CMSC351 Spring 2025 (§0101,§0201,§0301) Homework 5

Due Thursday Mar 13, 2025 by 23:59 on Gradescope.

Directions:

- Homework must be done on printouts of these sheets and then scanned properly, or via latex, or by downloading, writing on the PDF, and uploading.
- Do not use your own blank paper!
- The reason for this is that gradescope will be following this template to locate the answers to the problems so if your answers are organized differently they will not be recognized.
- Tagging is automatic, do not manually tag.

---

1. Suppose we start with a max heap (a 1-indexed list `A`) containing the following elements not in the order given: [12 pts]

   $$\{10, 20, 30, 40, 50, 60, 70\}$$

   We swap `A[1]` and `A[7]`, chop off `A[7]`, then run `maxheapify` on `A[0]`.

   Suppose we know what `A` looks like after this process and we know where `20` was before this process. What the rest of it look like before?

Index	1	2	3	4	5	6	7
`A`, before process	70	60	30	40	50	10	20
`A`, after process	60	50	30	40	20	10	–

   Put scratch work below; Scratch work is not graded:

2. Here is the pseudo-code for `MergeSort` and `Merge` with some `print` statements added:

```
function MergeSort(arr,start,end)
    if start < end:
        print "HI"
        middle = (start+end) // 2
        MergeSort(arr,start,middle)
        MergeSort(arr,middle+1,end)
        Merge(arr,start,middle,middle+1,end)
        print(arr)
    end if
end function
function Merge(arr,start1,end1,start2,end2)
    temp = array of same size as arr
    i = start1; j = start2; k = start1
    while i <= end1 and j <= end2:
        if arr[i] <= arr[j]:
            temp[k] = arr[i]; i++; k++
        else:
            temp[k] = arr[j]; j++; k++
        end if
    end while
    while i <= end1:
        temp[k] = arr[i]; i++; k++
    end while
    while j <= end2:
        temp[k] = arr[j]; j++; k++
    end while
    for i = start1 to end2 inclusive:
        print "HELLO"
        arr[i] = temp[i]
    end for
end function
```

(a) If we call `MergeSort([99,44,55,88,33,22,11]` the `print(arr)` statement will run exactly 6 times. What will it print each time? [12 pts]

First Time	44	99	55	88	33	22	11
Second Time	44	99	55	88	33	22	11
Third Time	44	55	88	99	33	22	11
Fourth Time	44	55	88	99	22	33	11
Fifth Time	44	55	88	99	11	22	33
Sixth Time	11	22	33	44	55	88	99

(b) When we run the code above on a list of length 7: [4 pts]

Number of times `HI` will be output?	6
Number of times `HELLO` will be output?	20

(c) When we run the code above on a list of length 4^k, how many times will `HI` be printed? Simplify your answer. [6 pts]

Number of times `HI` will be output?	$4^k - 1$

(d) Explain your answer to (c) [10 pts]

Solution:

`HI` is output every time there is a recursive call on a sub-list of at least 2 elements. We have a list of length $4^k = 2^{2k}$.

- 1 time on the first call of the list of length 2^{2k}.
- 2 times total on the recursive calls to the sublists of length $2^{2k}/2 = 2^{2k-1}$.
- 4 times total on the recursive calls to the sublists of length $2^{2k}/4 = 2^{2k-2}$.
- In general 2^i times total on the recursive calls to the sublists of length 2^{2k-i}.
- 2^{2k-1} times total on the recursive calls to the sublists of length 2^1

Thus the total is:

$$\sum_{i=0}^{2k-1} 2^i = 2^{2k} - 1 = 4^k - 1$$

3. Here is the pseudocode for Heapsort with a `print` statement added: [15 pts]

```
function heapsort(A,n)
    converttomaxheap(A,n)
    for i = n down to 2 inclusive:
        swap(A[1],A[i])
        maxheapify(A,1,i-1)
        print(A)
    end
end
```

Suppose we have just finished `converttomaxheap` and we have:

`A = [9,7,5,1,4,2]`

What will `A` look like after each time it is printed?

Starting `A =`	9	7	5	1	4	2
After the first time, `A =`	7	4	5	1	2	9
After the second time, `A =`	5	4	2	1	7	9
After the third time, `A =`	4	1	2	5	7	9
After the fourth time, `A =`	2	1	4	5	7	9
After the fifth time, `A =`	1	2	4	5	7	9

Put scratch work below; Scratch work is not graded:

4. Suppose we have a perfect binary tree with height $h \geq 0$ representing a heap, meaning it has $n = 2^{h+1} - 1$ keys indexed from 1 to $2^{h+1} - 1$. When we run `convertomaxheap` we run `maxheapify` in reverse order on every key with children.

   Let's examine the worst-case - In the worst-case every single key gets swapped all the way to the leaf level.

   (a) For each level in the tree there are a certain number of nodes and each of those nodes requires a certain number of swaps. Fill in the appropriate values/expressions in the table: [10 pts]

Level	Number of Keys	Number of Swaps per Key
0	1	h
1	2	$h-1$
2	4	$h-2$
$\vdots$	$\vdots$	$\vdots$
i	2^i	$h-i$
$\vdots$	$\vdots$	$\vdots$
h	2^h	0

   (b) Write down a sum for the total number of swaps required. This should involve h, not n. [10 pts]

Total	$\sum_{i=0}^{h} 2^i(h-i)$

(c) Evaluate this sum and then rewrite this as a function of n. You will probably have to look up an unfamiliar sum formula for this. Simplify this as much as possible - it will be very simple! [11 pts]

Solution:

We have:

$$\begin{aligned}\sum_{i=0}^{h} 2^i(h-i) &= h\sum_{i=0}^{h} 2^i - \sum_{i=0}^{h} i2^i \\ &= h\left(2^{h+1}-1\right) - \left[(h-1)2^{h+1}+2\right] \\ &= h\cdot 2^{h+1} - h - h\cdot 2^{h+1} + 2^{h+1} - 2 \\ &= 2^{h+1} - h - 2\end{aligned}$$

Since $n = 2^{h+1} - 1$ we have $2^{h+1} = n+1$ and $h = \lg(n+1) - 1$ and so we continue:

$$\begin{aligned}&= 2^{h+1} - h - 2 \\ &= n + 1 - (\lg(n+1) - 1) - 2 \\ &= n - \lg(n+1)\end{aligned}$$

5. Let's see why Heapsort is unstable. Consider the list `[2,2',1]`. Because the length is 3 we know there is a call to `converttomaxheap` followed by two sets of swap, chop, and `maxheapify`.

(a) What will this list look like after `convertomaxheap` is run on it? [2 pts]

Index	1	2	3
Value	2	2'	1

(b) What will this list look like after the first swap and chop? [2 pts]

Index	1	2	3
Value	1	2'	2

(c) What will this list look like after the subsequent call to `maxheapify`? [2 pts]

Index	1	2	3
Value	2'	1	2

(d) What will this list look like after the second swap and chop? [2 pts]

Index	1	2	3
Value	1	2'	2

(e) What will this list look like after the subsequent call to `maxheapify`? [2 pts]

Index	1	2	3
Value	1	2'	2