Dynamic Programming

- *repeated subproblems* refers to getting to the same subproblem with different sets of choices
  - e.g. (0-1 knapsack) there may be different ways to pick from the first *k* items that result in the same remaining capacity in the pack
  - not referring to merely different *orders* of making the same set of choices – redundant paths should be not be considered in the first place
- repeated subproblems can arise when what matters for solving a subproblem is the state resulting from the partial solution rather than the partial solution itself
  - e.g. knapsack subproblem is based on the remaining items and the remaining capacity, not exactly which items have been picked so far

Dynamic Programming

The idea of dynamic programming –

- formulate the problem as a backtracking problem
  - series of choices approach
    - solution is constructed by making a series of decisions
  - case analysis recursive structure
    - you consider the next possibilities for the current decision, then ask friends to solve the problem given the consequences of each choice
  - subproblem solution is just the subproblem solution, not a complete solution
- identify how to parameterize the subproblems so that subproblem solutions can be stored instead of recomputed – *memoization*
- compute subproblem solutions by iterating through the subproblem states rather than doing a depth-first search of the solution space