CSE 331/503
Computer Organization
Homework 2 Report

Burak Çiçek
1901042260

Functions: In my code there is one function called finderFunc for finding all increasing subsequences and the longest increasing subsequence of a given array. This function also prints all increasing subsequences and stores the longest subsequence in another array. In function we find sizes of all subsequences and the longest subsequence also. For printing the longest subsequence, I used the main label.

Pseudocode

```
for i=1 to sizeofGivenArray do
        for j = i+1 to sizeofGivenArray do
                sizeofSequence equals 0
                subSequenceArray[sizeOfSequence] equals givenArray[i]
                sizeOfSequence++
                        if2 givenArray[j] greater than or equals
                subSequenceArray[sizeOfSequence-1] than do
                subSequenceArray[sizeOfSequence] equals givenArray[j]
                                for k = j+1 to sizeofGivenArray do
                                if2 givenArray[k] greater than or equals
                subSequenceArray[sizeOfSequence-1] than do
                subSequenceArray[sizeOfSequence] equals givenArray[k];
                sizeOfSequence++
                EndOfLoopWithK
                printArrLoop:
                ...
        if3 sizeOfSequence greater than or equals sizeOfLongestSeq do
                sizeOfLongestSeq equals sizeOfSequence
                for z=0 to sizeOfSequence do
                        longestIncArr[z] equals subSequenceArray[k]
                EndOfLoopWithZ
        EndOfIf3
EndOfIf2
EndOfIf1
EndOfLoopWithJ
EndOfLoopWithI
```

Algorithm Definition: The algorithm includes 3 for loops. i, j, and k. The i loop is holding the initial index for traverse the array and i variable stores in **$t1** register in MIPS code. The j loop is initialized with **i+1** and proceeds from the next index and the k loop inside is initialized with **j+1** and proceeds. j variable stores in **$t4** and k variable stores in **$t7** SizeOfSequence is equal to **zero** every time we return inside the J loop. In assembly code, sizeOfSequence is kept in register **$t6**. It then sets the element in the sizeOfSequence index of the subSequenceArray to the ith index of the givenArray. If the element at index j of the givenArray is greater than or equal to the previous element of the subSequenceArray, I set the subSequenceArray's sizeOfSequence index to the jth index of the givenArray and then increment the sizeOfSequence. Then I switch to the k loop to compare it with other elements. I apply the same logic in this loop. There is an if statement to find the size of the longest Sequence after K loop if sizeOfSequence is greater than or equal to sizeOfLongestSeq then I set sizeOfLongestSeq to sizeOfSequence and enter Z loop to copy the array. Z variable also stores in **$s1** in MIPS code At the end of the function, the sizeOfLongestSeq variable is the size of the longest sequence. The LongestIncArr array also includes the longest sequence.

Time Complexity: O(n^3) , Θ(n^3) - Because of triple nested loops.
Space Complexity: O(1) - Because space has strict upper bounds.

Test Cases

1-

```
givenArray: .word 12, 16, 18, 13, 14, 15,11
```

```
[12,16,18,] Size = 3
[12,18,] Size = 2
[12,13,14,15,] Size = 4
[12,14,15,] Size = 3
[12,15,] Size = 2
[16,18,] Size = 2
[13,14,15,] Size = 3
[13,15,] Size = 2
[14,15,] Size = 2

Longest Increasing Subsequence: [12,13,14,15,] Size = 4
```

2-

```
givenArray: .word 3, 10, 7, 9, 4,11
```

```
[3,10,11,] Size = 3
[3,7,9,11,] Size = 4
[3,9,11,] Size = 3
[3,4,11,] Size = 3
[3,11,] Size = 2
[10,11,] Size = 2
[7,9,11,] Size = 3
[7,11,] Size = 2
[9,11,] Size = 2
[4,11,] Size = 2

Longest Increasing Subsequence: [3,7,9,11,] Size = 4
```

