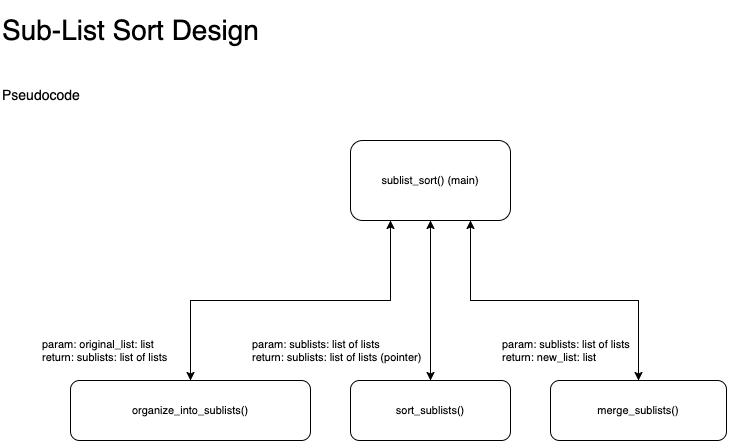
**Sub-List Sort Program Design**

Structure Chart:



Pseudocode:

FUNCTION sublist\_sort(original\_list):

Takes parameter list and returns new list which is

the same as the original list but sorted.

PARAM: original\_list: list

RETURN: new\_list: list

// regular stuff, asserts and whatnot

sublists = organize\_into\_sublists(original\_list)

sort\_sublists(sublists)

new\_list = merge\_sublists(sublists)

return new\_list

FUNCTION organize\_into\_sublists(original\_list):

Takes parameter list and organizes it into sublists,

where each sublist is each already-sorted portion

of the original list, and returns a list containing

these sublists.

PARAM: original\_list: list

RETURN: sublists: list of lists

index = 0

sublists = []

WHILE original\_list.length > 0:

previous = list\_instance[0]

FOR index, current IN original\_list:

IF current < previous:

// slice values from original list up to

// current value in list into new sublist

sublist = original\_list.slice(0, index)

ADD sublist to sublists[]

previous = current // update

// prevent bugs

// we'll get stuck if list is already in order,

// prevent that

IF index IS last index (original\_list.length):

// we've looped through the whole thing and

// nothing was done, this means list is already

// in order

RETURN copy of original\_list // quick exit

// if the rest of list is only 1 value long,

// it will compare against itself and we won't

// be able to exit the loop

IF original\_list.length == 1:

// make a sublist with last value in

// original\_list

sublist = [ original\_list.pop() ]

ADD sublist to sublists

FUNCTION sort\_sublists(sublists):

Takes the sublists passed in parameter and makes sure they are

each already organized.

The return value just returns a reference to the original

sublists variable, we edit that by reference in this function.

PARAM: sublists: list of lists

RETURN: sublists: list of lists

FOR index, sublist in sublists:

// last sublist sublist is possibly in order

IF index == sublist.length and sublist[0] < sublist[-1]:

// if last value is already in order,

// then there is no need to re-sort

CONTINUE // skip this sublist

IF sublist is empty: CONTINUE // skip it

// put in order

// (since last last value is always the min value,

// we'll just take it and insert it at the beginning)

last\_val = sublist.pop()

INSERT last\_val into sublist at index 0

FUNCTION merge\_sublists(sublists):

This function takes sublist and merges them, one sublist at a

time, into a new list called destination[].

The merging process involves removing the first value from

the instance sublist and placing it in destination[] in order.

PARAM: sublists: list of lists

RETURN: sorted\_list: list

destination = []

WHILE sublists.length > 0:

sublist = sublists.pop(0)

// get first sublist and remove to iterate

WHILE sublist.length > 0: // note: sublist not sublists

// get first value and remove to iterate

sublist\_value = sublist.pop(0)

// this will be the index at which we insert

// sublist\_value in destination[]

destination\_index = 0

FOR destination\_value in destination:

IF sublist\_value < destination\_value:

// we have found the position where

// sublist\_value needs to be inserted

// we have the destination index we need

break

// update tracker index, it will only be

// updated until condition from

// if statement above is met

destination\_index += 1

INSERT sublist\_value at destination\_index

RETURN destination