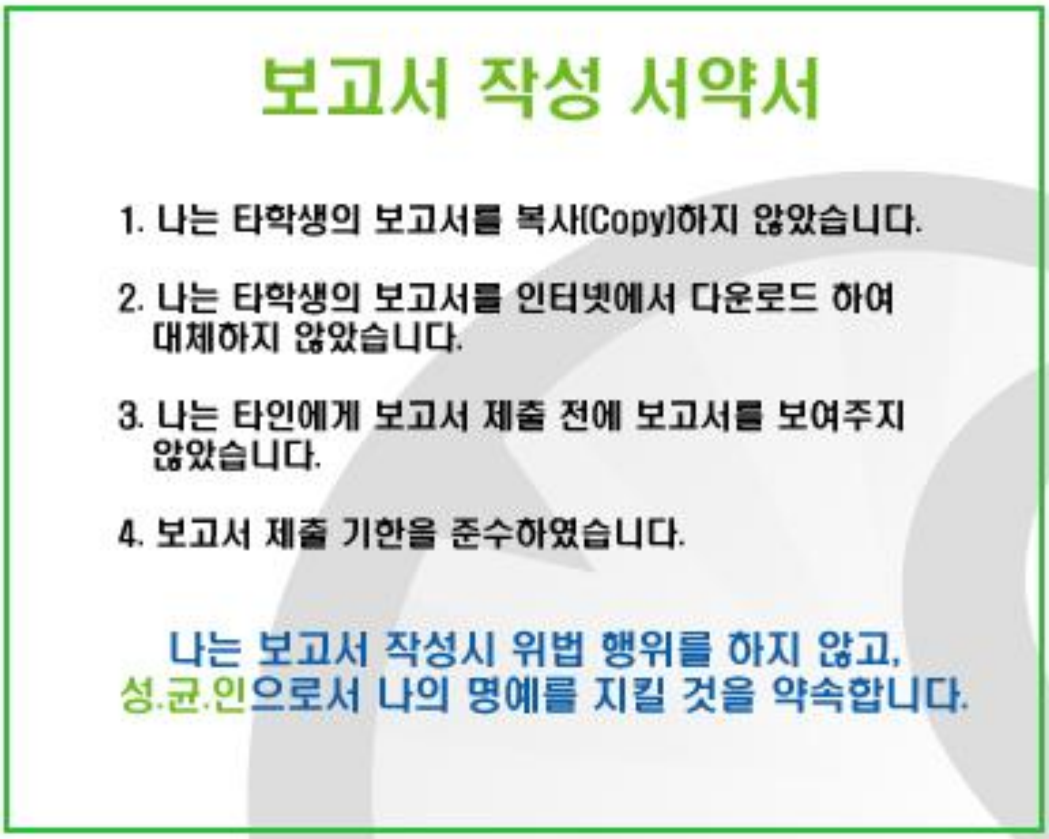
REPORT



보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과 목	:	컴퓨터 구조
과 제 명	:	Project phase
담 당 교 수	:	윤희용 교수님
학 과	:	컴퓨터공학
학 년	:	3학년
학번 / 이름	:	2013311940 이정호
		2013314967 정기빈
제 출 일	:	2015 / 6 / 16 (화)

Index

Phase III: Simulation using SPIM simulator. Report the simulation output. Compare the predicted execution time obtained in Phase II and measured execution time.

- Modification Assembly code

we modify our assembly code. it is because we had found defect in ppq_push.

We'll attach assembly code fixed in Phase 3.   In our code, count instruction is emphasized.