3-

```
givenArray: .word 50, 12, 9, 24, 35,47
```

```
[12,24,35,47,] Size = 4
[12,35,47,] Size = 3
[12,47,] Size = 2
[9,24,35,47,] Size = 4
[9,35,47,] Size = 3
[9,47,] Size = 2
[24,35,47,] Size = 3
[24,47,] Size = 2
[35,47,] Size = 2

Longest Increasing Subsequence: [9,24,35,47,] Size = 4
```

4-

```
givenArray: .word 72, 19, 40, 21
```

```
[19,40,] Size = 2
[19,21,] Size = 2

Longest Increasing Subsequence: [19,21,] Size = 2
```

5-

```
givenArray: .word 3, 10, 40, 21, 87,1,24,99
```

```
[3,10,40,87,99,] Size = 5
[3,40,87,99,] Size = 4
[3,21,87,99,] Size = 4
[3,87,99,] Size = 3
[3,24,99,] Size = 3
[3,99,] Size = 2
[10,40,87,99,] Size = 4
[10,21,87,99,] Size = 4
[10,87,99,] Size = 3
[10,24,99,] Size = 3
[10,99,] Size = 2
[40,87,99,] Size = 3
[40,99,] Size = 2
[21,87,99,] Size = 3
[21,24,99,] Size = 3
[21,99,] Size = 2
[87,99,] Size = 2
[1,24,99,] Size = 3
[1,99,] Size = 2
[24,99,] Size = 2

Longest Increasing Subsequence: [3,10,40,87,99,] Size = 5
```

6-

```
givenArray: .word 8, 5, 11, 6, 13,2
```

```
[8,11,13,] Size = 3
[8,13,] Size = 2
[5,11,13,] Size = 3
[5,6,13,] Size = 3
[5,13,] Size = 2
[11,13,] Size = 2
[6,13,] Size = 2
```

If you want to test the program, here is what you need to do:
1- changing the givenArray as you want
2-Replacing 24 to (size of new givenArray) * 4 in instruction slti $t0, $t7, 24 in Loop3End label (for ex: if new array's size is 9 than new instruction is **slti $t0, $t7, 36**)

```
Loop3End:
addi $t7, $t7, 4 #  k = k + 1, $t7= k
slti $t0, $t7, 24 # $t0 = 1 if $t7 < 24
bne $t0, $zero, Loop3 # go to Loop if $t7 < 24 ///// LOOP 3 END!
```

3-Replacing 24 to (size of givenArray * 4) in instruction slti $t0, $t7, 24 in Loop2End label (for ex: if new array's size is 9 than new instruction is **slti $t0, $t4, 36**)

```
Loop2End:
addi $t4, $t4, 4 #  j = J + 1, $t4 = j
slti $t0, $t4, 24 # $t0 = 1 if $t4 < 24
add $t7, $t4, 4 # k is initalized as k = j+1 end of loop2
bne $t0, $zero, Loop2 # go to Loop if $t4 < 28 ///// LOOP 2 END!
```

4-Replacing 20 to (size of givenArray-1) * 4) in instruction slti $t0, $t7, 20 in Loop1End label (for ex: if new array's size is 9 than new instruction is **slti $t5, $t1, 32**)

```
Loop1End:
addi $t1, $t1, 4 #i'yi arttırır
slti $t5, $t1, 20 # $t5 = 1 if $t1 < 20, i.e. i < 8
add $t4 , $t1 , 4 # j is initalized as j = i+1
add $t7, $t4, 4 # k is initalized as k = j+1
bne $t5, $zero, Loop1 # go to Loop if $t4 < 20 ///// LOOP 1 END!
jr $ra
```

Missing/Bonus/Adding Parts: I'm printing all subsequences with their sizes this is my bonus part. The file reading and writing part is also missing this is my missing part.

However, when my program is run, it gives a single output, but if you want to test the program, I have mention above how it will be.

Thanks for all your efforts
Best Regards,