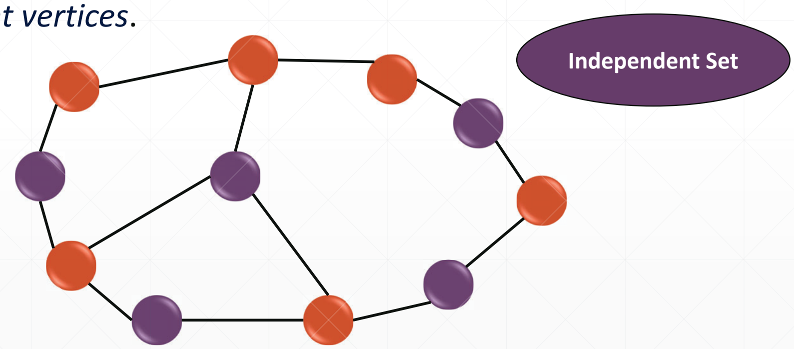
Dynamic Programming Algorithms

Maximum Independent Set

CS 336: Design and Analysis of Algorithms
Konstantin Makarychev

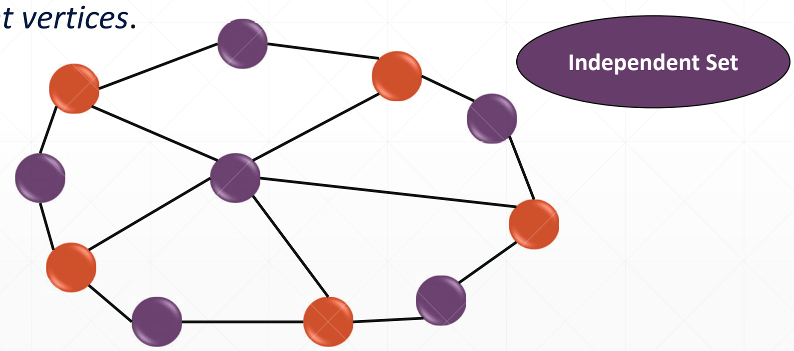
Independent Sets in Graphs

- I is an *independent set* in a graph G if I doesn't contain *adjacent vertices*.



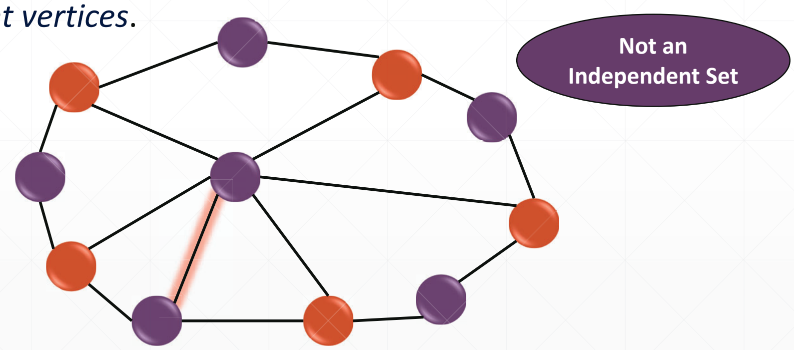
Independent Sets in Graphs

- I is an *independent set* in a graph G if I doesn't contain *adjacent vertices*.



Independent Sets in Graphs

- I is an *independent set* in a graph G if I doesn't contain *adjacent vertices*.



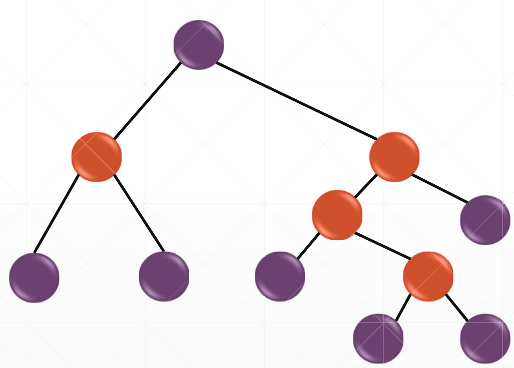
Maximum Independent Sets

Problem: Given a graph $G = (V, E)$, find a maximum independent set $I \subset V$.

