Sorting: Design

Lucais Sanderson

30 January 2023

1 Description of Program

A program that tests several different sorting techniques, comparing their respective efficiencies.

2 Pseudocode / Structure:

- shell.c

```
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]
                swap_count++
                j -= gap
            arr[j] = temp
```

- heap.c

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

fix_heap(arr[], first, last)
    found = False
    mother = first
    great = max_child(arr, mother, last)

    while mother <= last // 2 and not found:
```

```
        if arr[mother - 1] < arr[great - 1]:
            # SWAP
            arr[mother - 1], arr[great - 1] = arr[great - 1], arr[mother - 1]
            mother = great
            great = max_child(arr, mother, last)
        else:
            found = True

build_heap(arr[], first, last):
    for father in range(last // 2, first - 1, -1)
        fix_heap(arr, father, last)

heap_sort(arr)
    forst = 1
    last = len(arr)
    build_heap(arr, first, last)
    for leaf in range(last, first, -1)
        #SWAP
        arr[first - 1], arr[leaf - 1] = arr[leaf - 1], arr[first - 1]
        fix_heap(arr, first, leaf - 1)
```

- quick.c

```
### PARTITION
partition(arr[], lo, hi):
    i = lo - 1
    for j in range(lo, hi):
        if arr[j - 1] < arr[hi - 1]
            i ++
            arr[i - 1], arr[j - 1] = arr[j - 1], arr[i - 1]
    arr[i], arr[hi - 1] = arr[hi - 1], arr[i]
    return i + 1


### RECURSIVE QUICKSORT
quick_sorter(arr[], lo, hi):
    if lo < hi
        p = partition(arr, lo, hi)
        quick_sorter(arr, lo, p - 1)
        quick_sorter(arr, p + 1, hi)

quick_sort(arr[]):
    quick_sorter(arr, 1, len(arr))
```

- batcher.c

```
bit_length(a)
    b = 0
    for range(a), rightshift a
        b += 1
    return b

comparator(arr[], x, y):
    if arr[x] > arr[y]
        arr[x], arr[y] = arr[y], arr[x]

batcher_sort(arr[]):
    if len(arr) == 0
        return #(NULL?)

    n = len(arr)
    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(arr, i, i + d)
            d = q - p
            q >>= 1
            r = p

        p >>= 1
```

- sorting.c

```
get argc / argv
parse options using getopt
use a set to mark certain chosen options

using the set, run given marked sorts with
tertiary options if set i.e. number of elements, seed,...

print stats, sorted list, etc
```

3 Credit

All of this pseudocode is derived from the spec document.