Dynamic Programs

Antonio Bucchiarone

29-04-2021

How to Dynamic Program

Five easy steps!

1. Define **subproblems**
2. **Guess** something (part of solution)
3. Relate subproblem solutions (**recurrence**)
4. Recurse and **memoize** (top down) or Build DP table bottom up
5. **Solve** original problem via subproblems (usually easy)

How to Analyze Dynamic Programs

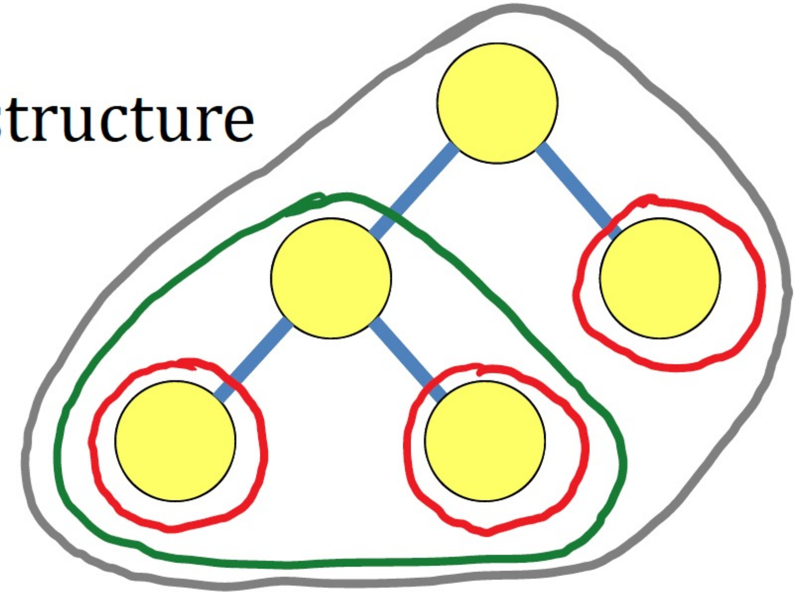
1. Define subproblems *count # subproblems*
2. Guess something *count # choices*
3. Relate subproblem solutions *analyze time per subproblem*
4. DP running time = *# subproblems x time per subproblem*
5. Sometimes *additional running time* to solve original problem

Two Kinds of Guessing

1. **Within a subproblem**
   - Crazy Eights: previous card in trick
   - Sequence alignment: align/drop one character
   - Bellman-Ford: previous edge in path
   - Floyd-Warshall: use vertex ?
   - Parenthesizing: last multiplication
   - Knapsack: include item ?
   - Tetris training: how to place piece
2. **Using additional subproblems**
   - Knapsack: how much space left in knapsack
   - Tetris training: current board configuration

Structural Dynamic Programming

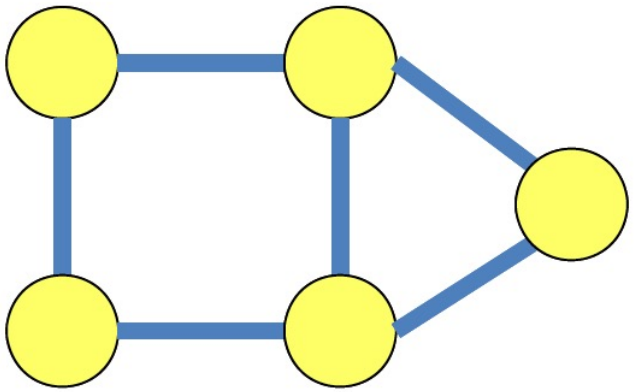
- Follow a combinatorial structure other than a sequence / a few sequences
  - Like structural vs. regular induction
- <u>Main example:</u> Tree structure
- <u>Useful subproblems:</u>
  for every vertex v,
  subtree rooted at v



Vertex Cover

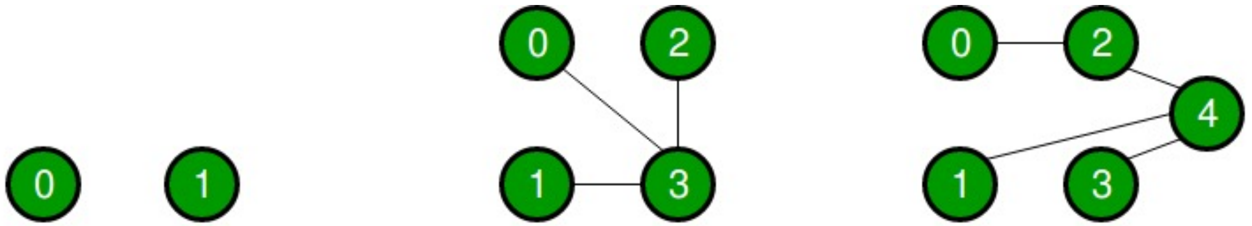
- Given an undirected graph $G = (V, E)$
- Find a minimum-cardinality set S of vertices containing at least one endpoint of every edge
  - Equivalently, find a minimum set of guards for a building of corridors, or (unaligned) streets in city

