Computer Science CSC263H — March 1, 2018
St. George Campus — University of Toronto

Solutions for Homework Assignment #4

Answer to Question 1. We claim that S_i equals the number of elements in $A[1\,..\,i]$ that are larger than $A[i+1]$. Indeed, by induction, at the i-th iteration of the for-loop the subarray $A[1\,..\,i]$ is sorted. Then, if there are k elements in $A[1\,..\,i]$ that are larger than $A[i+1]$, they will be in positions $A[i-k+1\,..\,i]$, and we will do a swap for $j = i, i-1, \ldots i-k+1$, and then exit the while loop.

For any two integers i and j between 1 and n, let $X_{i,j}$ be the indicator random variable which is equal to 1 if $A[i] < A[j]$ and equal to 0 otherwise. Then

$$S_i = |\{1 \le j \le i : A[i+1] < A[j]\}| = \sum_{j=1}^{i} X_{i+1,j}.$$

By linearity of expectation, $\mathbf{E}[S_i] = \sum_{j=1}^{i} \mathbf{E}[X_{i+1,j}]$. We claim that for any $i \neq j$, $\mathbf{E}[X_{i,j}] = \mathbf{P}[X_{i,j} = 1] = \frac{1}{2}$. Indeed, the number of permutations A of $\{1, \ldots, n\}$ such that $A[i] < A[j]$ is equal to the number of permutations such that $A[i] > A[j]$. To see this, observe that to any permutation such that $A[i] < A[j]$ corresponds exactly one permutation such that $A[i] > A[j]$, which we get by just swapping $A[i]$ and $A[j]$. Then, we have

$$\mathbf{E}[S_i] = \sum_{j=1}^{i} \mathbf{E}[X_{i+1,j}] = \frac{i}{2}.$$

The total number of swaps is

$$\mathbf{E}\left[\sum_{i=1}^{n-1} S_i\right] = \sum_{i=}^{n-1} \mathbf{E}[S_i] = \sum_{i=1}^{n-1} \frac{i}{2} = \frac{n(n-1)}{4}.$$

Answer to Question 2. Let $E_i = \{e_i, \ldots, e_m\}$, so that $G_i = (V, E_j)$. The edge e_i we need to output corresponds to the integer i such that G_i has a cycle, but G_{i+1} does not. (Here we use the fact that once a graph does not have any cycles, removing any of its edges cannot create a cycle.) We will use the Disjoint Set Forest Union-Find data structure to process the edges in reverse order, and stops at the first edge e_i that creates a cycle. The algorithm is as follows:

for $i = 1$ **to** n
    MAKE-SET(i)
for $i = m$ **to** 1
    $(u, v) = e_i$
    r_u = FIND(u)
    r_v = FIND(v)
    if $r_u == r_v$
        return e_i
    else UNION(r_u, r_v)
return e_1.

After processing edge e_i, the sets in the Union-Find data structure are the connected components of G_i. The algorithm returns the first edge e_i it encounters (the one with the largest i) that connects two nodes in the same connected component. Because any two nodes in the same connected component are connected by a path, e_i closes a cycle. It remains to show that G_{i+1} does not have a cycle. This follows by induction. Initially, we have an empty graph (i.e. graph with no edges), which obviously does not have any cycles. Every edge e_j with $j > i$ was between distinct connected components of G_{j+1}, or it would have been output by the algorithm. By the induction hypothesis, none of the connected components of G_{j+1} had a cycle. Therefore G_j cannot have a cycle either.

The algorithm makes n calls to Make-Set and m calls to Find. Therefore, if we use path compression and union by rank, the algorithm has total running time $O(n + m \log^* n)$.