Scott Ha

CPSC 335

Collaborated with Kenneth Gunderson and Sam Gutierrez

End to Beginning Pseudo code:

Main()

int size\_of\_list, number\_list, parallel\_list 1

input(number of list elements) 1

if number < 1 1+max(1,0)

number too small

number\_list = new\_list 1

parallel\_list = new\_list 1

for i to size of list n+2

input(number\_list [i])

parallel\_list[i] = 0

for i to size of list n+1

output(list element i)

//send list to be processed

end(size\_of\_list, number\_list, parallel\_list) 1

deallocate parallel\_list 1

deallocate number\_list 1

End(size\_of\_list, number\_list, parallel\_list)

parallel[size\_of\_list - 1] = 0 //set last element to 0 1

largest\_possible\_subset = 0 1

for size\_of\_list-2 to 0 1+n-2 + 1

for outer\_loop\_index + 1 to size\_of\_list 1+ n-1 + 1

if number\_list[outer\_loop\_index] < number\_list[inner\_loop\_index] 1 + max(1+max(1,0))

if parallel\_list[outer\_loop\_index] < parallel\_list[inner\_loop\_index]

parallel[outer\_loop\_index] = parallel[inner\_loop\_index] + 1

if parallel[outer\_loop\_index] > largest\_possible\_subset 1+max(1,0)

largest\_possible\_subet = parallel[outer\_loop\_index]

sequence = "" 1

for 0 to size\_of\_list n

if largest\_possible\_subset == parallel\_list[i] 1+max(2,0)

decrement largest\_possible\_subset

sequence.append(number\_list[i]

Time complexity: O(n^2)

Power Set pseudo code:

def longestSubSet(L):

X = subsets(L)

y = 0

R = []

for H in X:

tempR, tempy = checkForLongest(H, length(H))

if (tempR):

R = tempR

y = tempy

return R

def subsets(L):

result = [ [] ] # list containing one empty list

for x in L:

# make a copy of each list in result with x inserted in

with\_x = []

for subset in result:

with\_x.append(subset + [x])

# the new result is the current result plus with\_x

result = result + with\_x

return result

def checkForLongest:(L,n):

y = 0

R = []

for x in range(L):

if(L[x] < L[x+1]):

y++

else:

return

R = L

return (R,y)

Time complexity: O(n^2 + n)