Viveka Agrawal
Professor Veenstra
CSE13S, Winter 2023

CSE13S ASGN3 DESIGN DOC

Description of the program:

In this assignment, we are writing 4 different sorting algorithms:
Shell sort, Batcher sort, Heap sort, and Quick sort.

In this assignment, 14 files will be created and submitted on git. They include:

batcher.c: implements Batcher Sort

batcher.h: specifies the interface to batcher.c

heap.c: implements Heap Sort

heap.h: specifies the interface to heap.c

quick.c: implements recursive Quicksort

quick.h: specifies the interface to quick.c

set.c: implements bit-wise Set operations

set.h: specifies the interface to set.c

stats.c: implements the statistics module

stats.h: specifies the interface to the statistics module

shell.c: implements Shell Sort

shell.h: specifies the interface to shell.c

gaps.h: provides a gap sequence to be used by Shell sort

sorting.c: contains main() and may contain any other functions necessary to complete the

assignment

A makefile, readme document, and writeup must also be completed for this assignment.

Shell Sort
-Include shell.h, gaps.h, stats.h, inttypes.h, stdint.h, stdio.h, and unistd.h

-Create a void function named shell_sort that has parameters of pointer to stats variable stats, pointer to a 32 bit integer array A, and the length of array A called n which is a 32 bit integer variable

-for a 32 bit integer variable k initialized to 0, k less than GAPS, increment k by 1
-set the 32 bit integer variable gap equal to index k of gaps
-for 32 bit integer variable i equals gap, i is less than n, increment i by 1
-set a 32 bit integer variable j equal to i
-create another 32 bit integer variable named temp and set it equal to the move function with stats and the array A of index i
-while j is greater than or equal to gap and the comparison between stats, temp, and the array A of index j minus gap is less than 0
-set the index j of the array A equal to the move function with stats and the array A of index j minus gap
-set j equal to the value of j minus g
-set index j of the array A equal to move function with stats and temp

Heap Maintenance
-Include heap.h, stats.h, inttypes.h, stdint.h, stdio.h, unistd.h, and stdbool.h

-create an integer function called max_child which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and 2 a 32 bit integer variables (named first and last) as the parameters

-set a 32 bit integer variable named left equal to 2 times first
-set a 32 bit integer variable named right equal to left plus 1
-if right is less than or equal to last and index right minus 1 of the array is greater than index left minus 1 of the array
-return right
-otherwise return left

-create a void function called fix_heap which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and 2 a 32 bit integer variables (named first and last) as the parameters
-create a a 32 bit integer variable named found and set it equal to false

-create a a 32 bit integer variable named mother and set it equal to first
-create a 32 bit integer variable named great and set it equal to the function of max_child with parameters of A, mother, and last
-while mother is less than or equal to last divided by 2 (integer division) and found is not true
-compare if index mother minus 1 of the array is less than index great minus 1 of the array
-swap the address of index mother minus 1 of the array and index great minus 1 of the array
-set mother equal to great
-set great equal to the function max_child with the parameters of stats, the array, mother, and last
-else, set found equal to true

Heapsort

-create a void function called build_heap which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and 2 a 32 bit integer variables (named first and last) as the parameters
-in a for loop, set a 32 bit integer variable named father and set it equal to last divided by 2 (integer division)
-in the same for loop, the range of father is from last divided by 2 until father is greater than first minus 1 and decrement father by 1
-call the function fix_heap with the parameters of stats, the array, father, and last

-create a void function named heap_sort which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and a 32 bit integer variable named n (represents the length of array A) as the parameters
-set a 32 bit integer variable named first equal to 1
-set a 32 bit integer variable named last equal to the length of the array (n)
-call build_heap with stats, the array, first, and last as the parameters
-for a 32 bit integer variable named leaf in the range last to first, decrementing by 1
-swap the address of index first minus 1 of the array and index leaf minus 1 of the array
-call the function fix_heap with stats, the array, first, and leaf minus 1 as the parameters

Quicksort - Partition

-Include quick.h, stats.h, inttypes.h, stdint.h, stdio.h, and unistd.h

-create an integer function named partition which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and 2 a 32 bit integer variables (named lo and high) as the parameters

-create a 32 bit integer variable named i and set it equal to lo minus 1
-for a 32 bit integer variable j in the range of lo to hi and incrementing j by 1
-compare if index j minus 1 of the array is less than index hi minus 1 of the array
-increment i by 1
-swap the address of index i minus 1 of the array and index j minus 1 of the array
-swap the address of index i of the array and index hi minus 1 of the array
-return i plus 1

