Dynamic programming

Motto: Those who cannot remember the past are condemned to repeat it.

Václav Hlaváč

Czech Technical University in Prague

Czech Institute of Informatics, Cybernetics and Robotics (CIIRC)

160 00 Prague 6, Jugoslávských partyzánů 1580/3, Czech Republic

vaclav.hlavac@cvut.cz, http://people.ciirc.cvut.cz/hlavac/

What is DP ?

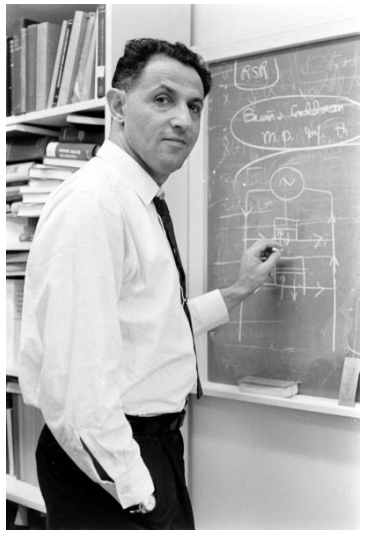
- Dynamic programming (DP) is a general algorithm design technique for solving optimization tasks defined by or formulated as recurrences with overlapping subinstances *(cf. to divide-and-conquer, where instances are non-overlapping).*
- "Programming" here means "resource management" or "planning".
- Related concepts are linear programming (minimizing a linear function subject to some linear constraints), mathematical programming (as before, linearity constraint released).
- Applies to problems where the cost function can be:
  - decomposed into a sequence (ordering) of stages, and
  - each stage depends on only a fixed number of previous stages.
- The cost function need not be convex (if variables are continuous).

Dynamic programming, the main idea

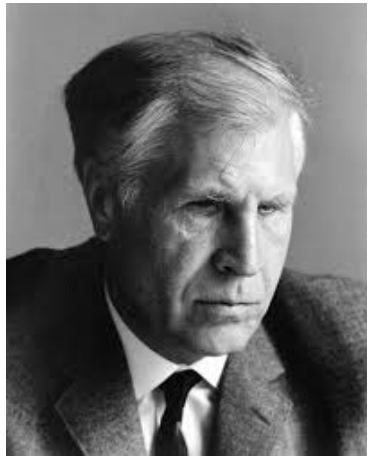
- Set up a recurrence relating a solution to a larger instance to solutions of some smaller instances.
- Solve smaller instances once.
- Record solutions in a table. *(Some people say that DP means "table filling".)*
- Extract solution to the initial instance from that table.
- DP reduces computation by
  - Solving subproblems in a bottom-up fashion.
  - Storing solution to a subproblem the first time it is solved.
  - Looking up the solution when subproblem is encountered again.
- Excellent tutorial Introduction to Dynamic Programming 1

Dynamic programming inventors

- American mathematician Richard Bellman (1920-1984) introduced the principle of optimality and dynamic programming in 1952.
- Principle of optimality: Given an optimal sequence of decisions or choices, each subsequence must also be optimal.
  Stuart Dreyfus: Richard Bellman on the Birth of Dynamic Programming. Operations Research, Vol. 50, No. 1, 48-51, 2002
- Related: Lev Pontryagin's (1908-1988) maximum principle formulated in 1956. He was blind from his age of 14.



Richard Bellman



Lev Pontryagin

Examples of DP algorithms

- Unix diff for comparing two files
- Computing a binomial coefficient
- Longest common subsequence
- Warshall's algorithm for transitive closure
- Floyd's algorithm for all-pairs shortest paths
- Constructing an optimal binary search tree
- Some instances of difficult discrete optimization problems:
  - Traveling salesman
  - Knapsack task: *Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.*

Motivation: Fibonacci numbers 1

- Fibonacci number, the definition,
  The sequence: $F(0) = 0$; $F(1) = 1$; for $n>1$ $F(n)=F(n-1) + F(n-2)$;
  $F(0) = 0; F(1) = 1; F(2) = 1 + 0 = 1; F(3) = F(2) + F(1) = 2;$

n	0	1	2	3	4	5	6	7	8	9	10	11	12	…	*n-2*	*n-1*	*n*
F(n)	0	1	1	2	3	5	8	13	21	34	55	89	144	…	*F(n-2)*	*F(n-1)*	*F(n)*

