HW5

資工三 曹O萱 409410082

(1)使用 time 函數得到的「運算時間各為多少」，例如：real、user、sys各為多少。並說明real、user、sys的意義

下面是我用全部CPU分別測試精確到第5,6,7,8位的結果

```
shiwulo@vm:~/HW/hw5$ time ./pi 8
numCPU:8
Pi = 3.14159265

real    0m18.640s
user    2m14.577s
sys     0m0.171s
shiwulo@vm:~/HW/hw5$ time ./pi 7
numCPU:8
Pi = 3.1415927

real    0m1.299s
user    0m8.829s
sys     0m0.013s
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:8
Pi = 3.141593

real    0m1.225s
user    0m8.724s
sys     0m0.025s
shiwulo@vm:~/HW/hw5$ time ./pi 5
numCPU:8
Pi = 3.14159

real    0m1.237s
user    0m8.463s
sys     0m0.053s
shiwulo@vm:~/HW/hw5$
```

real time:程式從開始執行到結束終止所需要的時間

user time:表示程式在user mode所佔用的CPU時間總和(多核心的CPU或多顆CPU計算時，會將每一個核心或每一顆CPU的時間加總起來)

sys time:表示程式在kernel mode所佔用的CPU時間總和

(2)如果你的程式可以指定不同的核心數量，請說明在同樣的精準度下，你的程式是否可以得到線性的加速。例如：畫圖，橫軸為core數量，縱軸為所需時間

平行度=user time/real time

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:1
Pi = 3.141593

real    0m4.840s
user    0m4.838s
sys     0m0.001s
shiwulo@vm:~/HW/hw5$
```

4.840/4.838 = 1.0000

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:2
Pi = 3.141593

real    0m3.968s
user    0m7.912s
sys     0m0.004s
shiwulo@vm:~/HW/hw5$
```

7.912/3.968 = 1.9939

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:3
Pi = 3.141593

real    0m2.684s
user    0m7.495s
sys     0m0.004s
shiwulo@vm:~/HW/hw5$
```

7.495/2.684 = 2.7924

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:4
Pi = 3.141593

real    0m2.062s
user    0m7.742s
sys     0m0.000s
shiwulo@vm:~/HW/hw5$
```

7.742/2.062 = 3.7546

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:5
Pi = 3.141593

real    0m1.613s
user    0m7.277s
sys     0m0.021s
shiwulo@vm:~/HW/hw5$
```

7.277/1.613 = 4.5114

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:6
Pi = 3.141593

real    0m1.388s
user    0m7.554s
sys     0m0.024s
shiwulo@vm:~/HW/hw5$
```

7.554/1.388 = 5.4351

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:7
Pi = 3.141593

real    0m1.314s
user    0m7.913s
sys     0m0.024s
shiwulo@vm:~/HW/hw5$
```

7.913/1.314 = 6.0220

