# Greedy algorithms

- Greed is a design strategy for solving combinatorial optimization problems :

  Construct a solution by making a sequence of decisions.

  Each decision augments a partial solution.

  No decision is ever retracted (i.e. no backtracking).

  The decision we make is to perform that augmentation that is *locally* best under the objective function (i.e. we are greedy).

- There are two aspects to greed :

  (1) Avarice : we seek the largest improvement in the objective function ;

  (2) Short-sightedness : we seek short-term rather than long-term improvement.

## Greedy algorithms, cont'd

- Very few problems can be solved by greedy algorithms, but when they can, such an algorithm is usually very efficient.

  (Solution by a greedy algorithm is an indication that the problem is computationally "easy". Most problems are computationally "hard". However, computationally hard problems can often be approximated by greedy procedures.)

- Greedy algorithms solve a problem bottom-up:

  "Put this element into the solution,
  then this element,
  then this ...."

  Dynamic programming algorithms solve a problem top-down:

  "What elements are in a solution?
  I don't know, but
  solve this subproblem,
  and this subproblem,
  and this one, ...
  and then I'll know."