HW2 Principle of invariance

**张炽 2014010158 建管4**

本实验选用Selection\_Sort验证算法的不变性原理。

* **编程语言**

在Thinkpad X1 Carbon intel®Core™i5-4210U CPU @1.70GHz 2.40GHz, RAM=4G的硬件环境下，在Win10系统下利用C和Python两种编程语言进行算法实现。随着数据量的增长，两语言下的排序时间基本保持着常数项的倍数关系。具体数值见图1。

图1-编程语言的不变性原理

上图中灰色折线代表c语言和python语言运行速度的倍数关系，可以看出，其倍数关系基本保持在较为稳定的范围内，随着数据量的增长c语言的相对运行效率会缓慢提升。由上述实验同样可以看出，C语言的运行速度在同样的算法实现上要远快于Python的实现速度。

* **硬件环境**

在硬件环境测试中采用的两台不同的计算机对同样的代码进行计算。分别为

* intel®Core™i5-4210U CPU @1.70GHz 2.40GHz；RAM=4G
* intel®Core™i5-7500 CPU @3.4GHz 3.4GHz；RAM=8G

操作系统均为Win10，两台计算机对同一代码的执行速度也保持着常数项的倍数关系。具体数值见图2.

图2-硬件环境的不变性原理

上图中灰色折线代表两种硬件环境下c语言执行速度的倍数关系，可以看出，其倍数关系基本保持在较为稳定的范围内，且差异不大。

* **附录-Python代码**

import random

import time

def random\_int\_list(start, stop, length):

start, stop = (int(start), int(stop)) if start <= stop else (int(stop), int(start))

length = int(abs(length)) if length else 0

random\_list = []

for i in range(length):

random\_list.append(random.randint(start, stop))

return random\_list

def selectionsort(A,n):

for i in range(0,n):

minj=i

minx=A[i]

for j in range(i+1,n):

if A[j]<minx:

minx=A[j]

minj=j

A[minj]=A[i]

A[i]=minx

# return A

n=50000

randomList=random\_int\_list(1,100,n)

repeat\_time=100

time\_start=time.time()

for repeat in range(0,repeat\_time):

selectionsort(randomList,n)

repeat=repeat+1

time\_end=time.time()

time\_spent=(time\_end-time\_start)/repeat\_time

print "It took " + str(time\_spent) +" seconds to run selectionsort at n="+ str(n)