Indian Institute of Technology Kharagpur

CS29003: Algorithms Laboratory, Spring 2022

Assignment 1: Time Complexity of Algorithms

2PM – 5PM 18th January, 2022

General Instructions (to be followed strictly)

Submit a single C/C++ source file.
Do not use global variables unless you are explicitly instructed so.
Do not use Standard Template Library (STL) of C++.
Use proper indentation in your code and include comments.
Name your file as `<roll_no>_a1.<extn>`
Write your name, roll number, and assignment number at the beginning of your program.

A musician named Una Lakim is tired of being accused of plagiarising tunes and songs. So she embarks on creating a good melody herself, with some help from you. Starting with a set of notes labelled $1, 2, \ldots, n$, Una attempts to arrange them in an *acceptable* sequence that makes the resulting tune *musical*. An *unacceptable* sequence is defined as follows. Let $T[0, 1, \ldots, n-1]$ denote the arrray containing the sequence of notes. T contains nothing but a pertmutation of $1, 2, \ldots, n$. Let $x, y, z \in \{1, 2, \ldots, n\}$ with $x < y < z$ and let i, j, k be their respective indices in T. We say that a placement of the notes x, y, z in T is unacceptable if $i < k < j$. In other words, x, y, z appear in the order x, z, y as a subsequence of $T[0, \ldots, n-1]$. Note that they may not appear in consecutive positions. The sequence T is unacceptable (or *unmusical*) if there exists a triple x, y, z whose placement in T is unacceptable. All other sequences are acceptable. For example, the sequence $8, 5, 6, 7, 3, 4, 2, 9, 1, 10$ is musical but the sequence $8, 5, 4, 7, 3, 6, 2, 9, 1, 10$ is not ($4, 6, 7$ appear in the order $4, 7, 6$). Una wants to find out if a sequence she has generated is musical or not.

(a) Write a function *algo*1 that takes as input an array T of length n containing the sequence and for every possible triple $x, y, z \in \{1, 2, \ldots, n\}$, finds their positions i, j, k and checks whether $i < k < j$. If this happens, the function returns 0 indicating that T is unmusical. If no such unacceptable triple is found, then *algo*1 returns 1. There are $O(n^3)$ possible triples and it takes $O(n)$ time to find the corresponding indices for each triple. The running time is therefore $O(n^4)$.

(b) In this part, you will implement a slightly better algorithm *algo*2 which looks at triples of indices rather than triples of integers from $[1, n]$. For every triple of indices i, j, k ($0 \leq i < j < k \leq n-1$), check whether $S[i] < S[k] < S[j]$. If T passes any check, then it is unmusical. The running time of *algo*2 would be $O(n^3)$ (better than that of *algo*1 by a factor of n).

(c) The above method can be refined further. For every index i, let $x = T[i]$. Then consider the subsequence $T[i+1, \ldots, n-1]$ containing notes $> x$. T is musical if and only if for each i this subsequence is strictly increasing. Checking whether the subsequence corresponding to a fixed i requires one pass over $T[i+1, \ldots, n-1]$ i.e., in $O(n)$ time. Doing this for each i leads to a total running time of $O(n^2)$. Write a function *algo*3 that implements this algorithm.

(d) Write a function *algo*4 implementing an $O(n)$ time algorithm. (Hint: Stacks.)

In the *main()* function, read n and the array T. Call the four functions and print the corresponding outcomes (musical/unmusical). Assume that T has the right form i.e., it is a permutation of $1, 2, \ldots, n$.

Sample Output 1

```
n = 15

Sequence: 15 13 11 10 9 4 2 1 3 5 6 7 8 12 14

Algo 1: Musical
Algo 2: Musical
Algo 3: Musical
Algo 4: Musical
```

Sample Output 2

```
n = 15

Sequence: 12 15 8 13 5 11 6 7 3 14 4 2 9 1 10

Algo 1: Unmusical
Algo 2: Unmusical
Algo 3: Unmusical
Algo 4: Unmusical
```