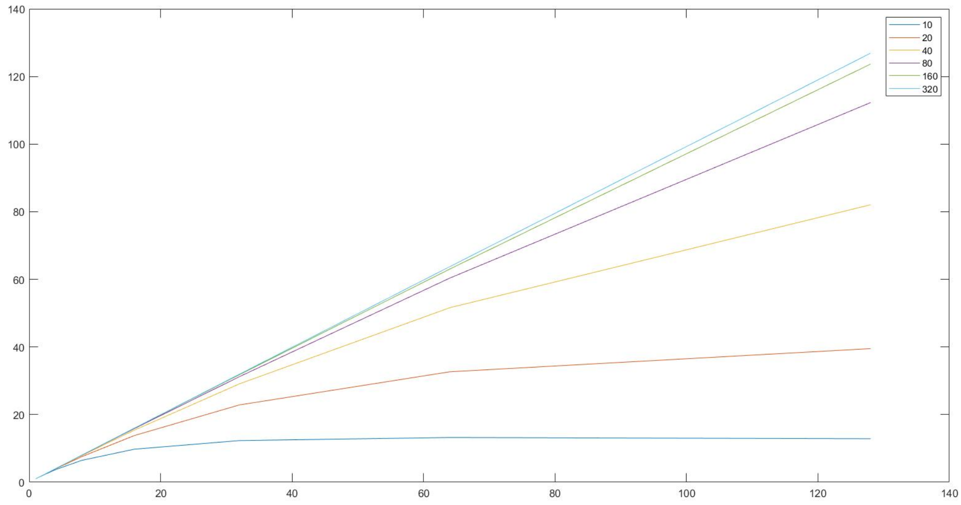
$$E = \frac{T_{串行}}{p \cdot T_{并行}}$$

matlab程序如下：

```
for i=0:5
    n=2^i*10;
    for j=1:7
        p=2^j;
        Ts=n^2;
        Tp=n^2/p+log2(p);
        S=Ts/Tp;
        E=S/p;
        display(n);
        display(p);
        display(S);
        display(E);
    end
end

for j=1:7
    p=2^j;
    for i=0:5
        n=2^i*10;
        Ts=n^2;
        Tp=n^2/p+log2(p);
        S=Ts/Tp;
        E=S/p;
```

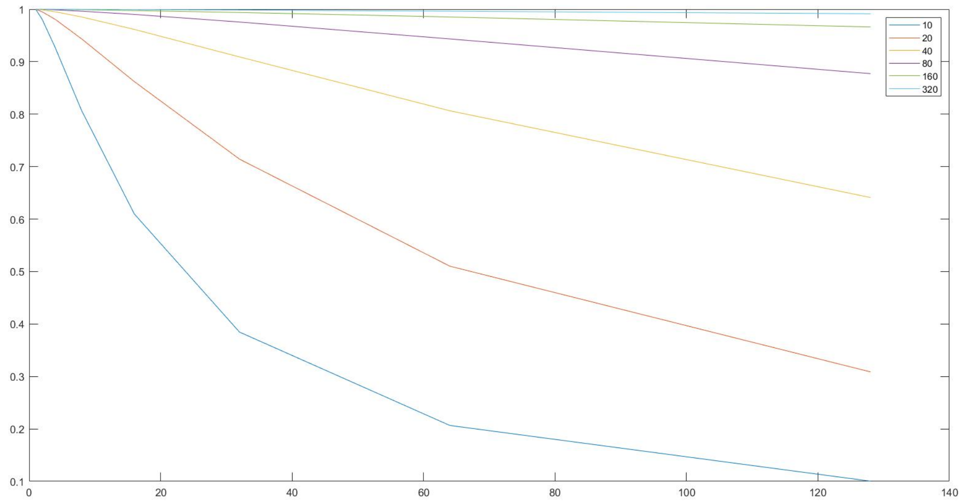
```
        display(n);
        display(p);
        display(S);
        display(E);
    end
end

