Vincent Siu
10/18/22
Mod: 10/25/22
Asgn 4

Final Design

Assignment 4

#Start code for shell.c

Include header file.
Include stats header file.

declare the function next gap.
This function will find the next gap that the numbers on should be compared with.
If the length of the given array is bigger than 1:
If the length of the given array is bigger than or equal to 2,
Return 1
Else
Return (5*n)//11
Else:
Return 0

Declare the shell sort function of type void with stats instance, array pointer, and the number of elements of the passed in array being the arguments.
#This function will have the main code on sorting an array using the shell sort algorithm.
For each iteration step, starting at the full length of the given array, while the gap is greater than 0, set the gap = to the nextgap() function.
For each iteration i, from gap to full length of array
Set a temp var j = i
Set a temp var temp index= current array index at i
While j (copy of iterator) is bigger than or equal to the gap AND the current array index at j - cap is bigger than the current array index, do the following
Set current array index = index of array_(j-gap)
Subtract gap from j
Set the array at index j equal to array_index i, namely, temp index

Vincent Siu
10/18/22
Mod: 10/25/22
Asgn 4

#Start code for bubble.c

Include header file.
Include stats header file.

#This function will implement the main code for the bubble sort algorithm, where each 2 subsequent elements are being checked for order and being swapped if necessary.

Set a boolean to show that if any swaps have occurred; I will call it if_swapped

Declare the bubble sort function of type void with stats instance, array pointer, and the number of elements of the passed in array being the arguments.
For each iterator i from 0 to the n - 1
    Set if_swapped = false.
    For each iterator j starting from length of array - 1 ending at iterator i, check
        If the current element is smaller than the last element, then
            Swap them
            If_swapped = true to show that a swap has occurred.
    If not swaps have happened:
        Terminate the loops

#Start code for heap.c
#This class will implement the c array sorting algorithm using the heap method
Include standard input and output for printing
Include header file.
Include stats header file.

#get the left child
Declare the left child function
    Return 2 * n + 1
#get the right child
Declare the right child function
    Return 2 * n + 2

Declare the up heap function of type void with stats instance, array pointer, and the number of elements of the passed in array being the arguments. Should be a function pointer.
    While the iterator is bigger than 0 and the current array index of iterator is bigger than the current array index of iterator_parent, do the following,

Vincent Siu
10/18/22
Mod: 10/25/22
Asgn 4

Swap the values of array index of iterator and array iterator_parent
Set the iterator equal to its parent

Declare the down heap function of type int with the arguments being the array pointer and the heap size. Should be a function pointer.

#Start the current node from the root
While the value of the left child of the node is bigger than the heap size, do/ check the following
If the value of the right child is equal to the size of the heap
Set the smaller array value to be equal to the left child
#this occurs if the right child doesn't exist.
Else, check
If the value of the array_left child is smaller than a right child
#what “array” is referring to is the array that IS PASSED in from the heap_sort function
Set the smaller node equal to the left child.
Else, the smaller node will be equal to the right child
If the array_current node is smaller than the value of array_smaller node
Terminate the while loop
Swap the values of array_current node and array_smaller node
Set the current node equal to the smaller node

Declare the build heap function of type int with the argument being the array. Should be a function pointer.

Allocate some memory for the array we will create, using malloc
#This will create a new array for the heap
For each iterator from 0 up until the length of the array, n
Set the memory pointer = to the current value, n, of the passed in array
Call the upheap function with arguments being the heap array (passed as a pointer), and the current iterator.

Declare the heapsort function of type void with stats instance, array pointer, and the number of elements of the passed in array being the arguments.

Get the heap array before the sorting process; this can be done by calling the heap function
Allocate some memory for the new sorted list using malloc
For each iterator from 0 until the length of the passed-in array
Set the current index of the sorted list equal to the first value in the heap array
Then set the first value of the heap array to be equal to the value of the heap array at the index full length of the array - iterator - 1 (we must do -1 so we do not go out of bounds past indices).

Vincent Siu
10/18/22
Mod: 10/25/22
Asgn 4

Call the down heap function with the arguments being heap and length of the array passed subtracted by the iterator.

#Start code for quick.c

#This class will contain the implementation for quicksort array sorting algorithm
Import the shell short header file #that will be used to do the rest of the sorting after the subarrays get small.
Include header file.
Include stats header file.

Declare the quicksort function with stats instance, array pointer, and the number of elements of the passed in array being the arguments.

Declare a variable that will be the minimum number for the length of the array in order to use shell short and then return (base case).

If the length of the array is smaller than the min length
    Call Shell sort with the current array being the argument.
    Return.

Calculate the pivot.

Declare and initialize the variable to track pivot position.

Set a variable swap counter equal to the number of elements -1; I will refer to this variable as j.

For each iterator from i to the last element - 1:
    If the current element is smaller than or equal to the pivot,
        Increment j by 1
        If this is the first iteration, set the swap counter = 0.
        If i does not equal j
            swap the values of i and j.
    If the current element is equal to the pivot,
        Swap the array index of j + 1 with i.
        Set variable pivot position = current index.

If the pivot is not in the array
    Call the function recursively in this way (passing in stats instance, array + j + 1, right bounds subtracting the number of elements to not be included in array, j.)
And (passing in stats instance, array, right bounds being j + 1)
If the pivot is in the array
    Call the function recursively in this way (passing in stats instance, array + pivot pos + 1, right bounds subtracting the number of elements to not be included in array, pivot pos.)
And (passing in stats instance, array, right bounds being j + 1)

Vincent Siu
10/18/22
Mod: 10/25/22
Asgn 4

#Start code for main
Include stdio
Include all header files for each sorting algorithm class.
Include mt header file.
Include inttypes.h to print the PRIu int types.

#declare the main function

#set all the default values first
Size default = 100
Print element default = 100
Seed number default = 13371453
Initialize a set s

Using a while loop parse command arguments with the choices within the options variables as defined above using getopt() and switch cases.

- If the option is a (employing all sorts):
  - add 0 to the set s
- If the option is s (enabling shell sort)
  - Add 1 to the set s
- If the option is b (enabling bubble sort):
  - Add 2 to the set s
- If the option is h (enabling heap sort):
  - Add 3 to the set s
- If the option is q (enabling quick sort):
  - Add 4 to the set s
- If the option includes a size, namely, n
  - If n is smaller than 1 or bigger than 250 million, then return -1
  - Set the size to the size specified
- If the option includes a seed, namely, r
  - Set the current seed to the seed specified
- If the option includes p
  - Enable printing of elements
- If the option includes H
  - Print out program usage
- End loop

Initialize the seed

Allocate memory for the array. Do this 4 times, since there will be 4 different sorts.

Vincent Siu
10/18/22
Mod: 10/25/22
Asgn 4

For each iterator from i until the specified array amount - 1
    Generate a random number for each index of the array by using the mersenne twister Generator

If the printing of elements is enabled:
    For each iterator until the number of elements - 1 specified to print
        While the number of elements is within the bounds of the array
            Print current index of the array.

If 0 is a member of the set s,
    Execute all sorts, with each using the array generated and the array length
    print stats results for each by using for loops to print until the number of elements specified by -p (if included)
Else
    If option 2 is a member of the set s
        Execute the bubble sort
        Print stats results for bubble
    If option 3 is a member of the set s
        Execute the heap sort
        Print stats results for heap
    If option 4 is a member of the set s
        Execute the quick sort.
        Print stats results for quick
    If option 1 is a member of the set s
        Execute the shell short
        Print stats results shell