[Assembly code]

```
.data
space: .asciiz "    "
line: .asciiz "\n"
goal: .asciiz "Goal"
info: .asciiz "Travel                                                              dist"
count: .asciiz "Count : "


#next
next      : .space 108

#Travel[7]
Travel   : .word 1,2,3,4,5,6,7,8,9,10,11,12,13,14

#path P_pq[1000];
P_pq      : .space 108000

#path P_pq_index[1000];
P_pq_index : .space 4000

#int current;
current  : .word  1

#int now;
now                  : .word 1

#path Sholtest ( 0 [0] 8 [1] 16 [2] 24 [3] 32 [4] 40 [5] 48 [6] 56 cost 64 distant 72 current 76 size
80... list[7]...108)
```

```
Sholtest : .space 108

         .text                                                    # begin

         .globl main

main:           addi $v1, $zero, 88
                addi    $sp, $sp, -8               # Make room for 2 words on the stack.
                sw      $s0, 0($sp)                                  # Preserve $s0 in
the first slot.
                sw      $ra, 4($sp)                                  # Preserve $ra in
the second slot.

                 li $v0, 4
                 la $a0, info
                 syscall

                 li $v0, 4
                 la $a0, line
                 syscall
                  li $v0, 4
                 la $a0, line
                 syscall

                la $t0, current
                sw $zero, 0($t0)

                la $t0, now
                sw $zero, 0($t0)


                li   $t1, 5
                li   $t2, 5

                la   $t0, Travel

                sw   $t1, 0($t0)
                sw   $t2, 4($t0)

                li   $t1, 2
                li   $t2, 3
                sw   $t1, 8($t0)
                sw   $t2, 12($t0)

                li   $t1, 8
                li   $t2, 4
                sw   $t1, 16($t0)
                sw   $t2, 20($t0)

                li   $t1, 7
                li    $t2, 2
                sw    $t1, 24($t0)
                sw    $t2, 28($t0)

                li    $t1, 1
                li    $t2, 6
                sw    $t1, 32($t0)
                sw    $t2, 36($t0)

                li    $t1, 9
                li    $t2, 6
                sw    $t1, 40($t0)
                sw    $t2, 44($t0)

                li    $t1, 3
                li    $t2, 2
                sw    $t1, 48($t0)
                sw    $t2, 52($t0)

                # Travel initializing



                la          $t0, Sholtest

                li $t1, 0
                mtc1.d $t1, $f0
                cvt.d.w $f0, $f0   # $f0 = 0

                s.d $f0, 72($t0)   # Sholtest.curret = 0
                s.d $f0, 64($t0)   # Sholtest.distant = 0
                s.d $f0, 0($t0)    # Sholtest.city[0] = 0

                li.d $f0, 1.41421 # Sholtest.city[1] = 1.41421
                s.d $f0, 8($t0)

                li.d $f0, 2.23607 # Sholtest.city[2] = 2.23607
                s.d $f0, 16($t0)

                li.d $f0, 2.23607 # Sholtest.city[3] = 2.23607
                s.d $f0, 24($t0)

                li.d $f0, 3.16228 # Sholtest.city[4] = 3.16228
                s.d $f0, 32($t0)

                li.d $f0, 2.23607 # Sholtest.city[5] = 2.23607
                s.d $f0, 40($t0)

                li.d $f0, 1.41421 # Sholtest.city[6] = 1.41421
                s.d $f0, 48($t0)
```

```
            sw      $zero, 80($t0)    # Sholtest.list[0] = 0

            li      $t1, -1
            sw      $t1, 84($t0)      # Sholtest.list[1] = -1
            sw      $t1, 88($t0)      # Sholtest.list[2] = -1
            sw      $t1, 92($t0)      # Sholtest.list[3] = -1
            sw      $t1, 96($t0)      # Sholtest.list[4] = -1
            sw      $t1, 100($t0)     # Sholtest.list[5] = -1
            sw      $t1, 104($t0)     # Sholtest.list[6] = -1

            li      $t1, 1
            sw      $t1, 76($t0)      # Sholtest.size = 1


            # Sholtest initializing

            la $a0, Sholtest

            addi $sp, $sp, -4
            sw $ra, 0($sp)


            jal ppq_push

            lw $ra, 0($sp)
            addi $sp, $sp, 4

            addi $sp, $sp, -4
            sw $ra, 0($sp)

            move $a1, $a0

            jal A_star

            lw $ra, 0($sp)
            addi $sp, $sp, 4


            lw      $s0, 0($sp)                                   # recovery
            lw      $ra, 4($sp)
            addi $sp, $sp, 8

            jr $ra



        .globl ppq_push
# parameter    $a0(adress of path k)
# result is none
```

```
ppq_push:            addi $v1, $v1, 54
                     la $t0, P_pq             # P_pq[0]
                     li $t2,    108                     # index
                     la $t3, current
                     lw $t4, 0($t3)
                     mul $t2, $t2, $t4      # index * current
                     add $t0, $t0, $t2

                     l.d $f0, 0($a0)          # k.city[0]
