Indian Institute of Technology Kharagpur

CS29003: Algorithms Laboratory, Spring

Assignment 0

2PM – 5PM

Submit a single C source file. Do not use global variables.

Let $a_1, a_2, \ldots, a_n$ be a permutation of the set $\{1, 2, \ldots, n\}$. Defined below are some notions about permutations.

- An *inversion* of the permutation is a pair (a_i, a_j) such that $i < j$ and $a_i > a_j$.
- The *inversion table* $b_1, b_2, \ldots, b_n$ of the permutation is defined by setting b_j as the number of inversions whose second component is j.
- The *index* of the permutation is the sum of all subscripts j such that $a_j > a_{j+1}$.

For example, the permutation $2, 5, 4, 1, 3$ has 6 inversions: $(2, 1)$, $(5, 4)$, $(5, 1)$, $(5, 3)$, $(4, 1)$ and $(4, 3)$. The inversion table is given by $3, 0, 2, 1, 0$. And the index is $2 + 3 = 5$.

Given a positive integer n, your task is to print all permutations of $1, 2, \ldots, n$ along with the number of inversions and the index for each permutation. Described below is an algorithm for the same.

(a) Write a function *init* to store $1, 2, \ldots, n$ in an array $A[1, \ldots, n]$. Define a direction (+1,−1 or 0) for each element of A and store them in a separate array $D[1, \ldots, n]$. Initialise D as follows: $D[1] = 0$ and $D[i] = -1$ for all $2 \leq i \leq n$. Note that $+1, -1$ indicate right and left respectively. Define a variable *invCount* that stores the number of inversions for the current permutation stored in A, initialised to 0. Use dynamic memory allocation to create A and D

(b) Write a function *index* that computes and returns the index for the permutation stored in A. The index can be computed by making one pass over A.

(c) Define a function *genNext* that updates A to contain the next permutation as follows: find the largest number with non-zero direction (say, $k = A[j]$); swap k with its neighbour in the indicated direction (i.e., $D[j]$). If after the swap, the larger of the two numbers swapped precedes the other in A, then increment *invCount*; otherwise decrement it.

Update D as follows: if the chosen element (k) has reached position 1 or n, or if the next element in the same direction is greater than k, then set the direction of k as 0. Also, modify the direction of all the elements greater than k with 0 direction so that they point towards k.

Return *invCount*.

(d) In the *main()* function, read n from the user, call *init* and then repeatedly do the following until all numbers in A have direction 0: call *genNext*; call *index*; print the permutation stored in A, followed by the number of inversions (returned by *genNext*) and its index in one line (appropriately spaced). (You should create pointers for A and D in *main()* and pass them to the functions appropriately.)

Sample Output 1

```
n = 3

Permutation   Inversions    Index

   1 2 3          0           0
   1 3 2          1           2
   3 1 2          2           1
   3 2 1          3           3
   2 3 1          2           2
   2 1 3          1           1
```