Assignment 4 Design

Shirin Rokni
shrokni

October 23, 2022

1 Purpose

The purpose of this program is to implement Shell Sort, Bubble Sort, Quick Sort and Heapsort. Shell Sort first sorts pairs of elements which are far apart from each other and increments down the array. Bubble Sort works by examining adjacent pairs of items to sort each element one by one. Quick Sort is a divide-and-conquer algorithm which partitions arrays into sub-arrays and then sorting them. Heapsort uses a binary tree structure which compares parent nodes to the values of its children. This program also has a test harness which accepts various command-line options that can enable all of the sorting algorithms or a selection of them, can set the random seed to an inputted value, can set the number of elements in the array to a specific size, can print out the elements number from the array, and can print the program usage.

2 bubble.c

purpose: implements Bubble Sort

2.1 Include Operations

```
# include <stdio.h>
# include <stdbool.h>
# include "stats.h"
# include "bubble.h"
```

2.2 bubble_sort function

purpose: comparing each adjacent element in the array and swapping them if the left is less than the right. Continuing this cycle until the for-loop (representing each of the elements) has terminated which means that every eleemnt has been sorted.

```
a for loop from 0 to n_elements - 1
    a bool with the variable 'swapped' being set to false
    a for loop from n_elements -1 to the variable of the previous for loop
        a comparison checking if arr[j] is less than arr[j-1]
            swap arr[j] and arr[j-1]
            setting the bool 'swapped' to true
```

if the 'swapped' bool is false, break

3 heap.c

purpose: implements Heap Sort

3.1 Include Operations

```
# include <stdio.h>
# include <stdbool.h>
# include <stdlib.h>
# include "heap.h"
# include "stats.h"
```

3.2 l_child function

*purpose: The left child will become the element present at the input * 2 + 1 index in the array*

return the input * 2 + 1

3.3 r_child function

*purpose: The right child will become the element present at the input * 2 + 2 index in the array*

return the input * 2 + 2

3.4 parent function

purpose: The parent will be element present at (input-1)/2 index in the array

return the input - 1 / 2

3.5 up_heap function

purpose: sorts the path from insertion to root

while the index is positive and arr[n] < arr[parent(n)]
swap arr[n] and arr[parent(n)]
n equals it's parent

3.6 down_heap function

purpose: sorts the from root, checking both children for less than the parent.

set up variables n and smaller
while the l_child is less than the heap size

if there is no right child
    the smaller value is the l_child
else
    if l_child > r_child
        the smaller value is the r_child
    else the smaller value is the l_child
if arr[n] < arr[smaller], break
swap arr[n] and arr[smaller]
the n variable becomes the smaller variable

3.7 build_heap function

purpose: the heap representation of the same array elements.

create a heap array using malloc
a for loop from 0 to n_elements
    swap heap[n] and arr[n]
    call the up_heap function with heap and n as arguments

3.8 heap_sort function

purpose: the driver function that first finds the smallest element and place the smallest element at the beginning. Repeat the same process for the remaining elements under the whole array is sorted.

redefine heap as the build_heap of arr and n_elements as arguments
create a sorted_array using malloc
a for loop from 0 to n_elements
    redefine sorted_list[n] with the first element of the heap
    redefine the first element of the heap with heap[n_elements -n -1]
    call the down_heap with the heap and n_elements -n as arguments
    redefine arr[n] with the sorted_list[n]
free the heap
free the sorted_list

4 shell.c

purpose: implements Shell Sort

4.1 Include Operations

```
# include <stdio.h>
# include "stats.h"
# include "shell.h"
```

4.2 next_gap function

purpose: The "gap sequence" determines the initial size of the gap and how it is made smaller with each iteration. Each iteration of Shell Sort decreases the gap until a gap of 1 is used.

if the gap is less than or equal to 0, return 0
if the gap is less than or equal to 2, return gap - 1
return 5 * gap / 11

4.3 shell_sort function

purpose: the driver function that first sorts elements that are far apart from each other and successively reduces the interval using gaps until the entire array is sorted

define a temp variable
a for loop from next_gap(n_elements) to 0, decrementing from next_gap(gap)
    a for loop from gap to n_elements
        define the variable j with i
        define the variable temp with arr[i]
        while j is greater than or equal to gap or temp > arr[j-gap]
            redefine arr[j] with arr[j-gap]
            decrement j with gap
        redefine arr[j] with temp