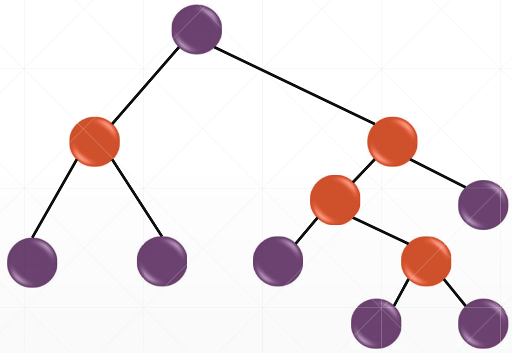
➢ **Bad news:** The problem is very hard to solve (NP-hard).

➢ **Good news:** The problem is easy on trees.

Maximum Independent Set in Trees



Maximum Independent Set in Trees

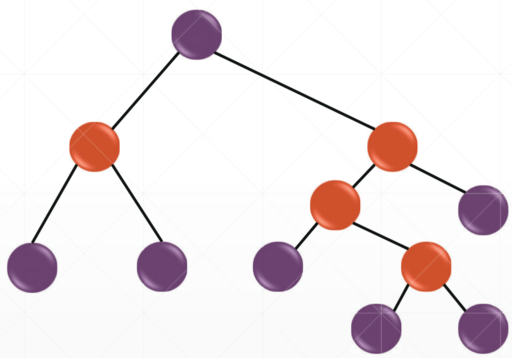


Greedy Algorithm for Trees

- Sort vertices by depth in decreasing order.
- For each node x:

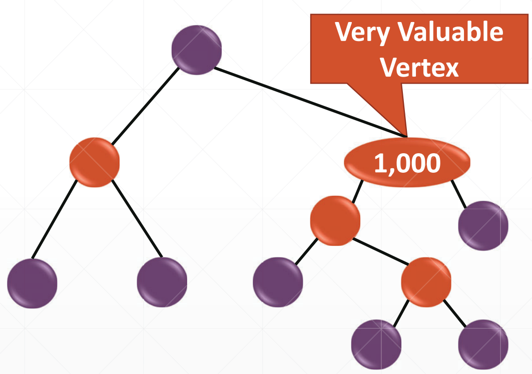
  if children of x are not in $\boldsymbol{I}$, add x.
  Else: discard x.

Maximum Independent Set in Weighted Trees



- Suppose vertices have weights.
- **Goal:** Find maximum weight independent set.

Maximum Independent Set in Weighted Trees



- Suppose vertices have weights.
- Goal: Find maximum weight independent set.
- Very expensive vertices must belong to $\boldsymbol{I}$.

Recursive Algorithm for Binary Trees

function Max_Independent_Set (x)

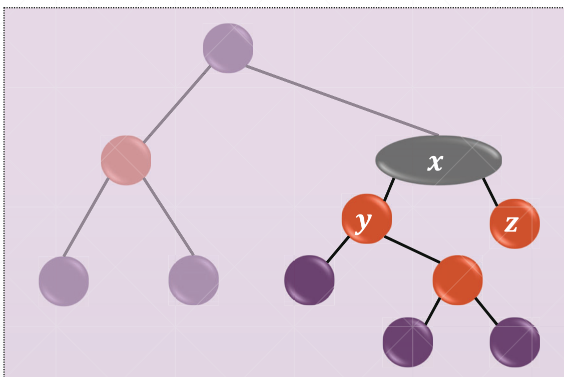
Let y and z be the children of x.

Find optimal solutions for subtrees rooted in y and z.

$$I_y = \text{Max\_Independent\_Set}(y)$$
$$I_z = \text{Max\_Independent\_Set}(z)$$

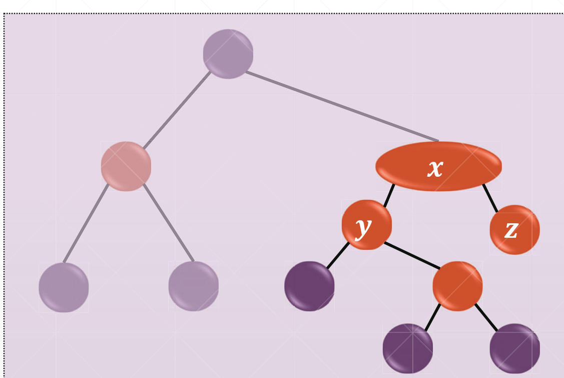
How to combine solutions I_y and I_z? A better subproblem?

