Multistage Decision Processes

- A *multistage decision process* is a process that can be separated into a number of sequential steps or stages(i.e. network problems). whcih may be completed in one or more ways.
- The options for completing the stages are *decisions*. A *policy* is a sequence of decisions, one for each stage of the process.
- The condition of the the process at a given stage is called the *state* at that stage; each decision effects a transition from the current state to a state associated with the next stage.
- A multistage decision process is *finite* if there are only a finite number of stages in the process, and a finite number of states associated with each stage.
- Many multistage decision processes have returns (costs or benefits) associated with each decision, and these returns may vary with both the stage and state of the process.
- The objective in analyzing such processes is to determine an optimal policy, one that results in the best overall return.
- A multistage decision process is said to be *deterministic* if the outcome of each decision (in particular, the state produced by each decision) is known exactly.

Dynamic Programming

Dynamic Programming is a powerful technique that allows one to solve many different types of problems in time $O(n^2)$ or $O(n^3)$ for which a naive approach would take exponential time.

- Dynamic programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions ideally, using a memory-based data structure.
- **[IMPORTANT]** The next time the same subproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time at the expense of a (hopefully) modest expenditure in storage space. (Each of the subproblem solutions is indexed in some way, typically based on the values of its input parameters, so as to facilitate its lookup.) The technique of storing solutions to subproblems instead of recomputing them is called "memoization".
- Dynamic programming algorithms are often used for optimization of multi-stage decision processes. Dynamic Programming is based on the Bellman Principle of Optimality.
- A dynamic programming algorithm will examine the previously solved subproblems and will combine their solutions to give the best solution for the given problem.
- (In comparison, a greedy algorithm treats the solution as some sequence of steps and picks the locally optimal choice at each step.
  (Using a greedy algorithm does not guarantee an optimal solution, because picking locally optimal choices may result in a bad global solution, but it is often faster to calculate.)

- Remark: Some greedy algorithms (such as Kruskal's or Prim's for minimum spanning trees are proven to lead to the optimal solution.)

Attributed of Dynamic Programming Problems

- There are two key attributes that a problem must have in order for dynamic programming to be applicable: optimal substructure and overlapping sub-problems.
- **[IMPORTANT]** If a problem can be solved by combining optimal solutions to non-overlapping sub-problems, the strategy is called "divide and conquer" instead.

Optimal substructure Optimal substructure means that the solution to a given optimization problem can be obtained by the combination of optimal solutions to its sub-problems.

Overlapping Subproblems The space of solutions is shared by several different subproblems.