                     l.d $f2, 8($a0)          # k.city[1]

                     s.d $f0, 0($t0)          # P_pq.city[0] = k.city[0]
                     s.d $f2, 8($t0)          # P_pq.city[1] = k.city[1]

                     l.d $f0, 16($a0)         # k.city[2]
                     l.d $f2, 24($a0)         # k.city[3]

                     s.d $f0, 16($t0)         # P_pq.city[2] = k.city[2]
                     s.d $f2, 24($t0)         # P_pq.city[3] = k.city[3]

                     l.d $f0, 32($a0)         # k.city[4]
                     l.d $f2, 40($a0)         # k.city[5]

                     s.d $f0, 32($t0)         # P_pq.city[4] = k.city[4]
                     s.d $f2, 40($t0)         # P_pq.city[5] = k.city[5]

                     l.d $f0, 48($a0)         # k.city[6]
                     s.d $f0, 48($t0)         # P_pq.city[6] = k.city[6]

                     l.d $f0, 56($a0)         # k.cost
                     s.d $f0, 56($t0)         # P_pq.cost = k.cost

                     l.d $f0, 64($a0)         # k.distant
                     s.d $f0, 64($t0)         # P_pq.distant = k.distant

                     lw $t3, 72($a0)                  # k.current
                     sw $t3, 72($t0)                  # P_pq.current = k.current

                     lw $t3, 76($a0)                  # k.size
                     sw $t3, 76($t0)                  # P_pq.size = k.size

                     lw $t3, 80($a0)                  # k.list
                     sw $t3, 80($t0)                  # P_pq.list = k.list

                     lw $t3, 84($a0)                  # k.list
                     sw $t3, 84($t0)                  # P_pq.list = k.list

                     lw $t3, 88($a0)                  # k.list
                     sw $t3, 88($t0)                  # P_pq.list = k.list
```

```
                lw $t3, 92($a0)                 # k.list
                sw $t3, 92($t0)                 # P_pq.list = k.list

                lw $t3, 96($a0)                 # k.list
                sw $t3, 96($t0)                 # P_pq.list = k.list

                lw $t3, 100($a0)        # k.list
                sw $t3, 100($t0)        # P_pq.list = k.list

                lw $t3, 104($a0)        # k.list
                sw $t3, 104($t0)        # P_pq.list = k.list

                la $t1, P_pq_index      # load P_pq_index
                sll $t3, $t4, 2         # current*4
                add $t1, $t1, $t3  # &P_pq_index[current]
                sw    $t0, 0($t1)       # P_pq_index[current] = &P_pq[current]

                la $t1, current
                lw $t3, 0($t1)
                addi, $t3, $t3, 1
                sw $t3, 0($t1)



                la $t0, current
                lw $t1, 0($t0)          # $t1 = current    .. i=current
                addi $t1, $t1, -1  # i = current - 1

                la $t3, now

                lw $t6, ($t3)

                li $s4, 1

loop_ppq_push:
```
addi $v1, $v1, 1

```
                slt $t3, $t6, $t1
                bne $t3, $s4, exit_loop_ppq_push
```
addi $v1, $v1, 9

```
                la $t0, P_pq_index
                sll $t3, $t1, 2         # i*4

                add $t0, $t0, $t3  # &P_pq_index[i]
                addi $t0, $t0, -4  # &P_pq_index[i-1]

                lw      $t2, 0($t0)             # P_pq_index[i-1]
                lw      $t4, 4($t0)             # P_pq_index[i]
```

```
                        addi $t3, $t2, 56     # P_pq_index[i-1]->cost
                        addi $t5, $t4, 56     # P_pq_index[i]->cost

                        ldc1 $f0, 0($t3)
                        ldc1 $f2, 0($t5)

                        c.lt.d $f2, $f0
                        bc1f exit_if_ppq_push
                        addi $v1, $v1, 2


                        sw $t2, 4($t0)
                        sw $t4, 0($t0)

exit_if_ppq_push:               addi $v1, $v1, 2
                        addi $t1, $t1, -1
                        j loop_ppq_push

exit_loop_ppq_push:     addi $v1, $v1, 2

                        jr $ra


                                .globl sqrt

# parameter ; double a
# result is $f0, input is in $f2

sqrt:                   addi $v1, $v1, 6
                        addi $sp, $sp, -4
                        sw $ra, 0($sp)

                        li $t0, 1
                        mtc1.d $t0, $f0
                        cvt.d.w $f0, $f0

                        move $s0, $zero                         # i = 0


forsqrt:                addi $v1, $v1, 1

                        slti $t0, $s0, 10               # $t0 = 0 if i >= 10
                        beq $t0, $zero, exitsqrt
                        addi $v1, $v1, 9
                        div.d $f6, $f2, $f0         # $f6 = input / x
                        add.d $f6, $f6, $f0             # $f6 = x + (input / x)