Maximum Independent Set in Weighted Trees



- Should we add x to I?
- It depends…

Maximum Independent Set in Weighted Trees

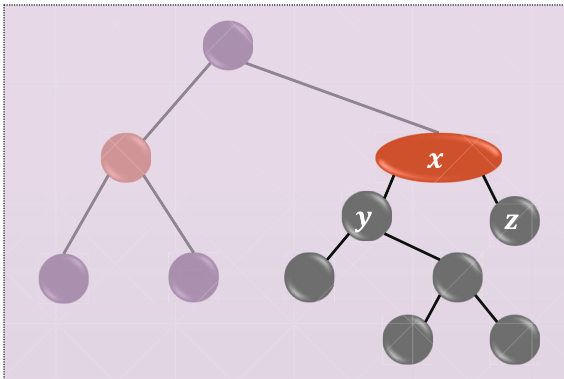


Consider two cases.

- If $x \notin I$, then

  we can pick arbitrary independent sets I_y and I_z in subtrees T_y and T_z rooted at y and z.

Maximum Independent Set in Weighted Trees

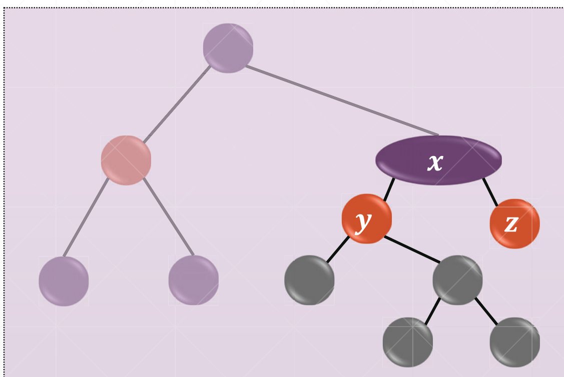


Consider two cases.

- If $x \notin I$, then

  independent sets I_y and I_z in subtrees T_y and T_z rooted at y and z are not restricted.

Maximum Independent Set in Weighted Trees



Consider two cases.

- If $x \in I$, then

  independent sets I_y^R and I_z^R in subtrees T_y and T_z rooted at y and z **cannot** contain y and z.

Two Subproblems

a. Maximum Independent Set in Subtree T_x:
   - The cost of the maximum weight independent set I_x in T_x.

b. Restricted Maximum Independent Set in Subtree T_x:
   - The maximum weight independent set I_x^R in T_x that does not contain x.

Maximum Independent Set in T_x

function Max_Independent_Set (x)

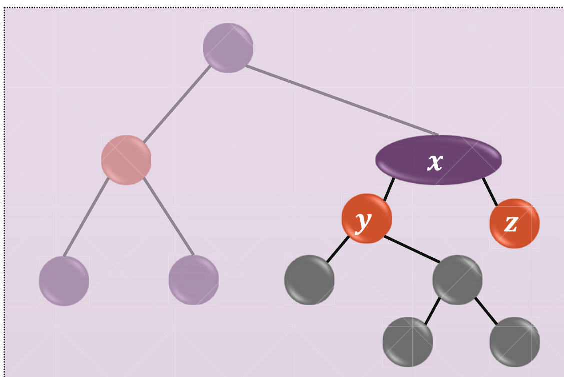
Let y and z be children of x:

I_y = Max_Independent_Set (y)
I_z = Max_Independent_Set (z)

I_y^R = Restricted_Max_Independent_Set (y)
I_y^R = Restricted_Max_Independent_Set (z)

Return the best of two solutions $I_y \cup I_z$ and $\{x\} \cup I_y^R \cup I_z^R$

Restricted Maximum Independent Set



We have $x \notin I_x^R$, hence

independent sets I_y and I_z in subtrees T_y and T_z rooted at y and z are not restricted.

