ESO207: Data Structures and Algorithms
Abhimanyu Sethia
Roll Number: 190023
Section: S9

Assignment 1

Abhimanyu Sethia

Q1 Recall from lecture 2, the definition of inversionsand number of inversions in an array. It is straightforward to compute the number of inversions in an array in $O(n^2)$ time. Using divide and conquer approach, as used in merge sort, an $O(nlogn)$ time algorithm can be designed for this problem
(a) *Write the pseudo-code for an O(n log n) time algorithm for computing the number of inversions in an array.*

```
mergeAndCount(A, i, k, j)
    p=i, q=k+1, r=i, inv =0, count =0
    φ1
    while r≤j do
        φ2
        if p≤k and q≤j and A[p]>A[q]
            B[count] = A[q]
            q=q+1, count= count+1, r=r+1, inv=k-p+1, continue
        if p≤k and q≤j and A[p]≤A[q]
            B[count] = A[p]
            p=p+1, count= count+1, r=r+1, continue
        if p≤k and q>j
            B[count] = A[p]
            p=p+1, count=count+1, r=r+1, continue
        if p>k and q≤j
            B[count] = A[q]
            q=q+1, count= count+1, r=r+1, continue
        φ3
    //endwhile
    φ4
    count=0, r=i
    while r<j do
        A[r]=B[count]
        count= count+1, r=r+1
    //endwhile
    return inv

countInv(A, i, j)
    totInv = 0
    if j>i
        k=i+⌊(j−i)/2⌋
```

```
        totInv = totInv + countInv(A,i,k)
        totInv = totInv + countInv(A,k+1,j)
        totInv = totInv + mergeAndCount(A,i,k,j)
    return totInv
```

(b) *Prove correctness of your algorithm, using loop invariant technique.Your proof may be informal but it should be precise and clear.*

The following proof uses the term cross inversion hereafter. **Cross inversions** between the arrays A[i,k] and A[k+1,j] is defined as the set of pairs (p,q) such that p ϵ[i,k] and q ϵ[k+1,j] and A[p]>A[q].

Objective of MergeAndCount

Input: Array A (pointer),i,k,j such that i≤k≤j and A[i,k] and A[k+1,j] are sorted arrays
Output:

- A[i,j] must be sorted
- A[i,j] should be a permutation of A[i,k]∪A[k+1,j]
- inv = number of cross inversions between A[i,k] and A[k+1,j] that is number of pairs (x,y) such that x ϵ[i,k] and y ϵ[k+1,j] and A[x]>A[y]

Loop Invariants

(i) i≤p≤k+1
k+1 ≤q ≤ j+1
i ≤r ≤ j+1
r=p+q-k-1

(ii) inv = number of cross inversions between A[i,k] and A[k+1,q] (i.e. number of pairs (x,y) such that xϵ[i,k] and yϵ[k+1,q-1] and A[x]>A[y])

(iii) B[i,r-1] is sorted

(iv) B[i,r-1] is a permutation of A[i,p-1]∪A[k+1,q-1]

(v) B[i,r-1]≤ A[p], A[q] if p≤k, q≤j
B[i,r-1]≤ A[q] if p>k, q≤j
B[i,r-1]≤ A[p] if p≤k, q>j

The approach followed for merge-sorting the arrays in the above algorithm, is identical to the one followed in Lecture 4. Hence, we will assume the correctness of Merge sorting in the algorithm and will not repeat the proof (which is given in Lecture 4).
Since the proof for correctness of loop invariants (iii),(iv) and (v) was also done in Lecture 4 (which we will not repeat here), we only need to prove that (i) and (ii) are loop invariants and thereby, prove that the number of cross inversions are rightly counted, as per our objective

Correctness of Loop Invariants

To prove that the loop invariants (i) and (ii) are correct, we need to prove that

(A) (i) and (ii) hold at ϕ_1 (i.e. before the while loop)

(B) if $\phi \wedge \neg$c holds at ϕ_2 (i.e. before the body of the loop),
then ϕ holds at ϕ_3 (i.e. after the body of the loop)
(Here ϕ is the loop invariant and c is the guard condition)

(A) (i), (ii) must hold at ϕ_1

(i) At ϕ_1
p=i $\implies$ i≤p≤k+1
q=k+1 $\implies$ k+1≤q≤j+1
r=i $\implies$ i≤r≤j+1
p+q-k-1 = (i)+(k+1)-k-1 = i = r
Hence, (i) holds true at ϕ_1

(ii) At ϕ_1
q=k=1 $\implies$ A[k+1,q] is an empty array
Hence, number of cross inversions between A[i,k] and A[k+1,q-1] is trivially, 0.
Also, inv =0
Hence, (ii) holds true at ϕ_1

(B) Let p=p_0, q=q_0, r=r_0, inv=inv_0 and count=c_0 at ϕ_2.
Assume that (i), (ii) and r_0 hold true at ϕ_2.

