

1 Longest Increasing Subsequence

The *longest increasing subsequence problem* is defined as follows:

We are given a sequence of numbers $(a_i | i = 0, \dots, n - 1)$, $a_i \in \mathbb{R}$ (just for concreteness).

The goal is to find an increasing subsequence of maximum length. That is, find a sequence of indices $(i_j | j = 0, \dots, k - 1)$ such that

$$j < j' \text{ implies } i_j < i_{j'} \text{ and } a_{i_j} < a_{i_{j'}} \quad (*)$$

and

k is maximal with with property $(*)$.

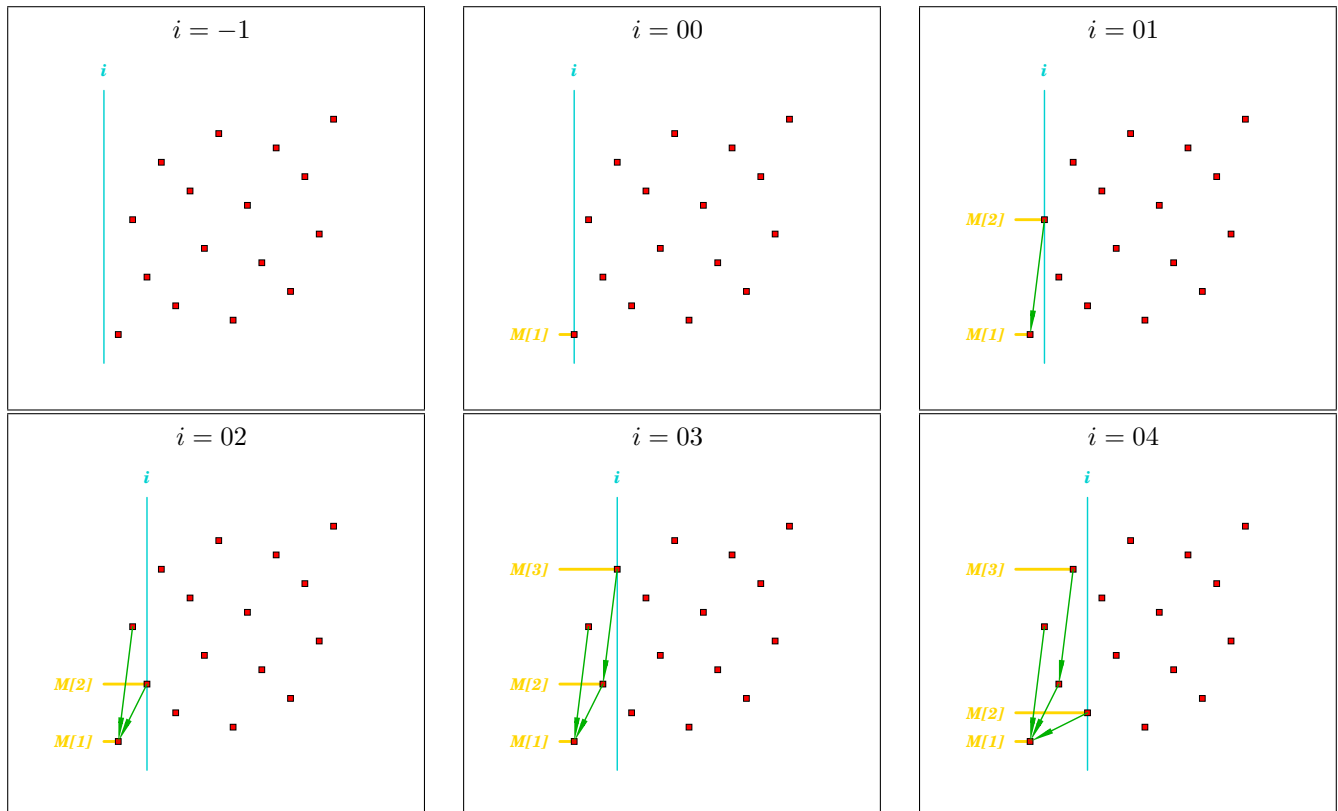
The *heaviest increasing subsequence problem* is defined similar, but we are additionally given a sequence of weights, $(w_i | i = 0, \dots, n - 1)$ and the goal is to find an increasing subsequence such that