                        li $t1, 2
```

```
                    mtc1.d $t1, $f8
                    cvt.d.w $f8, $f8                    # $f8 = 2
                    div.d $f0, $f6, $f8                         # $f0 = x + (input / x) / 2

                    addi $s0, $s0, 1                    # i = i + 1
                    j forsqrt
exitsqrt:
                    addi $v1, $v1, 3
                    lw $ra, 0($sp)
                    addi $sp, $sp, 4
                    jr $ra



        .globl dist
# parameter    $a0(adress of city a), $a2(adress of city b)
# result is $f0

dist :
                addi $v1, $v1, 17
                addi $sp, $sp, -4
                sw $ra, 0($sp)

                lw $t0, 0($a0)          # a.x
                lw $t1, 4($a0)          # a.y
                lw $t2, 0($a2)          # b.x
                lw $t3, 4($a2)          # b.y

                sub $t4, $t0, $t2  # a.x-b.x
                sub $t7, $t1, $t3  # a.y-b.y

                mul $t5, $t4, $t4 # (a.x-b.x)*(a.x-b.x)
                mul $t6, $t7, $t7 # (a.y-b.y)*(a.y-b.y)
                add $t7, $t5, $t6  # (a.y-b.y)*(a.y-b.y) + (a.x-b.x)*(a.x-b.x)

                mtc1 $t7, $f0
                cvt.d.w $f2, $f0

                jal sqrt

                lw $ra, 0($sp)
                addi $sp, $sp, 4

                jr $ra

        .globl A_star
# parameter    $a1(adress of path a)
# result is Printed
```

```
A_star :
                    addi $v1, $v1, 60
                     lw $t0, 80($a1) #         a.list[0]
                     lw $t1, 84($a1) #         a.list[1]
                     lw $t2, 88($a1) #         a.list[2]
                     lw $t3, 92($a1) #         a.list[3]
                     lw $t4, 96($a1) #         a.list[4]
                     lw $t5, 100($a1)          #         a.list[5]
                     lw $t6, 104($a1)          #         a.list[6]

                     addi $t0, $t0, 1
                     addi $t1, $t1, 1
                     addi $t2, $t2, 1
                     addi $t3, $t3, 1
                     addi $t4, $t4, 1
                     addi $t5, $t5, 1
                     addi $t6, $t6, 1



                     li $v0, 1
                     move $a0, $t0
                     syscall

                     li $v0, 4
                     la $a0, space
                     syscall

                     li $v0, 1
                     move $a0, $t1
                     syscall

                     li $v0, 4
                     la $a0, space
                     syscall

                     li $v0, 1
                     move $a0, $t2
                     syscall

                     li $v0, 4
                     la $a0, space
                     syscall

                     li $v0, 1
                     move $a0, $t3
                     syscall

                     li $v0, 4
```

```
la $a0, space
syscall

li $v0, 1
move $a0, $t4
syscall

li $v0, 4
la $a0, space
syscall

li $v0, 1
move $a0, $t5
syscall

li $v0, 4
la $a0, space
syscall

li $v0, 1
move $a0, $t6
syscall

  li $v0, 4
la $a0, space
syscall


l.d $f12, 64($a1)
li $v0, 3
syscall


li $v0, 4
la $a0, line
syscall

# cout << distant

addi $sp, $sp, -4
sw $ra, 0($sp)

jal check_goal

lw $ra, 0($sp)
addi $sp, $sp, 4

bne $v0, $zero, end_A
```
addi $v1, $v1, 5
```
li $v0, 4
```

```
                  la $a0, goal
                  syscall

                la $a0, line
                  syscall

                li $v0, 4
                  la $a0, count
                  syscall


                li $v0, 1                          #   IC print
                move $a0, $v1
                syscall

                  li $v0, 10
                  syscall

end_A :
                addi $v1, $v1, 16

                addi $sp, $sp, -4
                sw $ra, 0($sp)

                  jal Move

                lw $ra, 0($sp)
                addi $sp, $sp, 4

                la $t6, now
                lw $t0, 0($t6)
                sll $t1, $t0, 2 # now*4
                la $t0, P_pq_index
                add $t2, $t1, $t0
                lw $t1, 0($t2)

                move $a1, $t1

                addi $sp, $sp, -4
                sw $ra, 0($sp)

                jal A_star

                lw $ra, 0($sp)
                addi $sp, $sp, 4

                jr $ra

        .globl check_goal
# parameter    $a1(adress of path a)
# result is $v0

check_goal :                    addi $v1, $v1, 2
                                lw $t0, 104($a1) # a.list[6]
                                li $t1, 6
                                bne $t1, $t0,    end_check
                                addi $v1, $v1, 2
                                li $v0, 0
                                jr $ra

end_check :                     addi $v1, $v1, 2
                                li $v0, 1
                                jr $ra