5 quick.c

purpose: implements Quick Sort

5.1 Include Operations

```
# include <stdio.h>
# include "stats.h"
# include "quick.h"
# include "shell.h"
```

5.2 quick_sort function

purpose: using pointers to signify the left and right sides of the array. as the pointers get closer to the pivot, they determine whether the values they are pointing to are less than/ greater than the pivot. For the left side, the values must be less than the pivot and vice versa for the right. If both points find a value that doesn't follow that rule, they swap places and then recursively call quick sort on that smaller list of arrays. This keeps on happening until the entire array sequence is sorted.

define a SMALL variable as 8
if n_elements is less than SMALL, call shell_sort with the arguments arr and n_elements
define the left variable as 0 and right variable as n_elements - 1

define pivot as the summation of arr[left], arr[n_elements/2] and arr[right] then divide by 3
while left is less than right
    while arr[left] is less than or equal to pivot
        increment left
    if the left pointer is at the end of the array, return
    while arr[right] is greater than pivot and right is greater than left
        decrement right
    if left is less than right still
        swap arr[left] with arr[right] call quick_sort again with the arguments arr and left
call quick_sort again with arguments arr + left and n_elements - left

6 sorting.c

purpose: running the test harness and allows command line options to display the various sorting functions

6.1 Include Operations and Definitions

```
# include <unistd.h>
# include <stdio.h>
# include <inttypes.h>
# include <stdint.h>
# include <stdlib.h>
# include "bubble.h"
# include "heap.h"
# include "mtrand.h"
# include "quick.h"
# include "set.h"
# include "shell.h"
# include "stats.h"
# define OPTIONS with asbqhr:n:p:H
```

6.2 prints function

purpose: printing the arrays below the printed statements if the user so desires.

a for loop from 0 to the printing variable
    print space[i] within 13 characters
    print an extra line if not at the end of the code and five columns have been created
if there aren't no values to print, print a new line character

6.3 seeding function

purpose: seeding the mersenne twister random seed generator and bit masking it.

initiate the seeding using mtrand_seed

a for loop from 0 to the size value
    bit-masking by making space[i] be defined with the random value and bit-masking it to fit in 30 bits

6.4 main function

purpose: using a getopt switch case and utilizing the set functions to call certain blocks of code. Also, allowing the user to input their seed, size and element value as well as printing the usage statement if an incorrect flag is used. Then, below the getopt loop, it uses the set functions to activate certain chunks of code which will use certain sorting methods.

define an array with the Stats function and using calloc
define a char pointer ptr
define an integer opt
define uint32_t variables seed, size, elements space and printing
set an empty variable x with the Set function
a while loop using getopt
a switch case with opt as the expression
    case 'a'
        define x as the set_insert of 0
        define x as the set_insert of 1
        define x as the set_insert of 2
        define x as the set_insert of 3 and break
    case 'b'
        define x as the set_insert of 0 and break
    case 'h'
        define x as the set_insert of 1 and break
    case 'q'
        define x as the set_insert of 2 and break
    case 's'
        define x as the set_insert of 3 and break
    case 'r'
        set the seed variable to the user input using strtoul and optarg and break
    case 'n'
        set the size variable to the user input using strtoul and optarg
        if the user input is less than 1 or greater than 250,000,000, return 1
        break
    case 'p'
        set the element variable to the user input using strtoul and optarg and break
    case 'H'
        define x as the set_insert of 4 and break
    default case
        define x as the set_insert of 4 and break
        return a nonzero value

    if the size variable is less than the elements variable, define printing with size otherwise define it with elements

define the space variable as a new array with malloc
if the space variable is null, return 1
if set_member is 0
    call seeding function with seed, space and size arguments
    reset the arr array
    call bubble sort with arguments arr, space, and size
    print the size, moves and compares
    calling the prints function with arguments printing and space
if set_member is 1
    call seeding function with seed, space and size arguments
    reset the arr array
    call heap sort with arguments arr, space, and size
    print the size, moves and compares
    calling the prints function with arguments printing and space
if set_member is 2
    call seeding function with seed, space and size arguments
    reset the arr array
    call quick sort with arguments arr, space, and size
    print the size, moves and compares
    calling the prints function with arguments printing and space
if set_member is 3
    call seeding function with seed, space and size arguments
    reset the arr array
    call shell sort with arguments arr, space, and size
    print the size, moves and compares
    calling the prints function with arguments printing and space
if set_member is 4
    print the synopsis line and description
    print the usage line and description
    print all of the command line option possibilities with descriptions
    if there aren't no values to print, print a new line character
free the space array
return 0