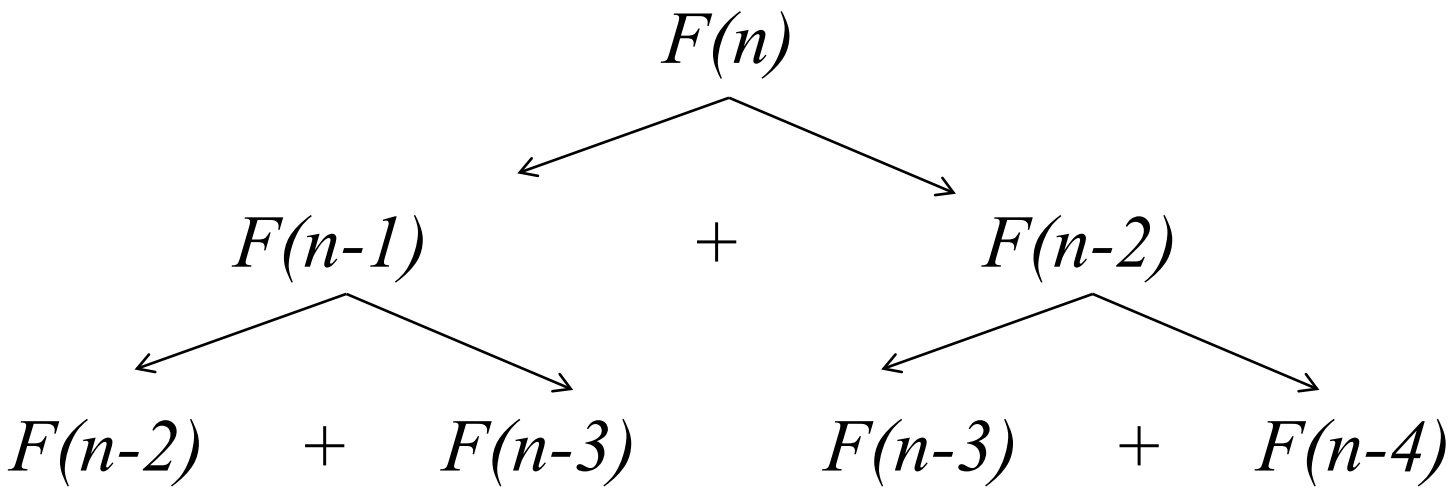
- By Leonardo Fibonacci Pisano, literally “Leonardo, son of Bonacci, of Pisa”, (published 1202)
-

```
int fib (int n) {
    if (n < 2)
        return 1;
    return fib(n-1) + fib(n-2);
}
```

```
void fib () {
    fibresult[0] = 1;
    fibresult[1] = 1;
    for (int i = 2; i<n; i++)
        fibresult[i] = fibresult[i-1] + fibresult[i-2];
}
```

Fibonacci numbers 2, example

Computing the n-th Fibonacci number recursively (top-down, also divide-and-conquer strategy):