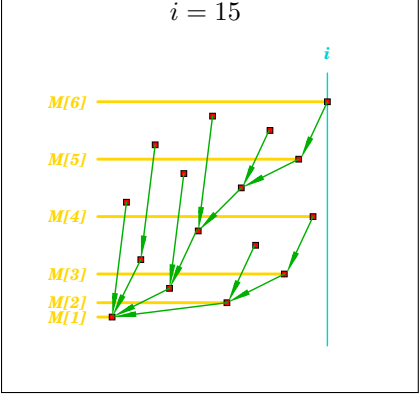
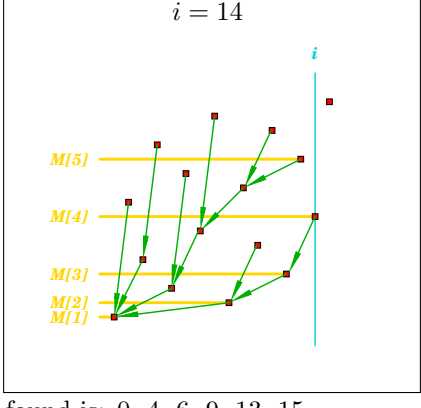
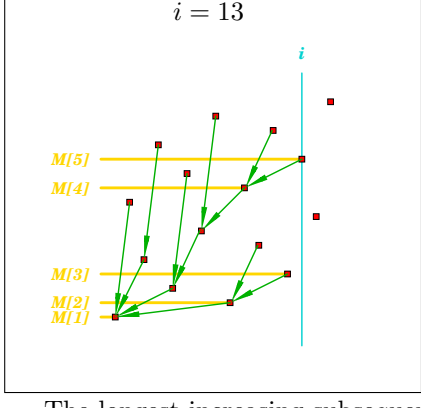
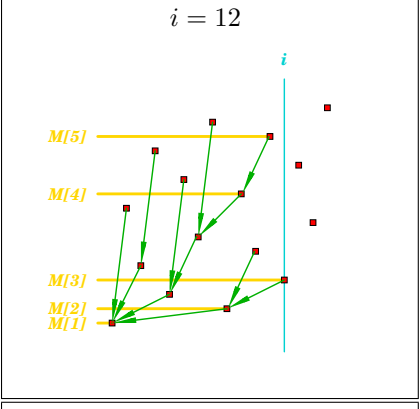
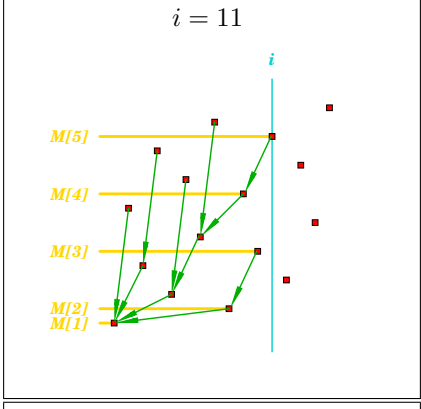
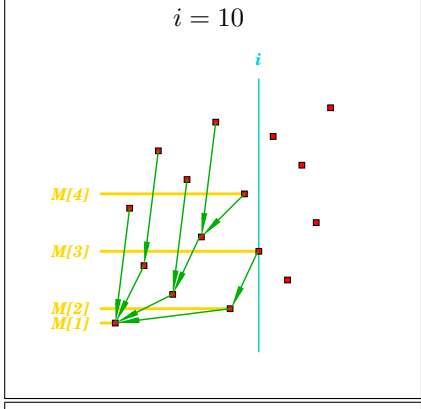
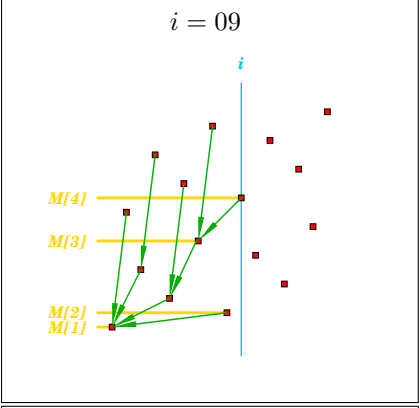
