ECE374 SP23 HW5

Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

Problem 1

Suppose we have a river and on either side are a number of cities numbered from 1 to n (North side: $N[1 \dots n]$, South side: $S[1 \dots n]$). The city planner wants to connect certain cities together using bridges and has a list of the desired crossings (x is a $2 \times k$ array where k is the number of planned bridges).

Unfortunately, as we know, bridges cannot cross one another over water, so the city planner must focus on building the most bridges from his plan that do not intersect. Describe an algorithm that finds the maximum number of non-intersecting bridges.

Solution

For $1 < i < n$, we sort the tuples $(N[i], S[i])$ by $N[i]$ in ascending order. Then we can find the maximum number of non-intersecting bridges by finding the longest increasing subsequence of $S[i]$ with the algorithm from **Problem 2**.

This algorithm yields valid answers since the LIS in $S[i]$ represents the LIS in both $N[i]$ (sorted) and $S[i]$ *at identical indices*, which in turn represents the maximum number of bridges whose start and end points are in strictly increasing order.

Sorting takes $O(n \log n)$ time, and finding the LIS takes $O(n^2)$ time (for filling out a $n \times n$ DP matrix). Thus, the total time complexity of our algorithm is $O(n^2)$.