(i) By our assumption at ϕ_2,
i≤ $p_0 \leq k+1$
$i \leq r_0 \leq j+1$
$k+1 \leq q_0 \leq j+1$
$r_0 = p_0 + q_0 - k_0 - 1$

Since $p_0 \leq k$ and $q_0 \leq j$ by our assumption at ϕ_2,
$\implies p_0 + 1 \leq k+1, q_0 + 1 \leq j+1$.
But at ϕ_3, either $p = p_0 + 1$ or $q = q_0 + 1$
Hence, at ϕ_3,i≤ $p \leq k+1, k+1 \leq q \leq j+1$ hold true.

Since $r_0 \leq j$ is true (loop guard)
so $r_0 + 1 \leq j + 1$
But at ϕ_3, r= $r_0 + 1$,
$\implies i \leq r \leq j+1$

From our assumption, $r_0 = p_0 + q_0 - k - 1$. At ϕ_3, r= $r_0 + 1$ and one of the two is true: either $p = p_0 + 1$ or $q = q_0 + 1$
Hence, if $p = p_0 + 1$, then $r_0 + 1 = (p_0 + 1) + q_0 - k - 1 \implies r = p + q - k - 1$
if $q = q_0 + 1$, then $r_0 + 1 = p_0 + (q_0 + 1) - k - 1 \implies r = p + q - k - 1$

Hence, (i) holds at ϕ_3

(ii) By our assumption at ϕ_2, inv_0= number of cross inversions between A[i,k] and A[k+1,$q_0 - 1$]

If second or third if condition is implemented,
no additional inversion is found.
Hence, at ϕ_3, q=q_0 and inv = inv_0,
$\implies$ inv = number of cross inversions between A[i,k] and A[k+1,q-1] (as follows from our assumption)

If fourth if condition is implemented
then by the if condition, p_0 = k+1 $\implies$ $r_0 \geq k+1$.
But since A[i,r_0] is sorted,
so no additional inversions which involve q_0 exist .
inv_0 = number of cross inversions between A[i,k] and A[k+1,q_0].
At ϕ_3, $q = q_0 + 1$,
It follows that inv= number of cross inversions between A[i,k] and A[k+1,q-1]

If first if statement is implemented,
then A[p_0]>A[q_0].
But since A[i,k] is sorted,
$\implies$ A[x] > A[q_0] $\forall x \epsilon [p_0, k]$.
Hence, number of cross inversions involving q_0 = k- p_0 +1.
So, inv_0+k- p_0 +1= number of cross inversions between A[i,k] and A[k+1,q_0]
But at ϕ_3, since q=q_0+1 and inv=inv_0+k- p_0 +1,
$\implies$ inv= number of cross inversions between A[i,k] and A[k+1,q-1]

Hence, (ii) holds at ϕ_3.

Since both (A) and (B) have been proven, we have proven that (i) and (ii) are correct valid loop invariants.

Correctness of MergeAndCount

Since all the loop invariants are correct, they must hold at ϕ_4.
$\implies$ $i \leq r \leq j+1$
But also at ϕ_4, r>j (as the while loop is exited)
Hence, it follows that r=j+1.
Also, since p≤k+1 and q≤j+1 (from loop invariant (i)) and since r=j+1= p+q-k-1,
it follows that p=k+1 and q=j+1.
Hence, by applying the loop invariants (ii), we get inv = number of cross inversions between A[i,k] and A[k+1,q-1]
but since q=j+1, we have inv = number of cross inversions between A[i,k] and A[k+1,j].
The proof that mergeAndSort correctly sorts is already done in Lecture 4.
Since all our objectives have been satisfied by the function mergeAndCount, we have proven the correctness of mergeAndCount function.

Correctness of countInv

Since i ≤ k ≤j, our recursion is proper.
For a single element array, the function countInv returns 0, which is precisely the number of inversions in a singleton array. Now, we assume that the the function returns the correct number of inversions for an array A[i,i+n-1] and it follows hence, that the array returns the correct number of inversions for the array A[i,i+n]. Therefore, by induction and the correctness of mergeAndCount, the correctness of countInv follows.