        .globl Move
# parameter    $a1(adress of path a)
# result is None

Move :
                addi $v1, $v1, 4
                la $t0, current
                la $t1, now
                lw $t2, 0($t0)
                lw $t3, 0($t1)

                beq $t2, $t3, move_if_end
                addi $v1, $v1, 2
                addi $t3, $t3, 1
                sw $t3, 0($t1)

move_if_end :   addi $v1, $v1, 1

                move $s7, $zero

for_move :      addi $v1, $v1, 1
                li $t0, 7
                beq    $s7, $t0, end_for_move
                addi $v1, $v1, 5
                addi $sp, $sp, -4
                sw $ra, 0($sp)

                jal is_visited

                lw $ra, 0($sp)
                addi $sp, $sp, 4


                beq $v0, $zero, end_if_move                 # if true -> end_if_move
                addi $v1, $v1, 5
                addi $sp, $sp, -4
                sw $ra, 0($sp)

                jal PUSH

                lw $ra, 0($sp)
                addi $sp, $sp, 4

end_if_move :   addi $v1, $v1, 2
                addi $s7, $s7, 1
                j for_move
end_for_move :  addi $v1, $v1, 1

                jr $ra


        .globl is_visited
# parameter    $a1(adress of path a) $s7(i)
# result is $v0
is_visited:                     addi $v1, $v1, 4
                        move $t0, $zero
                        li $t2, 4
                        li $t1, 7
                        li $v0, 1
for_visited :
                beq $t0, $t1, end_for_visited
                addi $v1, $v1, 4
                mul $t3, $t2, $t0 # 4*i
                add $t3, $t3, $a1
                addi $t3, $t3, 80

                lw $t4, 0($t3)      # P.list[i]

                bne $t4, $s7, end_if_visited
                addi $v1, $v1, 2
                move $v0, $zero
                jr $ra

end_if_visited: addi $v1, $v1, 2
                addi $t0, $t0, 1
                j for_visited

end_for_visited:        addi $v1, $v1, 1
                jr $ra
```

```
            .globl PUSH
# parameter    $a1(adress of path a) $s7(i)
# result is none
PUSH:           addi $v1, $v1, 92
                addi $sp, $sp, -108

                ldc1 $f0, 0($a1)  #city[0]
                sdc1 $f0, 0($sp)

                ldc1 $f0, 8($a1)  #city[1]
                sdc1 $f0, 8($sp)

                ldc1 $f0, 16($a1)#city[2]
                sdc1 $f0, 16($sp)

                ldc1 $f0, 24($a1)#city[3]
                sdc1 $f0, 24($sp)

                ldc1 $f0, 32($a1)#city[4]
                sdc1 $f0, 32($sp)

                ldc1 $f0, 40($a1)#city[5]
                sdc1 $f0, 40($sp)

                ldc1 $f0, 48($a1)#city[6]
                sdc1 $f0, 48($sp)

                ldc1 $f0, 56($a1)# cost
                sdc1 $f0, 56($sp)

                ldc1 $f0, 64($a1)# distant
                sdc1 $f0, 64($sp)

                lw $t1, 72($a1)             # current
                sw $t1, 72($sp)

                lw $t1, 76($a1)             # size
                sw $t1, 76($sp)

                lw $t1, 80($a1)             # list[0]
                sw $t1, 80($sp)

                lw $t1,   84($a1)           # list[1]
                sw $t1, 84($sp)

                lw $t1, 88($a1)             # list[2]
                sw $t1, 88($sp)
```

```
lw $t1, 92($a1)            # list[3]
sw $t1, 92($sp)

lw $t1, 96($a1)            # list[4]
sw $t1, 96($sp)

lw $t1, 100($a1) # list[5]
sw $t1, 100($sp)

lw $t1, 104($a1) # list[6]
sw $t1, 104($sp)


lw $t0, 72($a1)            # now.current
move $t2, $s7              # n_city

sll $t0, $t0, 3                        # now.current * 8
sll $t2, $t2, 3                        # n_city * 8

addi $sp, $sp, -8

sw $ra, 0($sp)
sw $a0, 4($sp)

la $a0, Travel             # addr
la $a2, Travel             # addr

add $a0, $a0, $t0
add $a2, $a2, $t2

jal dist                               # $f0




lw $a0, 4($sp)
lw $ra, 0($sp)



addi $sp, $sp, 8


l.d $f10, 64($sp)
add.d $f10, $f10, $f0
s.d $f10, 64($sp)

sll $t1, $s7, 3
add $t0, $sp, $t1
```

```
s.d $f0, 0($t0)                    # next.city[n_city] = $f0

sw $s7, 72($sp)           # next.current = n_city


mtc1 $zero, $f2
cvt.d.w $f4, $f2

l.d $f0, 0($sp)
add.d $f4, $f4, $f0
l.d $f0, 8($sp)
add.d $f4, $f4, $f0
l.d $f0, 24($sp)
add.d $f4, $f4, $f0
l.d $f0, 16($sp)
add.d $f4, $f4, $f0
l.d $f0, 32($sp)
add.d $f4, $f4, $f0
l.d $f0, 40($sp)
add.d $f4, $f4, $f0
l.d $f0, 48($sp)
add.d $f4, $f4, $f0

s.d $f4, 56($sp)   # cost += all city


lw $t1, 76($a1)
sll $t1, $t1, 2

addi $t1, $t1, 80
add $t1, $t1, $sp

sw $s7, 0($t1)                    # next.list + now.size*4 =    n_city

lw $t1, 76($a1)
addi, $t1, $t1, 1

sw $t1, 76($sp)

move $a0, $sp


addi $sp, $sp, -8
sw $a0, 4($sp)
sw $ra, 0($sp)


jal ppq_push

sw $a0, 4($sp)
```

```
lw $ra, 0($sp)
addi $sp, $sp, 8

addi $sp, $sp, 108

jr $ra
```