<u>Example:</u>



Vertex-Cover

- A vertex cover of an **undirected graph** is a *subset of its vertices* such that for every edge (u, v) of the graph, **either ‘u’ or ‘v’ is in the vertex** cover.
- ***Given an undirected graph, the vertex cover problem is to find minimum size vertex cover***.

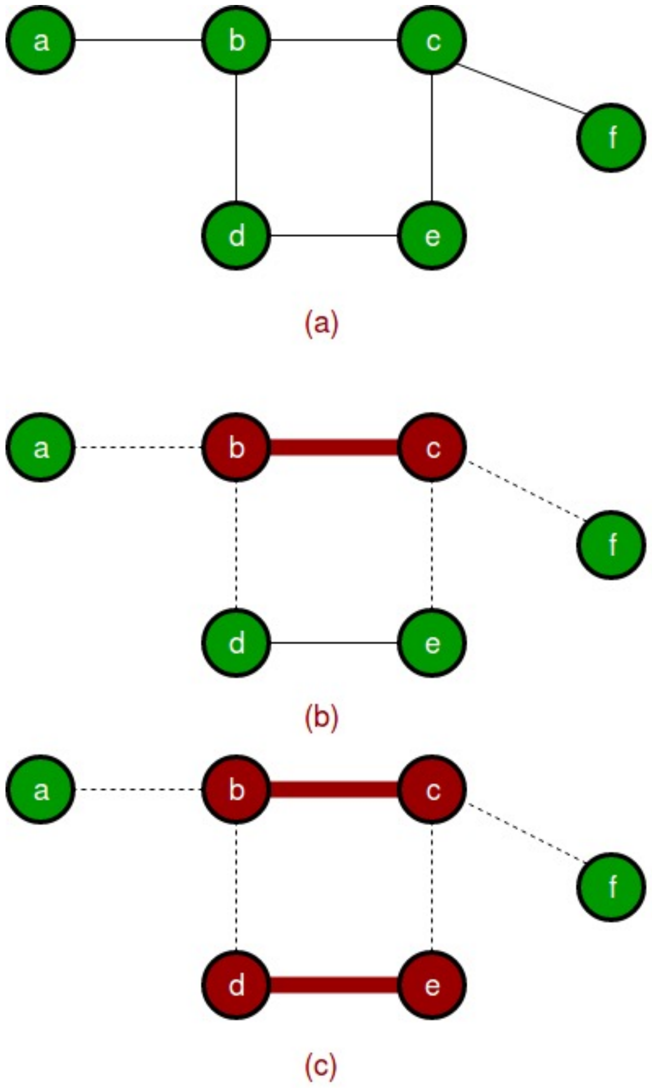


Minimum vertex cover is empty{} | Minimum vertex cover is {3} | Minimum vertex cover is {4, 2} or {4, 0}

Approximate Algorithm for Vertex-Cover

1) Initialize the result as {}
2) Consider a set of all edges in given graph. Let the set be E.
3) Do following while E is not empty ..
   a) Pick an arbitrary edge (u, v) from set E and add 'u' and 'v' to result ...
   b) Remove all edges from E which are either incident on u or v.
4) Return result

Running Example

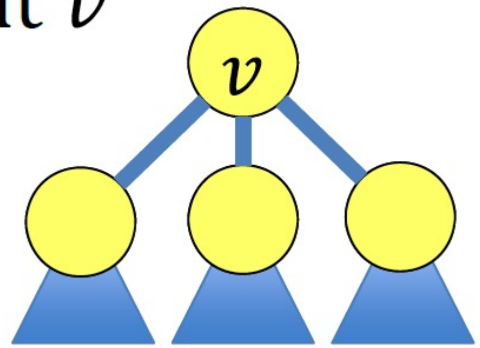


Minimum Vertex Cover is {b, c, d} or {b, c, e}

Vertex Cover in Tree DP

0. Root the tree arbitrarily.

1. **Subproblems:** for $v \in V$: size of smallest vertex cover in subtree rooted at v



2. **Guess:** is v in the cover?
   - <u>YES:</u>
     - Cover children edges
     - Left with children subtrees
   - <u>NO:</u>
     - All children must be in cover
     - Left with grandchildren subtrees