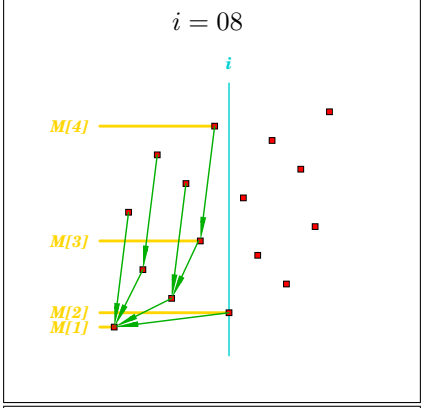
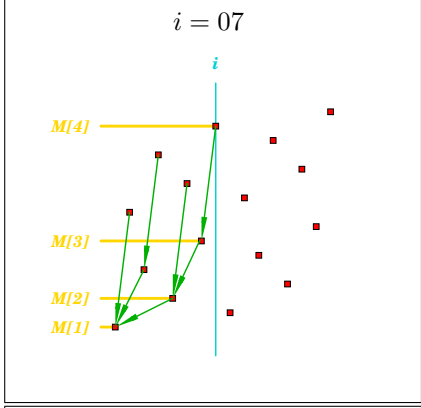
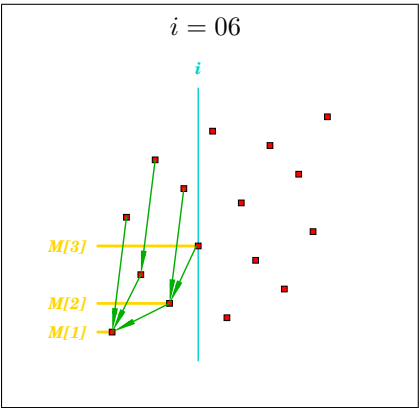
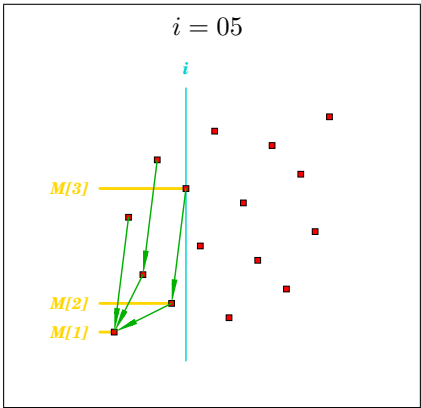
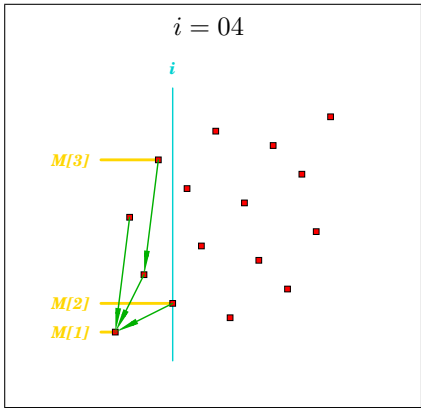
$$\sum_{j=0}^{k-1} w_{i_j} \text{ is maximal with property } (*).$$