- Modification Source code

For optimization of assembly code, we have modified our source code

[Source code]

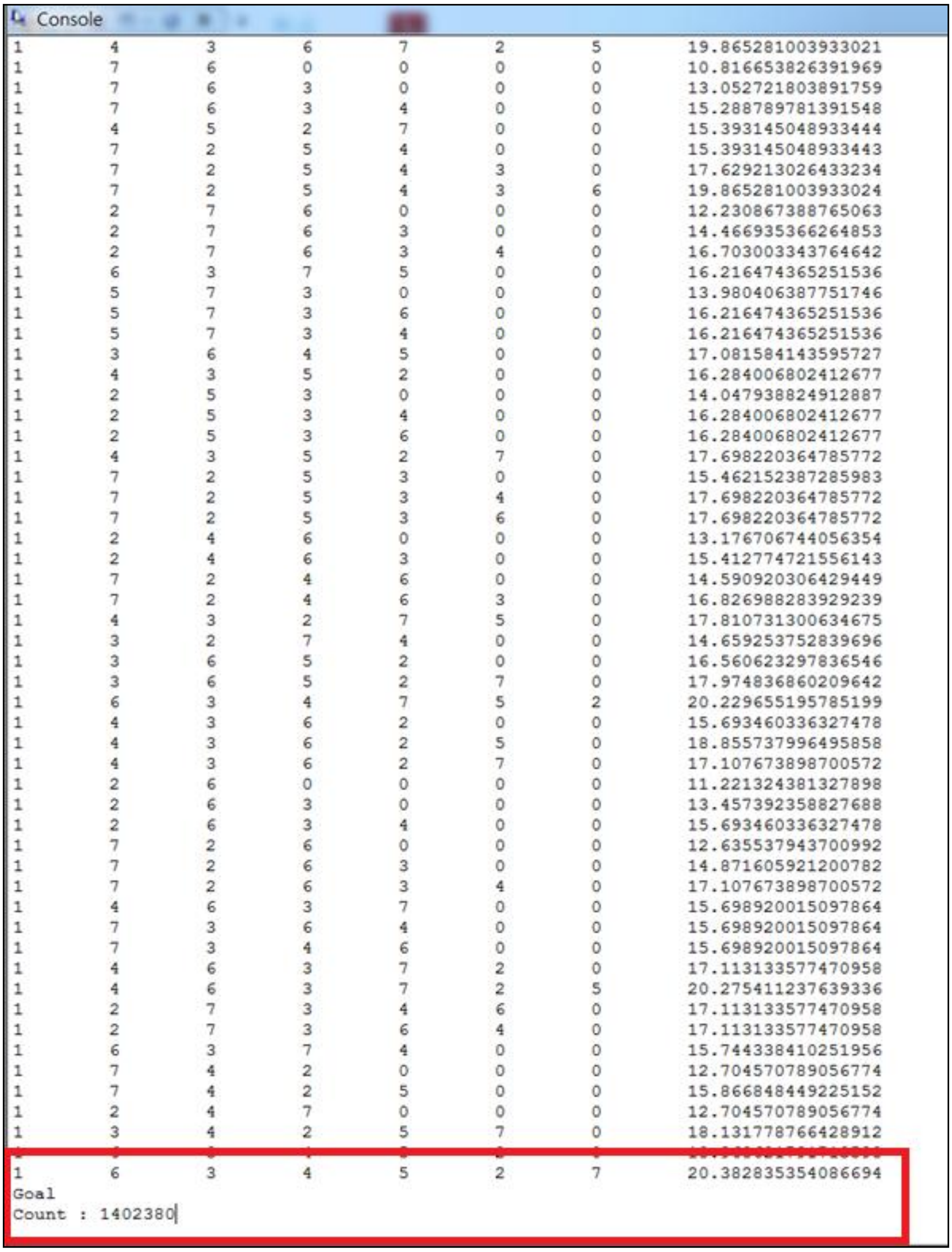
```
#include <iostream>
#include <math.h>
#include <queue>
#include <math.h>
using namespace std;

struct city{
        int x;
        int y;
};

struct path{
        double city[7];
        double cost;
        double distant;
        int current;
        int size;
        int list[7];
};

path P_pq[1000];
path* P_pq_index[1000];
int current;
int now;

void P_pq_push(path k){
        P_pq[current] = k;
        P_pq_index[current] = &P_pq[current];
        current++;

        path* temp = NULL;

        for (int i = current-1; i    > now; i--){
                if (P_pq_index[i]->cost < P_pq_index[i - 1]->cost){
                        temp = P_pq_index[i];
                        P_pq_index[i] = P_pq_index[i - 1];
                        P_pq_index[i - 1] = temp;
                }
        }
}
```

```
city Travel[7];

double dist(city a, city b){
        return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}



bool is_visited(path P, int a){
        for (int i = 0; i < 7; i++){
                if (P.list[i] == a)
                        return true;
        }
        return false;

}

        void PUSH(path now, int n_city){
                path next = now;
                next.city[n_city] = dist(Travel[now.current], Travel[n_city]);
                next.distant += next.city[n_city];
                next.current = n_city;
                next.cost = next.city[0] + next.city[1] + next.city[2] + next.city[3] + next.city[4]
+ next.city[5] + next.city[6];
                next.list[now.size] = n_city;
                next.size = now.size + 1;
                P_pq_push(next);
        }

        void move(path P){
                if (now != current)
                        now++;
                for (int i = 0; i < 7; i++){
                        if (is_visited(P, i) == false){
                                        PUSH(P, i);
                        }
                }

        }


        bool check_goal(path P){
                if (P.list[6] == 6)
                        return true;
                else
                        return false;
        }

        void A_star(path P){
```

```
		int end;
		for (int i = 0; i < 7; i++)
			cout << P.list[i] + 1 << "  ";
		cout << "distant : " << P.distant
;
		cout << endl;
		if (check_goal(P)){
			cout << "Goal!" << endl;
			cin >> end;
		}

		move(P);
		A_star(*P_pq_index[now]);
	}
	path Sholtest;
	int main(void){

		Travel[0].x = 5, Travel[0].y = 5;
		Travel[1].x = 2, Travel[1].y = 3;
		Travel[2].x = 8, Travel[2].y = 4;
		Travel[3].x = 7, Travel[3].y = 2;
		Travel[4].x = 1, Travel[4].y = 6;
		Travel[5].x = 9, Travel[5].y = 6;
		Travel[6].x = 3, Travel[6].y = 2;


		Sholtest.city[0] = 0;
		Sholtest.current = 0;
		Sholtest.list[0] = 0;
		Sholtest.size = 1;
		Sholtest.distant = 0;
		for (int i = 1; i < 7; i++)
			Sholtest.list[i] = -1;

		Sholtest.city[1] = 1.41421;
		Sholtest.city[2] = 2.23607;
		Sholtest.city[3] = 2.23607;
		Sholtest.city[4] = 3.16228;
		Sholtest.city[5] = 2.23607;
		Sholtest.city[6] = 1.41421;
		P_pq_push(Sholtest);
		A_star(Sholtest);
}
```

