12.4

Longest Increasing Subsequence
Sequences

**Definition**

**Sequence**: an ordered list \( a_1, a_2, \ldots, a_n \). **Length** of a sequence is number of elements in the list.

**Definition**

\( a_{i_1}, \ldots, a_{i_k} \) is a subsequence of \( a_1, \ldots, a_n \) if \( 1 \leq i_1 < i_2 < \ldots < i_k \leq n \).

**Definition**

A sequence is **increasing** if \( a_1 < a_2 < \ldots < a_n \). It is **non-decreasing** if \( a_1 \leq a_2 \leq \ldots \leq a_n \). Similarly **decreasing** and **non-increasing**.
Sequences

Example

1. Sequence: 6, 3, 5, 2, 7, 8, 1, 9
2. Subsequence of above sequence: 5, 2, 1
3. Increasing sequence: 3, 5, 9, 17, 54
4. Decreasing sequence: 34, 21, 7, 5, 1
5. Increasing subsequence of the first sequence: 2, 7, 9.
Longest Increasing Subsequence Problem

**Input**  A sequence of numbers $a_1, a_2, \ldots, a_n$

**Goal**  Find an **increasing subsequence** $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ of maximum length

**Example**

1. Sequence: 6, 3, 5, 2, 7, 8, 1
2. Increasing subsequences: 6, 7, 8 and 3, 5, 7, 8 and 2, 7 etc
3. Longest increasing subsequence: 3, 5, 7, 8
Longest Increasing Subsequence Problem

**Input**  A sequence of numbers $a_1, a_2, \ldots, a_n$

**Goal**  Find an increasing subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ of maximum length

### Example

1. Sequence: 6, 3, 5, 2, 7, 8, 1
2. Increasing subsequences: 6, 7, 8 and 3, 5, 7, 8 and 2, 7 etc
3. Longest increasing subsequence: 3, 5, 7, 8
Naïve Enumeration

Assume $a_1, a_2, \ldots, a_n$ is contained in an array $A$

```
algLISNaive(A[1..n]):
    max = 0
    for each subsequence B of A do
        if B is increasing and $|B| > max$ then
            max = $|B|$
    Output max
```

Running time: $O(n2^n)$.

$2^n$ subsequences of a sequence of length $n$ and $O(n)$ time to check if a given sequence is increasing.
Naïve Enumeration

Assume $a_1, a_2, \ldots, a_n$ is contained in an array $A$

```
algLISNaive(A[1..n]):
    max = 0
    for each subsequence $B$ of $A$ do
        if $B$ is increasing and $|B| > max$ then
            max = $|B|$
    Output max
```

Running time: $O(n2^n)$.

$2^n$ subsequences of a sequence of length $n$ and $O(n)$ time to check if a given sequence is increasing.
Naïve Enumeration

Assume $a_1, a_2, \ldots, a_n$ is contained in an array $A$

```
algLISNaive(A[1..n]) :
    max = 0
    for each subsequence $B$ of $A$ do
        if $B$ is increasing and $|B| > max$ then
            max = $|B|
    Output max
```

Running time: $O(n2^n)$.

$2^n$ subsequences of a sequence of length $n$ and $O(n)$ time to check if a given sequence is increasing.
Recursive Approach: Take 1

**LIS**: Longest increasing subsequence

Can we find a recursive algorithm for **LIS**?

**LIS**(\(A[1..n]\)):

1. **Case 1**: Does not contain \(A[n]\) in which case
   \[\text{LIS}(A[1..n]) = \text{LIS}(A[1..(n-1)])\]

2. **Case 2**: contains \(A[n]\) in which case \(\text{LIS}(A[1..n])\) is not so clear.

**Observation**

For second case we want to find a subsequence in \(A[1..(n-1)]\) that is restricted to numbers less than \(A[n]\). This suggests that a more general problem is

\(\text{LIS\_smaller}(A[1..n], x)\) which gives the longest increasing subsequence in \(A\) where each number in the sequence is less than \(x\).
Recursive Approach: Take 1

LIS: Longest increasing subsequence

Can we find a recursive algorithm for LIS?

LIS\((A[1..n])\):

1. Case 1: Does not contain \(A[n]\) in which case
   \[LIS(A[1..n]) = LIS(A[1..(n - 1)])\]
2. Case 2: contains \(A[n]\) in which case \(LIS(A[1..n])\) is not so clear.

Observation

For second case we want to find a subsequence in \(A[1..(n - 1)]\) that is restricted to numbers less than \(A[n]\). This suggests that a more general problem is \(LIS\text{-smaller}(A[1..n], x)\) which gives the longest increasing subsequence in \(A\) where each number in the sequence is less than \(x\).
Recursive Approach: Take 1

LIS: Longest increasing subsequence

Can we find a recursive algorithm for LIS?

LIS($A[1..n]$):

1. **Case 1**: Does not contain $A[n]$ in which case
   \[
   LIS(A[1..n]) = LIS(A[1..(n-1)])
   \]

2. **Case 2**: contains $A[n]$ in which case $LIS(A[1..n])$ is not so clear.

**Observation**

For second case we want to find a subsequence in $A[1..(n-1)]$ that is restricted to numbers less than $A[n]$. This suggests that a more general problem is $\text{LIS\_smaller}(A[1..n], x)$ which gives the longest increasing subsequence in $A$ where each number in the sequence is less than $x$. 
Recursive Approach: Take 1

LIS: Longest increasing subsequence

Can we find a recursive algorithm for LIS?

LIS($A[1..n]$):
1. Case 1: Does not contain $A[n]$ in which case
   \[ \text{LIS}(A[1..n]) = \text{LIS}(A[1..(n-1)]) \]

Observation

For second case we want to find a subsequence in $A[1..(n-1)]$ that is restricted to numbers less than $A[n]$. This suggests that a more general problem is $\text{LIS\_smaller}(A[1..n], x)$ which gives the longest increasing subsequence in $A$ where each number in the sequence is less than $x$. 
Recursive Approach

**LIS\_smaller**(\(A[1..n], x\)) : length of longest increasing subsequence in \(A[1..n]\) with all numbers in subsequence less than \(x\)

\[
\text{LIS\_smaller}(A[1..n], x) :
\begin{align*}
&\text{if } (n = 0) \text{ then return } 0 \\
&m = \text{LIS\_smaller}(A[1..(n - 1)], x) \\
&\text{if } (A[n] < x) \text{ then} \\
&m = \max(m, 1 + \text{LIS\_smaller}(A[1..(n - 1)], A[n])) \\
\text{Output } m
\end{align*}
\]

**LIS**(\(A[1..n]\)) :
\[
\text{return LIS\_smaller}(A[1..n], \infty)
\]
Example

Sequence: $A[1..7] = 6, 3, 5, 2, 7, 8, 1$
THE END

... (for now)