An $O(n \log n)$ algorithm for the longest increasing subsequence problem is explained at: http://en.wikipedia.org/wiki/Longest_increasing_subsequence_problem

2 An example run of Fredman's algorithm (1971)

We use the van der Corput sequence of length 16 as an example instance: 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15. It is represented by red boxes. The blue vertical line indicates the loop variable i . The entries of the table M are indicated by horizontal orange lines. The array P which is used for backtracking is indicated by green arrows.





The longest increasing subsequence found is: 0, 4, 6, 9, 13, 15.

3 Debug output used for drawing those figures

The debug output of my own implementation of Fredman's algorithm is quoted below.

```
input_sequence: 0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
i: 0
input_sequence[i]: 0
lower_bound: 0
P: -1
M: -1 0
reverse_result: 0
subseq_indices: 1

i: 1
input_sequence[i]: 8
while(): lower_bound: 1 upper_bound: 1
lower_bound: 1
P: -1 0
M: -1 0 1
reverse_result: 1 0
subseq_indices: 0 1

i: 2
input_sequence[i]: 4
while(): lower_bound: 1 upper_bound: 2
lower_bound: 1
P: -1 0 0
M: -1 0 2
reverse_result: 2 0
subseq_indices: 0 2

i: 3
input_sequence[i]: 12
while(): lower_bound: 1 upper_bound: 2
lower_bound: 2
P: -1 0 0 2
M: -1 0 2 3
reverse_result: 3 2 0
subseq_indices: 0 2 3

i: 4
input_sequence[i]: 2
while(): lower_bound: 1 upper_bound: 3
while(): lower_bound: 1 upper_bound: 2
lower_bound: 1
P: -1 0 0 2 0
M: -1 0 4 3
reverse_result: 3 2 0
subseq_indices: 0 2 3

i: 5
input_sequence[i]: 10
while(): lower_bound: 1 upper_bound: 3
while(): lower_bound: 2 upper_bound: 3
lower_bound: 2
P: -1 0 0 2 0 4
M: -1 0 4 5
reverse_result: 5 4 0
subseq_indices: 0 4 5

i: 6
input_sequence[i]: 6
while(): lower_bound: 1 upper_bound: 3
while(): lower_bound: 2 upper_bound: 3
lower_bound: 2
P: -1 0 0 2 0 4 4
M: -1 0 4 6
reverse_result: 6 4 0
subseq_indices: 0 4 6

i: 7
input_sequence[i]: 14
while(): lower_bound: 1 upper_bound: 3
while(): lower_bound: 2 upper_bound: 3
lower_bound: 3
P: -1 0 0 2 0 4 4 6
M: -1 0 4 6 7
reverse_result: 7 6 4 0
subseq_indices: 0 4 6 7

i: 8
input_sequence[i]: 1
while(): lower_bound: 1 upper_bound: 4
```

```
while(): lower_bound: 1 upper_bound: 2
lower_bound: 1
P: -1 0 0 2 0 4 4 6 0
M: -1 0 8 6 7
reverse_result: 7 6 4 0
subseq_indices: 0 4 6 7

i: 9
input_sequence[i]: 9
while(): lower_bound: 1 upper_bound: 4
while(): lower_bound: 2 upper_bound: 4
while(): lower_bound: 3 upper_bound: 4
lower_bound: 3
P: -1 0 0 2 0 4 4 6 0 6
M: -1 0 8 6 9
reverse_result: 9 6 4 0
subseq_indices: 0 4 6 9

i: 10
input_sequence[i]: 5
while(): lower_bound: 1 upper_bound: 4
while(): lower_bound: 2 upper_bound: 4
while(): lower_bound: 2 upper_bound: 3
lower_bound: 2
P: -1 0 0 2 0 4 4 6 0 6 8
M: -1 0 8 10 9
reverse_result: 9 6 4 0
subseq_indices: 0 4 6 9

i: 11
input_sequence[i]: 13
while(): lower_bound: 1 upper_bound: 4
while(): lower_bound: 2 upper_bound: 4
while(): lower_bound: 3 upper_bound: 4
lower_bound: 4
P: -1 0 0 2 0 4 4 6 0 6 8 9
M: -1 0 8 10 9 11
reverse_result: 11 9 6 4 0
subseq_indices: 0 4 6 9 11

i: 12
input_sequence[i]: 3
while(): lower_bound: 1 upper_bound: 5
while(): lower_bound: 1 upper_bound: 3
while(): lower_bound: 2 upper_bound: 3
lower_bound: 2
P: -1 0 0 2 0 4 4 6 0 6 8 9 8
M: -1 0 8 12 9 11
reverse_result: 11 9 6 4 0
subseq_indices: 0 4 6 9 11

i: 13
input_sequence[i]: 11
while(): lower_bound: 1 upper_bound: 5
while(): lower_bound: 3 upper_bound: 5
while(): lower_bound: 4 upper_bound: 5
lower_bound: 4
P: -1 0 0 2 0 4 4 6 0 6 8 9 8 9
M: -1 0 8 12 9 13
reverse_result: 13 9 6 4 0
subseq_indices: 0 4 6 9 13

i: 14
input_sequence[i]: 7
while(): lower_bound: 1 upper_bound: 5
while(): lower_bound: 3 upper_bound: 5
while(): lower_bound: 3 upper_bound: 4
lower_bound: 3
P: -1 0 0 2 0 4 4 6 0 6 8 9 8 9 12
M: -1 0 8 12 14 13
reverse_result: 13 9 6 4 0
subseq_indices: 0 4 6 9 13

i: 15
input_sequence[i]: 15
while(): lower_bound: 1 upper_bound: 5
while(): lower_bound: 3 upper_bound: 5
while(): lower_bound: 4 upper_bound: 5
lower_bound: 5
P: -1 0 0 2 0 4 4 6 0 6 8 9 8 9 12 13
M: -1 0 8 12 14 13 15
reverse_result: 15 13 9 6 4 0
subseq_indices: 0 4 6 9 13 15
```