- The simulation output

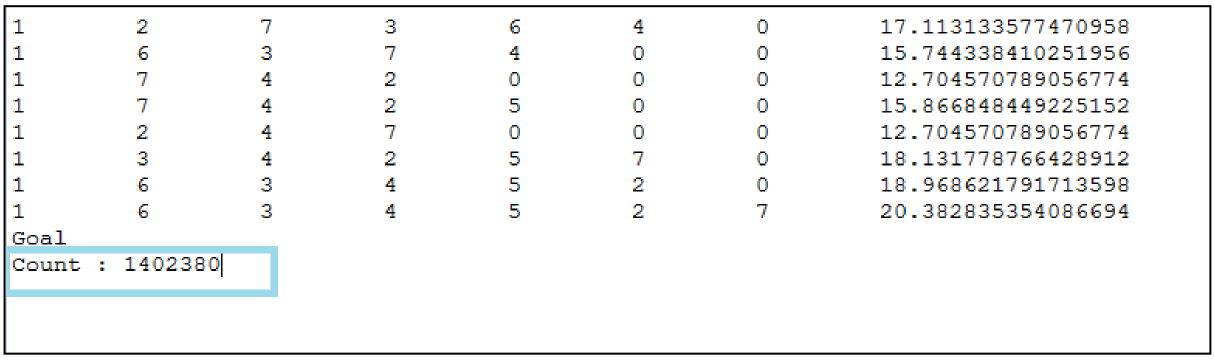
We succed !!
It is our output


Console

```
1         4         3         6         7         2         5          19.865281003933021
1         7         6         0         0         0         0          10.816653826391969
1         7         6         3         0         0         0          13.052721803891759
1         7         6         3         4         0         0          15.288789781391548
1         4         5         2         7         0         0          15.393145048933444
1         7         2         5         4         0         0          15.393145048933443
1         7         2         5         4         3         0          17.629213026433234
1         7         2         5         4         3         6          19.865281003933024
1         2         7         6         0         0         0          12.230867388765063
1         2         7         6         3         0         0          14.466935366264853
1         2         7         6         3         4         0          16.703003343764642
1         6         3         7         5         0         0          16.216474365251536
1         5         7         3         0         0         0          13.980406387751746
1         5         7         3         6         0         0          16.216474365251536
1         5         7         3         4         0         0          16.216474365251536
1         3         6         4         5         0         0          17.081584143595727
1         4         3         5         2         0         0          16.284006802412677
1         2         5         3         0         0         0          14.047938824912887
1         2         5         3         4         0         0          16.284006802412677
1         2         5         3         6         0         0          16.284006802412677
1         4         3         5         2         7         0          17.698220364785772
1         7         2         5         3         0         0          15.462152387285983
1         7         2         5         3         4         0          17.698220364785772
1         7         2         5         3         6         0          17.698220364785772
1         2         4         6         0         0         0          13.176706744056354
1         2         4         6         3         0         0          15.412774721556143
1         7         2         4         6         0         0          14.590920306429449
1         7         2         4         6         3         0          16.826988283929239
1         4         3         2         7         5         0          17.810731300634675
1         3         2         7         4         0         0          14.659253752839696
1         3         6         5         2         0         0          16.560623297836546
1         3         6         5         2         7         0          17.974836860209642
1         6         3         4         7         5         2          20.229655195785199
1         4         3         6         2         0         0          15.693460336327478
1         4         3         6         2         5         0          18.855737996495858
1         4         3         6         2         7         0          17.107673898700572
1         2         6         0         0         0         0          11.221324381327898
1         2         6         3         0         0         0          13.457392358827688
1         2         6         3         4         0         0          15.693460336327478
1         7         2         6         0         0         0          12.635537943700992
1         7         2         6         3         0         0          14.871605921200782
1         7         2         6         3         4         0          17.107673898700572
1         4         6         3         7         0         0          15.698920015097864
1         7         3         6         4         0         0          15.698920015097864
1         7         3         4         6         0         0          15.698920015097864
1         4         6         3         7         2         0          17.113133577470958
1         4         6         3         7         2         5          20.275411237639336
1         2         7         3         4         6         0          17.113133577470958
1         2         7         3         6         4         0          17.113133577470958
1         6         3         7         4         0         0          15.744338410251956
1         7         4         2         0         0         0          12.704570789056774
1         7         4         2         5         0         0          15.866848449225152
1         2         4         7         0         0         0          12.704570789056774
1         3         4         2         5         7         0          18.131778766428912
[illegible]
1         6         3         4         5         2         7          20.382835354086694
Goal
Count : 1402380
```

