Algorithm Lab

Week 5: Longest Increasing Subsequence

For a sequence $S = (a_1, a_2, \dots, a_n)$, we can make a subsequence of it by removing arbitrary elements. E.g., for a sequence $S = (a, b, c)$, we have 8 subsequences: $()$, (a), (b), (c), (a, b), (a, c), (b, c), and (a, b, c). Note that, elements should have the same order, thus (b, a) is not a subsequence of (a, b, c). We say a sequence is in increasing order if every element is bigger than or equal to previous one. We can define the longest increasing subsequence problem as followed:

Instance: A sequence S

Result: The longest increasing subsequence (or at least, its length)

Description

For convenient, we note a leading subsequence of sequence $S = (a_1, a_2, \dots, a_n)$, by one integer, i.e., $S_i = (a_1, a_2, \dots, a_i)$.

To construct a specific subsequence, we have 2 kind of elements: be removed and be kept.

We can define 3 functions to help us to find longest increasing subsequence.

- $f(i)$: Length of longest subsequence of S_i.
- $g(i)$: Length of longest subsequence of S_i that kept a_i.
- $h(i)$: specific j that can maximize $g(j)$ where $j < i$ and $a_j \le a_i$.

For $f(i)$, if a_i is kept, then answer is $g(i)$. If not, answer will be $f(i-1)$.

For $g(i)$, $g(i) = g\big(h(i)\big) + 1$.

Questions

1 Design an algorithm to find $h(i)$. Can your algorithm work in $O(\log_2 i)$ time?

2 Design an algorithm to find $f(i)$.

3 Design an algorithm to reconstruct the subsequence that length is $f(i)$.

4 Analyze space complexity and time complexity of algorithms in 1~3.

5 Solve **ALG04B** on http://oj.csie.ndhu.edu.tw/