Assignment 3 Sorting: Putting your affairs in order

Design Document

Derrick Ko - Winter 2023

1 Overview

In this assignment we will be using different sorting algorithms to sort numbers while calculating the stats of amount of elements sorted, moves it took to get sorted and compares between numbers.

2 batcher.c

Merge Exchange Sort (Batcher's Method) in Python

```
def comparator (A: list , x: int , y: int):
    if A[x] > A[y]:
        A[x], A[y] = A[y], A[x]

def batcher_sort (A: list ):
    if len(A) == 0:
        return

    n = len(A)
    t = n. bit_length ()
    p = 1 << (t - 1)

    while p > 0:
    q = 1 << (t - 1)
    r = 0
    d = p

        while d > 0:
            for i in range (0, n - d):
                if (i & p) == r:
                    comparator (A, i, i + d)
                    d = q - p
                    q >>= 1
                    r = p

        p >>= 1
```

Compares will be tracked in the many of the if statements and while statements. Comparator

- Tracked in the if condition
- Swaps within the if scope

n.length

- bitwise right shift operation on the variable n
- add count for each loop

3 shell.c

Shell Sort in Python

```
def shell_sort(arr):
   for gap in gaps:
        for i in range (gap, len(arr)):
           j = i
           temp = arr[i]
            while j >= gap and temp < arr[j - gap]:
                arr[j] = arr[j - gap]
                j -= gap
            arr[j] = temp

```

- Compare in the while loop
- Move i
- move arr[j - gap]
- move temp

4 heap.c

Heap maintenance in Python

```
def max_child (A: list , first : int , last : int ):
    left = 2 * first
    right = left + 1
    if right <= last and A[ right - 1] > A[ left - 1]:
        return right
    return left

    def fix_heap (A: list , first : int , last : int ):
        found = False
        mother = first
        great = max_child (A, mother , last )

    while mother <= last // 2 and not found :
        if A[ mother - 1] < A[ great - 1]:
            A[ mother - 1], A[ great - 1] = A[ great - 1], A[
    mother - 1]
            mother = great
            great = max_child (A, mother , last)
        else :
            found = True
```

Fix Heap

- compare in if condition
- swap in if scope

Heapsort in Python

```
def build_heap (A: list , first : int , last : int):
    for father in range ( last // 2, first - 1, -1):
        fix_heap (A, father , last )

def heap_sort (A: list ):
    first = 1
    last = len(A)
    build_heap (A, first , last )
    for leaf in range (last , first , -1):
        A[ first - 1], A[ leaf - 1] = A[ leaf - 1], A[ first - 1]
        fix_heap (A, first , leaf - 1)
```

- Swap in for loop

5 quick.c

Partition in Python

```
def partition (A: list , lo: int , hi: int):
    i = lo - 1
    for j in range (lo , hi):
        if A[j - 1] < A[hi - 1]:
            i += 1
            A[i - 1], A[j - 1] = A[j - 1], A[i - 1]
        A[i], A[hi - 1] = A[hi - 1], A[i]
        return i + 1
```

- compare in if condition
- swap in if scope
- swap above return i + 1

Recursive Quicksort in Python

```
# A recursive helper function for Quicksort .
def quick_sorter (A: list , lo: int , hi: int ):
    if lo < hi:
        p = partition (A, lo , hi)
        quick_sorter (A, lo , p - 1)
        quick_sorter (A, p + 1, hi)

def quick_sort (A: list ):
    quick_sorter (A, 1, len(A))
```

6 sorting.c (Test Harness)

- -a : Employs all sorting algorithms.
- -h : Enables Heap Sort.
- -b : Enables Batcher Sort.

- -s : Enables Shell Sort.
- -q : Enables Quicksort.
- -r seed : Set the random seed to seed. The default seed should be 13371453.
- -n size : Set the array size to size. The default size should be 100.
- -p elements : Print out elements number of elements from the array. The default number of elements to print out should be 100. If the size of the array is less than the specified number of elements to print, print out the entire array and nothing more.
- -H : Prints out program usage. See reference program for example of what to print.

7 set.c

7.1 Set empty(void)

This function is used to return an empty set. In this context, an empty set would be a set in which all bits are equal to 0.

7.2 Set universal(void)

This function is used to return a set in which every possible member is part of the set. Set

7.3 Set insert(Set s, uint8_t x)

This function inserts x into s. That is, it returns set s with the bit corresponding to x set to 1. Here, the bit is set using the bit-wise OR operator. The first operand for the OR operation is the set s. The second operand is value obtained by left shifting 1 by x number of bits.

7.4 Set remove(Set s, uint8_t x)

This function deletes (removes) x from s. That is, it returns set s with the bit corresponding to x cleared to 0. Here, the bit is cleared using the bit-wise AND operator. The first operand for the AND operation is the set s. The second operand is a negation of the number 1 left shifted to the same position that x would occupy in the set. This means that the bits of the second operand are all 1s except for the bit at x's position. The function returns set s after removing x.

7.5 bool member(Set s, uint8_x)

This function returns a bool indicating the presence of the given value x in the set s. The bit-wise AND operator is used to determine set membership. The first operand for the AND operation is the set s. The second operand is the value obtained by left shifting 1 x number of times. If the result of the AND operation is a non-zero value, then x is a member of s and true is returned to indicate this. false is returned if the result of the AND operation is 0.

7.6 Set union(Set s, Set t)

The union of two sets is a collection of all elements in both sets. Here, to calculate the union of the two sets s and t, we need to use the OR operator. Only the bits corresponding to members that are equal to 1 in either s or t are in the new set returned by the function.

7.7 Set intersect(Set s, Set t)

The intersection of two sets is a collection of elements that are common to both sets. Here, to calculate the intersection of the two sets s and t, we need to use the AND operator. Only the bits corresponding tomembers that are equal to 1 in both s and t are in the new set returned by the function.

7.8 Set difference(Set s, Set t)

The difference of two sets refers to the elements of set s which are not in set t. In other words, it refers to the members of set s that are unique to set s. The difference is calculated using the AND operator where the two operands are set s and the negation of set t. The function then returns the set of elements in s that are not in t.

7.9 Set complement(Set s)

This function is used to return the complement of a given set. By complement we mean that all bits in the set are flipped using the NOT operator. Thus, the set that is returned contains all the elements of the universal set U that are not in s and contains none of the elements that are present in s.