- Comparison

we compared measured execution time and   predicted execution time in Phase 2



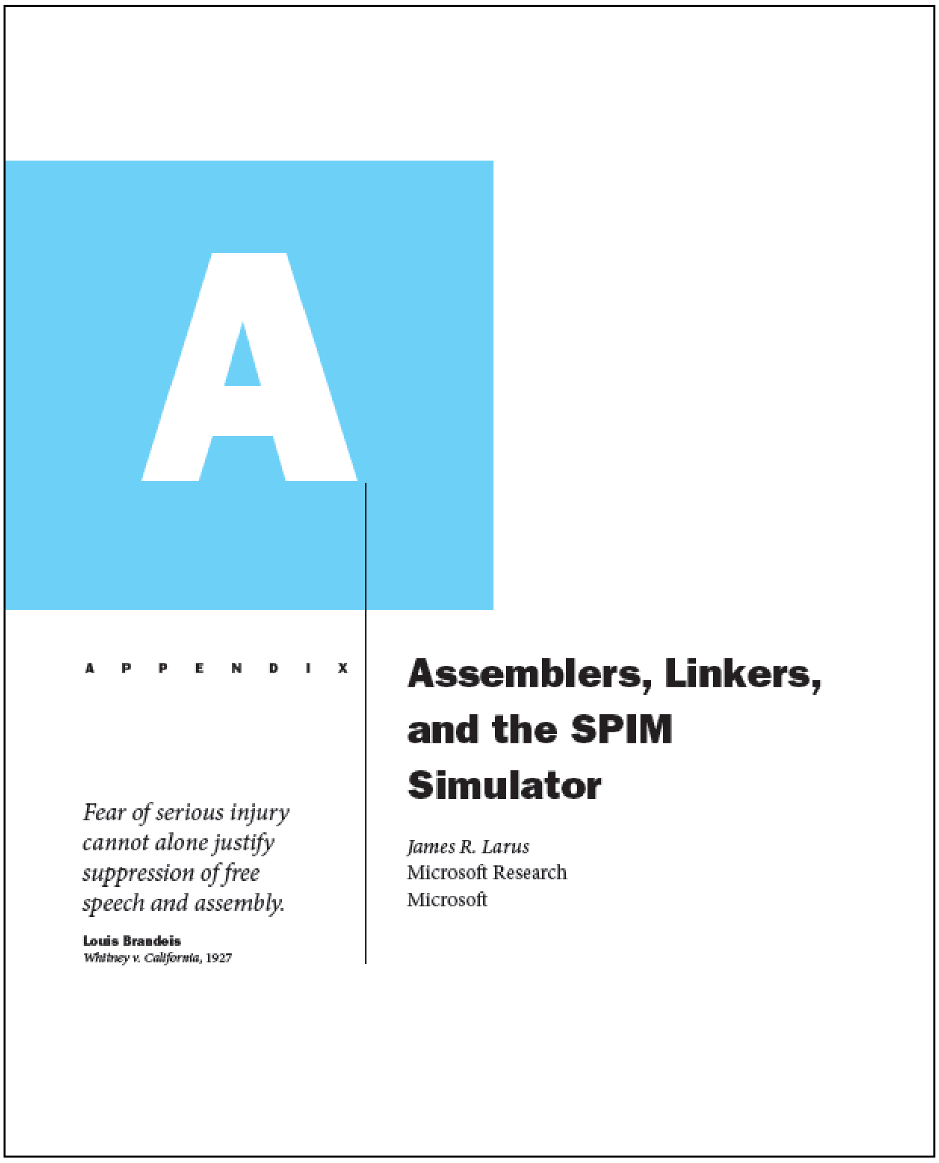
```
1        2        7        3        6        4        0        17.113133577470958
1        6        3        7        4        0        0        15.744338410251956
1        7        4        2        0        0        0        12.704570789056774
1        7        4        2        5        0        0        15.866848449225152
1        2        4        7        0        0        0        12.704570789056774
1        3        4        2        5        7        0        18.131778766428912
1        6        3        4        5        2        0        18.968621791713598
1        6        3        4        5        2        7        20.382835354086694
Goal
Count : 1402380
```

Predicted execution time(IC) : 68000

Meausred execution time(IC) : 1402380

- Reference

We wrote code by referring Appendix A in text book.
It was helpful to us.



A P P E N D I X

A

Assemblers, Linkers, and the SPIM Simulator

James R. Larus
Microsoft Research
Microsoft

Fear of serious injury cannot alone justify suppression of free speech and assembly.

Louis Brandeis
Whitney v. California, 1927