EL9343 Homework 2 Solutions

---

1. If we divide the original array A into 3 equal-sized sub-arrays S1, S2, and S3, we have 3 special cases to consider in the *combine* phase of the divide-and-conquer algorithm:

(a) head in S1, tail in S2 : linear sweep in S1 towards A.head + linear sweep in S2 towards A.tail + summation = $\Theta(2n/3) + \Theta(1)$

(b) head in S2, tail in S3 : linear sweep in S2 towards A.head + linear sweep in S3 towards A.tail + summation = $\Theta(2n/3) + \Theta(1)$

(c) head in S1, tail in S3 : re-use the linear sweep in S1 towards A.head and the linear sweep in S3 towards A.tail, so just summation = $\Theta(1)$

Since the idea in the *divide* and *conquer* phases is the same as before, we have

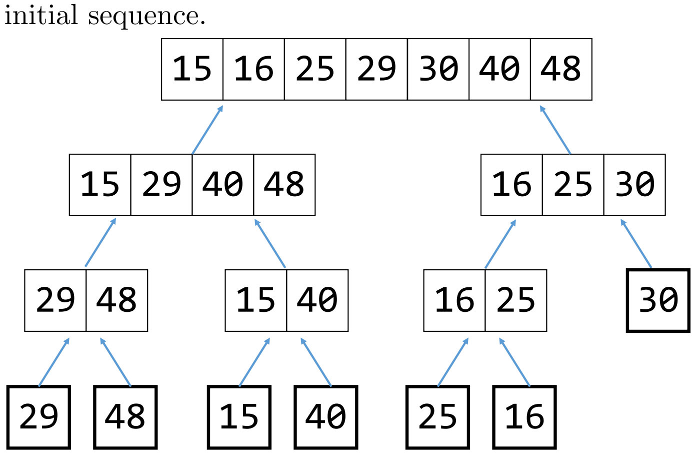
$$T(n) = 3T(n/3) + \Theta(4n/3),$$

which results in $T(n) = \Theta(n \log n)$.

2. **Insertion sort** on [11, 8, 7, 5, 3, 1]:
1. [8, 11, 7, 5, 3, 1]
2. [8, 7, 11, 5, 3, 1]→[7, 8, 11, 5, 3, 1]
3. [7, 8, 5, 11, 3, 1]→[7, 5, 8, 11, 3, 1]→[5, 7, 8, 11, 3, 1]
4. [5, 7, 8, 3, 11, 1]→[5, 7, 3, 8, 11, 1]→[5, 3, 7, 8, 11, 1]→[3, 5, 7, 8, 11, 1]
5. [3, 5, 7, 8, 1, 11]→[3, 5, 7, 1, 8, 11]→[3, 5, 1, 7, 8, 11]→[3, 1, 5, 7, 8, 11]→[1, 3, 5, 7, 8, 11]

Bubble sort on [11, 8, 7, 5, 3, 1]:
1. [11, 8, 7, 5, 1, 3]→[11, 8, 7, 1, 5, 3]→[11, 8, 1, 7, 5, 3]→[11, 1, 8, 7, 5, 3]→[1, 11, 8, 7, 5, 3]
2. [1, 11, 8, 7, 3, 5]→[1, 11, 8, 3, 7, 5]→[1, 11, 3, 8, 7, 5]→[1, 3, 11, 8, 7, 5]
3. [1, 3, 11, 8, 5, 7]→[1, 3, 11, 5, 8, 7]→[1, 3, 5, 11, 8, 7]
4. [1, 3, 5, 11, 7, 8]→[1, 3, 5, 7, 11, 8]
5. [1, 3, 5, 7, 8, 11]

3. Let's assume the first partition has size $\lceil a.size/2 \rceil$. The bold cells at the bottom depict the initial sequence.



4. (Problem 2-1 in CLRS)

(a) Each sub-array of length k can be sorted in $\Theta(k^2)$ time in worst-case. There are n/k sub-arrays in total, so the time complexity is $\Theta(\frac{n}{k}k^2) = \Theta(nk)$.

(b) Time complexity $T(N)$ of merging N arrays of length k can be written recursively as

$$T(N) = 2T(N/2) + \Theta(Nk).$$

Here, we know that the complexity of the *combine* phase is linear in the sum of the lengths of the sub-arrays, which is Nk. From the master theorem, we have

$$T(N) = \Theta(Nk \log N).$$

Plugging $N = n/k$ gives the result.

