CS3230 Homework 1

Qi Ji

26th October 2018

1 K-sorted Array

1.1

Statement 1.1.1. Fix j, suppose for any $i \in \{1, \dots, j-k-1\}$, $B[i]$ contains the i-th smallest element of A. Then the value extracted from the heap will be the $(j-k)$-th smallest element of A.

1.2

For any $i \in \{1, \dots, n\}$, we let $M(i)$ denote the i-th smallest element of A.

Proof. We first observe that the elements

$$X := \{A[1], \dots, A[k], \dots, A[\max(j, n)]\}$$

have been added to the heap S in previous (if any) and current iterations of the **for** loop. Since A is k-sorted, $M(j-k) \in \{A[1], \dots, A[\max(j-k+k, n)]\} = X$. By our assumption that $B[i]$ contains the i-th smallest element of A for each $i \in \{1, \dots, j-k-1\}$. We see that the elements

$$Y := \{M(1), \dots, M(j-k-1)\}$$

have already been extracted from S in previous iterations. As A contains distinct integers, we see that $M(j-k) \notin Y$. Now we see that the heap S contains precisely $X \setminus Y$. All elements less than $M(j-k)$ are not in S, so $M(j-k)$ is minimal in S, and it will be the extracted value. $\square$

1.3

Proof. Proceed by induction on $j-k$. Applying Statement 1.1.1 with $j = k+1$ proves the base case that $B[1]$ will contain the smallest element of A. Similarly, Statement 1.1.1 proves the inductive case. This means for every $i \in \{1, \dots, n\}$, $B[i] = M(i)$ so in particular, B contains the elements of A in sorted order. $\square$

2 Inversions

2.1

Solution. Given the array $\langle 2, 3, 8, 6, 1\rangle$. The inversions are $(1, 5), (2, 5), (3, 4)$, $(3, 5), (4, 5)$. ■

2.2

Solution. The array given by $\langle n, n-1, \dots, 1\rangle$ has inversion count $\binom{n}{2}$. ■

2.3

Algorithm 1: Counting inversions with modified mergesort.

```
Data: an array A[1, ..., n] containing a permutation of the n elements
Result: the number of inversions in A
inversions ← 0
subroutine modified-merge (left, mid, right) is
    Data: indices of start of left subarray, start of right subarray and end
          of right subarray, where both subarrays sorted
    Result: two subarrays merged, inversions incremented
    Initialise array B[left, ..., right]
    i ← left; j ← mid
    k ← left
    while k ≤ right do
        if i < mid ∧ (j > right ∨ A[i] < A[j]) then
            B[k] ← A[i]
            i ← i + 1
        else
            B[k] ← A[j]
            j ← j + 1
            inversions ← inversions + (mid − i)
        k ← k + 1
    copy B[left, ..., right] into A[left, ..., right]
function mergesort (L, R) is
    Data: start L and end R indices of subarray to mergesort
    if L = R then
        return
    M ← ⌊(L+R)/2⌋ + 1
    mergesort(L, M − 1)
    mergesort(M, R)
    modified-merge(L, M, R)
mergesort (1, n)
output inversions
```