Restricted Maximum Independent Set in T_x

function Restricted_Max_Independent_Set (x)

Let y and z be children of x:

I_y = Max_Independent_Set (y)
I_z = Max_Independent_Set (z)

Return set $I_y \cup I_z$

Dynamic Programming

function Max_Independent_Set (x)

➢ Lookup result for x in the cache or DP table.

Let y and z be children of x:

$$I_y = \text{Max\_Independent\_Set } (y)$$
$$I_z = \text{Max\_Independent\_Set } (z)$$

$$I_y^R = \text{Restricted\_Max\_Independent\_Set } (y)$$
$$I_y^R = \text{Restricted\_Max\_Independent\_Set } (z)$$

Store the result in the cache or DP table.

Return the best of two solutions $I_y \cup I_z$ and $\{x\} \cup I_y^R \cup I_z^R$

Dynamic Programming

function Restricted_Max_Independent_Set (x)

➢ Lookup result for x in the cache or DP table.

Let y and z be children of x:

$$I_y = \text{Max\_Independent\_Set } (y)$$
$$I_z = \text{Max\_Independent\_Set } (z)$$

➢ Store the result $(I_y \cup I_z)$ in the cache or DP table.

Return set $I_y \cup I_z$

Filling values in DP Table

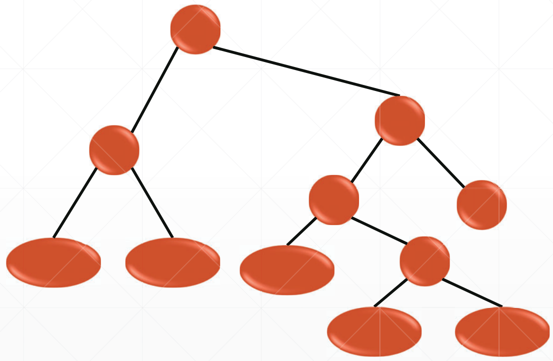
We can

❖ Recursively call Max_Independent_Set and Restricted_Max_Independent_Set as discussed above.

or

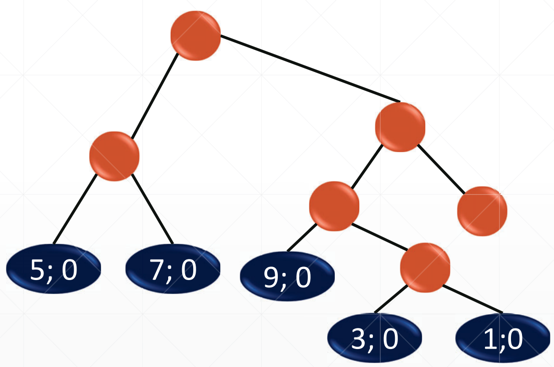
❖ Fill DP table bottom up.

Bottom-up Approach



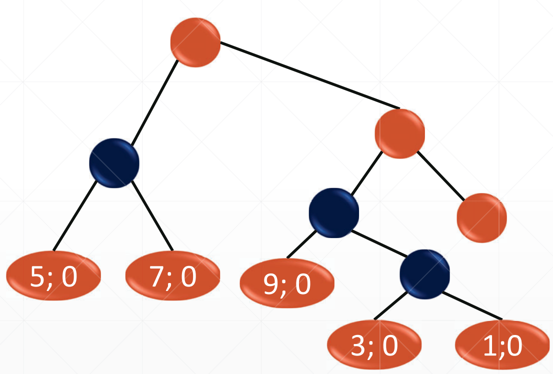
- Start with values in leaves.

Bottom-up Approach



- Start with values in leaves.

Bottom-up Approach



- Start with values in leaves.
- Fill in DP table level by level.
- When **Alg** needs a result, it is already available.

Which Approach is Better?

- In terms of the running time, both solutions are (about) the same.
- The choice depends on the specific problem and implementation.
- A bottom-up approach lets you avoid recursion. Sometimes, recursion may be expensive.
- However, the top-down approach may eliminate some unnecessary computations.

Running Time

- We perform a constant number of operations for every node in the ***binary*** tree.
- Hence, the running time is $O(n)$.

Questions?