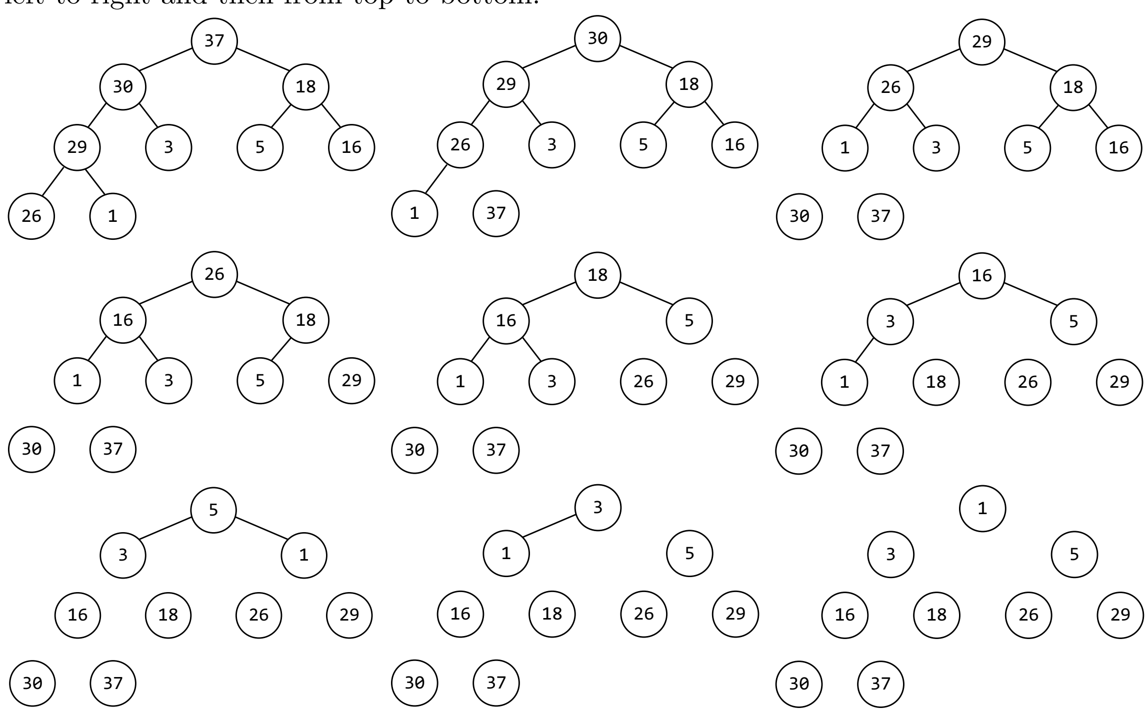
(c) We want $\Theta(n \log n) = \Theta(nk + n\log(n/k)) = \Theta(n(k + \log n - \log k))$. Here, k already dominates $\log k$. Also, if k grows faster than $\log n$, it will also dominate the $\log n$ term. Therefore, it should at most grow with $\log n$.

(d) $k = \log n$

5. (Exercise 6.1-3 in CLRS) From the definition of a tree, for any arbitrary node i, there is a unique path that connects it with the root. Due to the max-heap property, each node is smaller than its parent on that path, therefore the root is greater than node i. The argument is true for any i, so the root must be the greatest.

6. (Exercise 6.2-6 in CLRS) The maximum number of recursions of max-heapify is the height of the heap, which happens when the smallest element in the array sits at the root. Since the height of a heap of length n is $\lfloor \log n \rfloor$, the time complexity of max-heapify is $\Omega(\log n)$.

7. First heap is created by calling build-max-heap. The diagrams should be followed first from left to right and then from top to bottom.



8. (Problem 6-2 in CLRS)

(a) The nodes on the heap are put in the array starting iteratively, starting from depth 0 up to depth $\lfloor \log_d n \rfloor$ and from left to right in each depth. The index of the m^{th} child of node i is given by $di + m$, assuming indexing starts from 0, while the index of the parent of node i is given by $\lfloor (i-1)/d \rfloor$.

(b) $H = \lfloor \log_d n \rfloor$

(c) (d) (e) Only max-heapify has to be modified, the rest is the same. extract-max runs in $O(d \log_d(n))$, the other two run in $O(\log_d(n))$.

Algorithm 1 max-heapify(a,i)

```
largest = i
for m = 1 : d do
    child = di+m
    if child ≤ a.size and a[child] ≥ a[largest] then
        largest = child
    end if
end for
if largest ≠ i then
    swap( a[i], a[largest] )
    max-heapify( a, largest )
end if
```

9. **Lomuto's partition** on $A =$[13, 19, 9, 12, 8, 7, 5, 4, 2, 6, 11, 14]

0. $x = A[end] = 14,\ i = -1$
1. $j = 0$, $i = 0$, [13, 19, 9, 12, 8, 7, 5, 4, 2, 6, 11, 14]
2. $j = 1$, $i = 0$, [13, 19, 9, 12, 8, 7, 5, 4, 2, 6, 11, 14]
3. $j = 2$, $i = 1$, [13, 9, 19, 12, 8, 7, 5, 4, 2, 6, 11, 14]
4. $j = 3$, $i = 2$, [13, 9, 12, 19, 8, 7, 5, 4, 2, 6, 11, 14]
5. $j = 4$, $i = 3$, [13, 9, 12, 8, 19, 7, 5, 4, 2, 6, 11, 14]
6. $j = 5$, $i = 4$, [13, 9, 12, 8, 7, 19, 5, 4, 2, 6, 11, 14]
7. $j = 6$, $i = 5$, [13, 9, 12, 8, 7, 5, 19, 4, 2, 6, 11, 14]
8. $j = 7$, $i = 6$, [13, 9, 12, 8, 7, 5, 4, 19, 2, 6, 11, 14]
9. $j = 8$, $i = 7$, [13, 9, 12, 8, 7, 5, 4, 2, 19, 6, 11, 14]
10. $j = 9$, $i = 8$, [13, 9, 12, 8, 7, 5, 4, 2, 6, 19, 11, 14]
11. $j = 10$, $i = 9$, [13, 9, 12, 8, 7, 5, 4, 2, 6, 11, 19, 14]
12. final swap $\rightarrow$ [13, 9, 12, 8, 7, 5, 4, 2, 6, 11, 14, 19]
13. return $i + 1$

In the recursive call, quicksort will then sort up to index i.

Hoare's partition on [13, 19, 9, 12, 8, 7, 5, 4, 2, 6, 11, 14]

0. $x = A[0] = 13,\ i = -1,\ j = 12$
1. $j = 10$, $i = 0$, swap $\rightarrow$ [11, 19, 9, 12, 8, 7, 5, 4, 2, 6, 13, 14]
2. $j = 9$, $i = 1$, swap $\rightarrow$ [11, 6, 9, 12, 8, 7, 5, 4, 2, 19, 13, 14]
4. $j = 8$, $i = 9$, break and return j

In the recursive call, quicksort will then sort up to index j.