```
shiwulo@vm:~/HW/hw5$ time ./pi 6
numCPU:8
Pi = 3.141593

real    0m1.194s
user    0m8.410s
sys     0m0.082s
shiwulo@vm:~/HW/hw5$
```

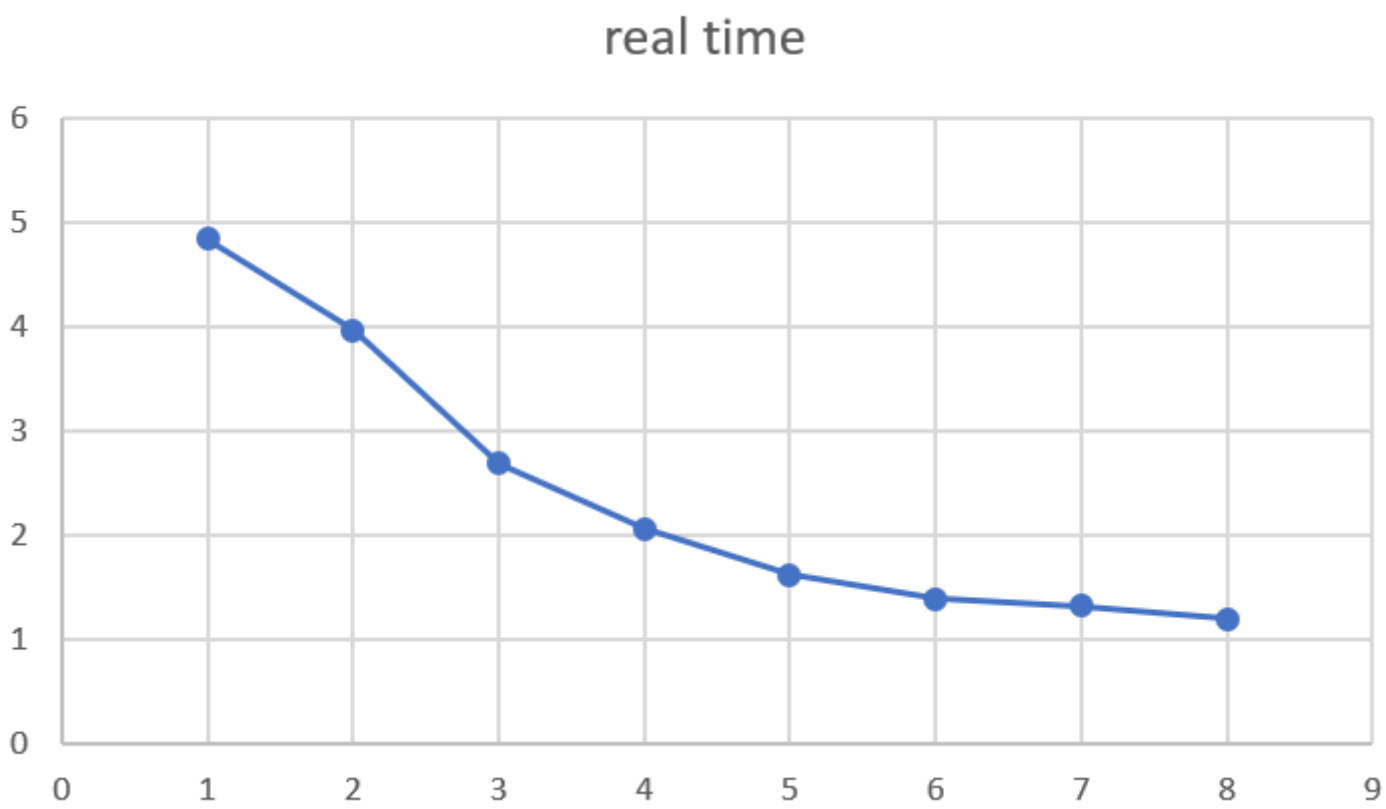
8.410/1.194 = 7.0435

平行度算出來大致上都符合核心個數

Core	平行度
1	1.0000
2	1.9939
3	2.7924
4	3.7546
5	4.5114
6	5.4351
7	6.0220
8	7.0435

時間上大致上核心越多越快，不過到後面核心數較多的時候加速的曲線略為緩和

Core	Real time
1	4.838
2	3.968
3	2.684
4	2.062
5	1.613
6	1.388
7	1.314
8	1.194



(3)請說明你是否使用特別的方法加速你的運算？

因為下界其實就是上界往左shift一個bar，所以我只計算了上界，下界直接拿上界的結果去做
下界=上界-最左邊(最長)的bar+下界最右邊(最短)的bar

```
down=up- 1/loopCount*(1) + 1/loopCount*sqrt(1-pow(1-1/loopCount,2));
```

參考資料

Linux的time指令 | Jason note
https://jasonblog.github.io/note/linux_system/1511.html
(https://jasonblog.github.io/note/linux_system/1511.html)