Mylo Lynch

Darrel Long

Cse 13s

2/5/23

Sorting: Putting your affairs in order

Deliverables:

Batcher.c implements Batcher Sort.

batcher.h interface to batcher.c.

shell.c implements ShellSort.

shell.h interface to shell.c.

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

heap.c implementsHeapSort.

heap.h specifies the interface to heap.c.

quick.c implement recursive Quicksort.

quick.h interface to quick.c.

set.c implements bitwise Set operations.

set.h interface to set.c.

stats.c implements the statistics module.

stats.h specifies the interface to the statistics module.

sorting.c contains main() and *may* contain any other functions necessary to complete the assignment. IS TEST HARNESS

Sorting.c

`./sorting -H` prints

“SYNOPSIS
A collection of comparison-based sorting algorithms.
USAGE
./sorting [-Hasbhqn:p:r:] [-n length] [-p elements] [-r seed]

OPTIONS
-H Display program help and usage.
-a Enable all sorts.
-s Enable Shell Sort.
-b Enable Batcher Sort.
-h Enable Heap Sort.
-q Enable Quick Sort.
-n length Specify number of array elements.
-p elements Specify number of elements to print.
-r seed Specify random seed (default: 13371453).”

These are the command line arguments that need to work. Use the set operations in set.c in the case statements for these arguments in order to change the command being called, then use a loop with if statements to check the command. Create a random array and then use calloc and memcpy to copy the data of that array into the one with memory allocated to it to be passed as a parameter in the sorting algorithm test.

Insertion sort

For each i from $0 \leq i \leq (n-1)$, compare i with each previous i until sorted in increasing order

Shell sort

Variation of insertion sort, but instead of doing each elements jump by different sized gaps. The gaps get smaller with each iteration of the loop comparing elements. The gaps can be reduced using bitshift or variable assignment. Pratt sequence in the given header file gaps.h has has the fastest gap sizes.

Heapsort

Heap can be implemented as an array with a heap-like structure, in which for any index i, the index of its left child is $2i$ and the index of its right child is $2i + 1$.

Parent is i/2 and this heap is a max heap

Create max heap and then fix it by putting the largest elements that are at the beginning of the array into the end of the array

Quicksort

Select a pivot value

Less than pivot go left

Bigger than pivot go right’

Recursively divide and conquer

Batcher

K-sort then k/2 then k/4 until 1-sort

Uses comparing logic along wires to perform sort

One wire for each input to sort and are connected by comparators that compare and swap the values when necessary

Since there are a fixed number of comparators, a sorting network must sort any input using a fixed number of comparisons.

General tips:

- Heapsort will need binary tree
- Can’t bitsize array have to dynamically allocate
- Look into calloc instead of malloc
- Look into valgrind may not be installable by arm—-- was installable
- Bitshifts can be used to simulate multiplication and division on unsigned integers
- To get bit length, use a loop to shift the value of size ‘n’ until it = 0 and count the bitshifts

Makefile

```
CC      = clang
CFLAGS  = -Wall -Wextra -Wpedantic -Werror -g
TARGET  = sorting

all: $(TARGET)

sorting: sorting.c heap.c batcher.c quick.c shell.c set.c stats.c
	$(CC)   -o sorting heap.c batcher.c quick.c shell.c set.c stats.c sorting.c -lm

sorting.o: *.c
	$(CC)   $(CFLAGS) -c *.c

format:
	clang-format -i -style=file *.[ch]

clean:
	rm -f sorting *.o
```