1 Longest increasing subsequence

Example

k	1	2	3	4	5	6	7	8	9
L[k]	90	50	20	80	70	30	10	60	40
S[k]									

1.1 Recurrence

1.2 Pseudocode

```
LIS(L)
Initialize S[1...n]
for k = 1 to n
    S[k] = 1

    for j = 1 to k-1
        if L[k] > L[j] & S[k] < S[j] + 1
            S[k] = S[j] + 1

return
```

Running time:

2 Knapsack (KT §6.4)

Definition In the **knapsack problem** we are given

- a set of n items
- each item i has specified size s_i
- item i has value v_i

Goal: Find subset of items of maximum total value such that the sum of their sizes is at most S.

Example $S = 10$

i	1	2	3	4
v_i	10	40	30	50
s_i	5	4	6	3

2.1 Recurrence

2.2 Pseudocode

```
Knapsack(s, n, S)
    Initialize K[0, i] =    K[w, 0] =
    for i = 1 to n
        for w = 1 to S
            if s[j] > w

            else

return
```

Running time:

Remark.

3 Interval Scheduling/Activity Selection Problem (KT §6.1, CLRS §16.1)

Input: List of intervals $S =$

Goal: Find a subset

First attempt: Dynamic Programming

1. Subproblems: for any $i < j$, the optimal solution for intervals

2. Guess an interval

3. Recurrence:

Second attempt: Improved dynamic programming
Sort the activities by:

Guess whether

Subproblems:

Recurrence:

3.1 Intro to greedy

Maybe we don't need to try all possible activities? Can we identify an activity that is used in an optimal solution?

Ideas:

- Activity with the
- Shortest
- Activity intersecting

Turns out