Quicksort - Recursive

-create a void function called quick_sorter which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and 2 32 bit integer variables (named lo and high) as the parameters
-if lo is less than hi
-set a 32 bit integer variable named p equal to the function partition with the stats, array, lo, and hi as the parameters
-call the function quick_sorter with the stats, array, lo, and p minus 1 as the parameters
-call the function quick_sorter with the stats, array, p plus 1, and hi as the parameters

-create a void function called quick_sort which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and a 32 bit integer variable named n which is the length of array A as the parameters
-call the function quick_sorter with the stats, array, 1, and the length of the array (n) as the parameters

Batcher Sort

-Include the math.h library since log will be used, batcher.h, stats.h, inttypes.h, stdint.h, stdio.h, unistd.h, and stdbool.h

-create a void function called comparator which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and 2 32 bit integer variables named x and y as the parameters
-compare if index x of the array is greater than index y of the array
-swap the address of index y of the array and index x of the array

-create a void function called batcher_sort which takes a pointer to stats variable stats, pointer to a 32 bit integer array A, and a 32 bit integer variable named n which is the length of array A as the parameters
-if n equal 0
-return

-set a 32 bit integer variable named t equal to log base 2 of n plus 1
-left-shift 1 by (t minus 1) and store that value in a 32 bit integer named p
-while p is greater than 0
-left-shift 1 by (t minus 1) and store that value in a 32 bit integer named q
-set a 32 bit integer variable named r equal to 0
-set a 32 bit integer variable named d equal to p
-while d is greater than 0
-for a 32 bit integer variable named i in range of 0 to (n minus d) and incrementing i by 1
-if the bit-wise and of i and p equals r
-call the function comparator with the stats, array, i, and i plus d as the parameters
-set d equal to q minus p
-right-shift q by 1
-set r equal to p
-right-shift p by 1

Sorting.c

This is the function where main() is.
-Include shell.h, heap.h, batcher.h, quick.h, stats.h, set.h, inttypes.h, stdint.h, stdio.h, unistd.h, stdlib.h, and stdbool.h
Define OPTIONS with all the command line options used for this assignment and BIT_MASK 0x3FFFFFFF
typedef enum with all the names of the sorting methods {shell, batcher, heap, quick}
Create array of names to name for the typedef enum: {“shell sort”, “batcher sort”, “heap sort”, “quick sort”}
In the main function with parameters integer argc and character argv deference twice,
Have integer variable opt initialized to 0
Create an empty set and set it to Set variable s
Stats stats
Reset the address of stats
Initialize a 32 bit integer variable named seed to 13371453
Initialize 2 32 bit integer variables named size and element to 100
Set the integer variable no_user_input to true
Set the integer variable test_H to false
Create a while loop which will check the command line options
Set no_user_input to false
Create a switch statement and a case statement for each command line option
For case a set s to set_universal() and break
For case ‘h’ set HEAP into set s and break

For case ‘b’ set BATCHER into set s and break
For case ‘s’ set SHELL into set s and break
For case ‘q’ set QUICK into set s and break
For case ‘r’ set seed equal to strtoul with parameters optarg, null, and 10 and then break
For case ‘n’ if strtoul with parameters optarg, null, and 10 is greater than 100, set size equal to 100 else set size equal to strtoul with parameters optarg, null, and 10 and then break
For case ‘p’ if strtoul with parameters optarg, null, and 10 is greater than 100, set element equal to 100 else set element equal to strtoul with parameters optarg, null, and 10 and then break
For case ‘H’ set test_H equal to true and then break
Otherwise, default break
Have srandom(seed)
Dynamically allocate memory using calloc
Use a for loop to get a random number and bitmask
If the size is less than element, set element equal to size
If there is no user input or if -H is entered, print out the necessary message
at the beginning of each if statement, make sure to reset the number of moves and compares, have srandom(seed), and use a for loop to get a random number and bitmask
If -a is chosen, then print out all the sorts with an array element width of 13 and for printing the table in 5 columns
If -s is chosen, print results for shell sort with an array element width of 13 and for printing the table in 5 columns
If -b is chosen, print results for batcher sort with an array element width of 13 and for printing the table in 5 columns
If -h is chosen, print results for heap sort with an array element width of 13 and for printing the table in 5 columns
If -q is chosen, print results for quick sort with an array element width of 13 and for printing the table in 5 columns
Free the dynamically allocated memory for array A in order to pass valgrind
Return 0