Exercises, chapter 12, solutions

---

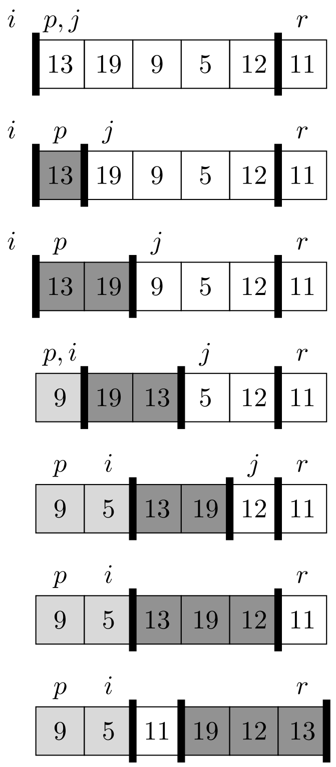
1. **Algorithm:**
   First, apply merge sort to A. Next, traverse the sorted array while checking if any two consecutive elements are equal; if so, answer "yes". If no consecutive elements were equal, answer "no".

   Time complexity analysis:
   Since A contains n elements, doing a merge sort takes $O(n \lg n)$ time. Next, traversing the sorted array and comparing the adjacent elements to each other takes $O(n)$ time in total. The overall time complexity is $O(n \lg n) + O(n) = O(n \lg n)$.

2. The following algorithm uses the technique from `ArrayMerge` in chapter 12.

```
let m = |A|, n = |B|, A[m+1] = ∞, B[n+1] = ∞     // Insert sentinels at the end of A and B
i = 1
j = 1
while ((i ≤ m) or (j ≤ n))
    if (A[i] < B[j]) then i = i + 1
    else if (A[i] > B[j]) then j = j + 1
    else     // (A[i] == B[j])
        tmp = A[i]
        output tmp
        while (A[i] == tmp)
            i = i + 1
```

   Time complexity analysis:
   The algorithm runs in $O(|A|+|B|)$ time because indices i and j start at 1 and are incremented by 1 a total of $O(|A| + |B|)$ times, and $O(1)$ operations are carried out between any two consecutive increments.

3. 

4. As an example, consider an array $A = \langle 5, 5, 8, 4 \rangle$ of keys for four records. To distinguish between the first and the second occurrence of the key 5, denote them by $5'$ and $5''$, respectively, i.e., write $A = \langle 5', 5'', 8, 4 \rangle$. The first iteration of selection sort will swap $5'$ and 4, which yields $\langle 4, 5'', 8, 5' \rangle$. In the second iteration, $5''$ will stay where it is. Finally, in the third iteration, 8 and $5'$ will be swapped, giving $\langle 4, 5'', 5', 8 \rangle$. The output array will be $\langle 4, 5, 5, 8 \rangle$, but the relative order between the two records having the key 5 was not preserved. This shows that selection sort is not a stable sorting method.

5. By the definition of a max-heap, the largest integer in H is stored in the root. Moreover, the second largest integer is in a child of the root, and the third largest integer is in either a child of this node or in the other child of the root. Hence, we can use the following method:

   - Do a pairwise comparison between the integers stored in the root's two children. Let a be the child of the root containing the larger integer and b the other child.
   - Next, do a comparison between the integers stored in the two children of a. Let c be the child of a containing the larger integer.
   - Finally, do a comparison between the two integers stored in b and c, and output the larger one.

   In total, three pairwise comparisons are used.