Kat Pe Benito

Professor Veenstra

CSE13S

31 January 2023

Assignment 3 Design Doc.

Description of Program:

C programs that contain implementations of different sorts such as Shell, Batcher, Heap, and recursive Quicksort, as well as a test harness containing an array of ‘random’ elements to test differences between each sorts.

Files to be included in directory “asgn1”:

- batcher.c: implements the Batcher Sort
- batcher.h: specifies the interface to batcher.c
- shell.c: implements shell short
- shell.h: specifies the interface to shell.c
- gaps.h: provides a gap sequence to be used by shell sort
- heap.c: implements Heap sort
- heap.h: specifies the interface to quick.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: stats.c implements the statistics module

- stats.h: specifies the interface to the statistics module
- sorting.c: contains main() and other functions

- Makefile: Makes and cleans all c files, compiles and formats them as well.
- README.md: Describes how to use script and Makefile in Markdown syntax, explains command-line options that program accepts. Lists and bugs or errors if any
- DESIGN.pdf: Describes design for program thoroughly with pseudocode and descriptions.
- WRITEUP.pdf: What I learned from different sorting algorithms, what conditions do sorts perform well or poorly, and conclusions from my findings. Graphs explaining performance of the sorts of variety of inputs, and analysis of such.

Pseudocode:

batcher.c

- def comparator(A< x, y
- if A[x] > A[y]
    - interchange A[x], A[y]
- def batcher sort(A, n)
- if n = 0
    - return
- t = leading zeros of n - leading zeros 0
- p = 2^(t-1)
- r = 0

- d = p
- while p > 0
    - q = 2^(t-1)
    - r = 0
    - d = p
    - while d > 0
        - for i = 0, n-d
            - if i and p = r
            - run comparator (A,i, i+d)
        - d = q - p
        - q = q shift right 1
        - r = p
    - p = p shift right 1

shell.c

- def shell_sort(stsats , A , n
- for i = log (3+2 * n) / log(3); i >0; i–
- for i = gapl; i < n; i++
- j = i, temp = move(stats, A[i]
- while j >= gap and cmp stats,temp, A[j - gap] == -1
    - A[j] = move (stats, A[j - gap]
    - j -= gap;
    - A[j] = move (stats, temp);

heap.c

- headers: stdio, stlib, stdbool
- def build heap(A, first, last)
    - for father = last divided by 2; father >= first; father–
    - run fix heap (A, father, last)
- def heap sort(A, n)
- first = 1
- last = n
- run build heap(A, first, last)
- for leaf = last; leaf>= first; leaf–
    - temp = A first - 1
    - set Afirst to aleaf-1
    - a leaf -1 = temp
    - run fix heap (A, first, leaf -1)
- def max child (A, first, last)
- left = 2 * first
- right = left +1
- if right <= last and A right-1 > A left -1
    - return right
- return left
- def fix heap(int a[], int first, int last)
    - set bool found to false
    - mother = first

- great = max_child (A, mother, last);
- while mother <= last / 2 and not found
    - if A[mother - 1] < A[great-1]
        - interchange A[mom - 1], A[great -1]
        - mom = great
        - great = max child (A, mom, last)
    - else
    - found is true

quick.c

- partition(A, lo, hi);
- def quick sorter(stats, a, lo, hi)
- def quick sort(stats, A, n)
    - run quick_sorter(stats,a, 1, n)
    - return
- partition (stats, a, lo, hi
- i = lo - 1
- for (j = lo; j < hi; j++
    - if cmp(stats, A[j - 1], A[hi - 1];
- swap(stats, &[A[i]], A[hi - 1]
- return i + 1;

sorting.c

- headers: stdio, stdlib, time
- define default vars for seed and size

- define options for switch case
- create void function to make random array of elements
- int i
- for i = 0 int < size; i++
- arr[i] = rand();

- define main function with arc char **argv
    - run function for generating random array with desired or default seed and size
    - set all options to 0,
    - while opt = get opt (argc, argv, options are not -1)
    - switch cases for each command option and break
    - if *insert command option* run sort with previously generated array

Credit:

- https://www.geeksforgeeks.org/generating-random-number-range-c/
- geeksforgeeks youtube video series on different types of sorts
- Ran into issues with undefined reference to main, got help from Ben, realized my Makefile was compiling everything in my directory, which included other irrelevant files
- asked a friend about an error who suggested I might be declaring the wrong pointer for some variables.
- got a lot of segmentation errors when running command line options, Ben's Piazza post helped debug.