```

当 p 增加、 n 保持恒定时， 加速比变化如下：



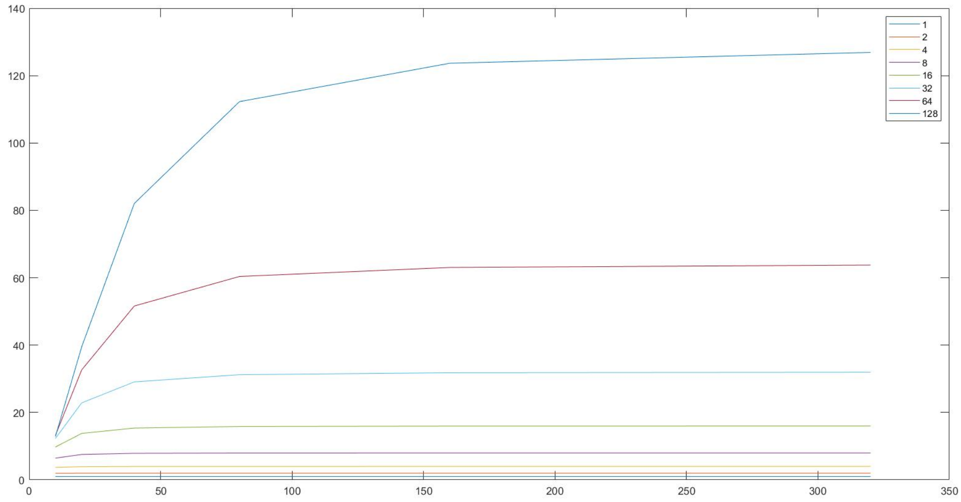
由该图可以看出，加速比随核数p增加而增大，核数增加对规模较小的程序影响不大，对规模较大的程序加速比有显著提升，但规模达到一定程度时加速比增大的速度不明显；

当 p 增加、 n 保持恒定时， 效率变化如下：



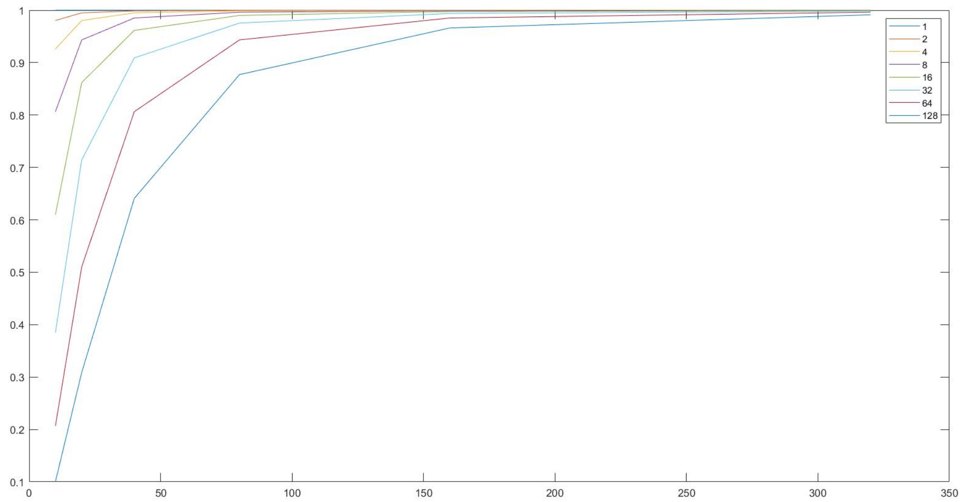
由该图可以看出，效率随核数p增加而减小，对中小规模的程序减小速度较大，对大规模的程序减小速度较慢，甚至趋近于不变；

当 p 保持恒定而 n 增加时，加速比变化如下：



由该图可以看出，当核数较少时，加速比几乎不随着程序规模增大而增大，当核数较多时，加速比随着程序规模增大而增大，且增大到一定程度后保持不变；

当 p 保持恒定而 n 增加时，效率变化如下：



由该图可以看出，效率随问题规模增大而提升，核数越多效率的提升效果越明显。

b. 由a中式子可知

$$\begin{aligned} E &= \frac{T_{串行}}{p \cdot T_{并行}} \\ &= \frac{T_{串行}}{T_{串行} + p \cdot T_{开销}} \\ &= \frac{1}{1 + p \cdot T_{开销} / T_{串行}} \end{aligned}$$

由该式子可知，若$T_{开销}$比$T_{串行}$的增长得慢，则$T_{开销}/T_{串行}$减小，故效率增加；

反之，效率减小。

习题2.17

当程序规模较大时，若该程序读取数据时都能在相关联的Cache中命中，这时并行程序比串行程序在访问内存时Cache缺失次数少，因此该程序有可能克服资源限制，拥有大于p的加速比。

习题2.19

由习题2.16可得如下式子

$$E = \frac{n}{n + p \cdot log_2(p)}$$

其中n是问题规模，p是核数

设问题规模增加m倍，若效率不变，则有下式成立

$$\frac{n}{n + p \cdot log_2(p)} = \frac{mn}{mn + kp \cdot log_2(p)}$$

故有n的增长速度如下

$$m = \frac{k \cdot log_2(kp)}{log_2(p)}$$

由题意可知p=8，k=2，因此

$$m = \frac{2 \times log_2(2 \times 8)}{log_2(8)} = \frac{8}{3}$$

故n应增加$8/3$倍；由于问题规模增大的倍率和进程数增加的倍率不同，故该程序不是可扩展的。

习题2.20

一个可以获得线性加速比的程序是强扩展的，无论进程数为多少，它的效率都是1，若问题规模为n，进程数为p，它的效率为1，若进程数增加到q（q>p)，它的效率也还是1.