Intuition behind DP

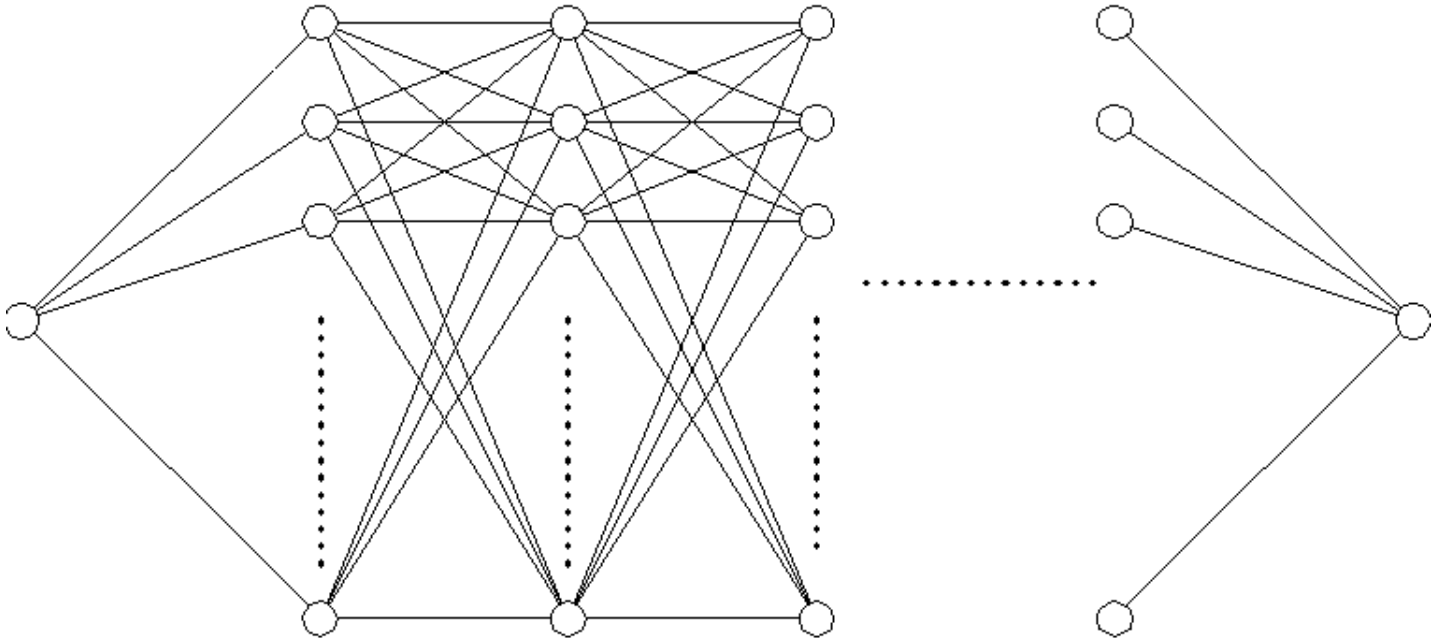
- **The idea:** Compute the solutions to the subsub-problems *once* and store the solutions in a table, so that they can be reused (repeatedly) later.
- **Remark:** We trade space for time.

- **Dynamic programming** (DP) solves every subsubproblem exactly once, and is therefore more efficient in those cases where the subsubproblems are not independent.

Trellis graph

- A **trellis** is a graph whose nodes are ordered into vertical slices (*time*) with each node at each time connected to at least one node at an earlier and at least one node at a later time.
- The earliest and latest times in the trellis have only one node.
- Trellis graphs are used in encoders and decoders for communication theory and encryption. They are also the central datatype used in
  - **Viterbi Algorithm** for Hidden Markov Models - a dynamic programming algorithm for finding the most likely sequence of hidden states—called the Viterbi path
  - **Baum–Welch algorithm** - is a special case of the EM algorithm used to find the unknown parameters of a hidden Markov model

Trellis graph example

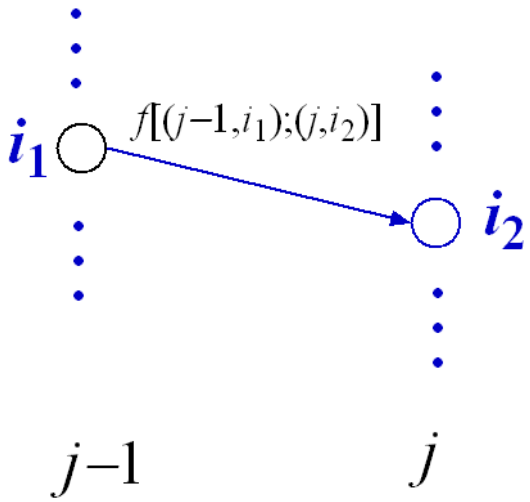


1 2 3 4 K-1 K

$$f\big((k-1,i),(k,j)\big)$$

Trellis graph 2

- Distance (weight) from point i_1 at stage $(j-1)$ to point i_2 at stage j:



- The total value of cost function:

$$L = \sum_{j=1}^{K} f\left((j-1, i_{j-1}), (j, i_j)\right) \rightarrow \min$$

Preparation for DP

- Cost function:

$$G_k(n) = \min\left\{\sum_{j=1}^{k} f\left(p_{j-1}(i_{j-1}), p_j(i_j)\right)\right\}$$

- Recursive equation:

$$G_k(n) = \min_i\left\{G_k(i) + f\left((k-1,i),(k,n)\right)\right\}$$

- Initialization: $G_0(n) = 0$

Complexity issues

- Exhaustive search: $O(n^K)$
- Dynamic programming algorithm: $O(Kn^2)$, where
  - K is the number of stages,
  - n is the number of points in a stage