# COL 351 : Analysis & Design of Algorithms

## LECTURE 22

## DYNAMIC PROGRAMMING I:

## WEIGHTED INDEPEDENT SET

SEPT 20, 2024 | ROHIT VAISH

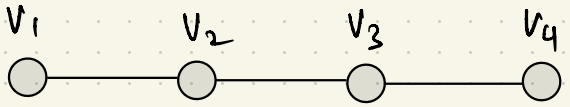
# MAX WEIGHT INDEPENDENT SET ON PATHS

# MAX WEIGHT INDEPENDENT SET ON PATHS

input: a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

# MAX WEIGHT INDEPENDENT SET ON PATHS

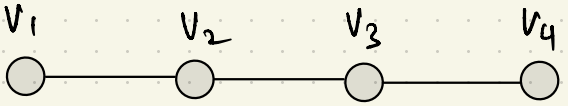
input: a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$



# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

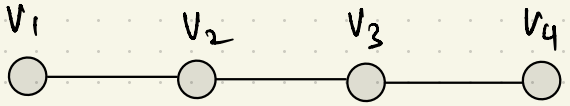


# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices

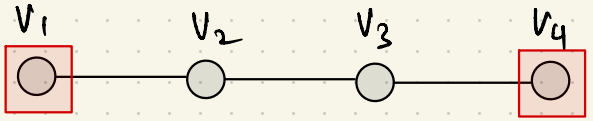


# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices

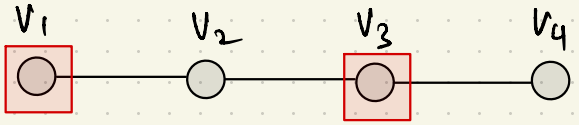


# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices

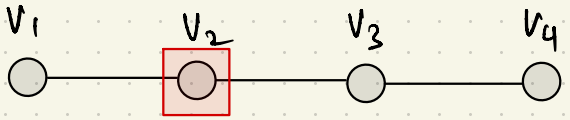


# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices



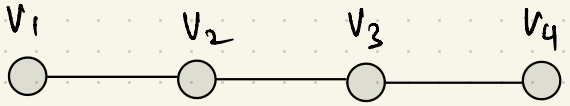
# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices

How many independent sets?



# MAX WEIGHT INDEPENDENT SET ON PATHS

**input:** a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

**output:** an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices


How many independent sets? 8

$\emptyset$, four singletons, $\{v_1, v_3\}$, $\{v_2, v_4\}$, $\{v_1, v_4\}$

# MAX WEIGHT INDEPENDENT SET ON PATHS

input: a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$

output: an independent set $S \subseteq V$ of $G$ with maximum $\sum_{v \in S} w_v$

subset of non-adjacent vertices


How many independent sets? 8

$\emptyset$, four singletons, $\{v_1, v_3\}$, $\{v_2, v_4\}$, $\{v_1, v_4\}$

in general: exponential in $n$ ☹

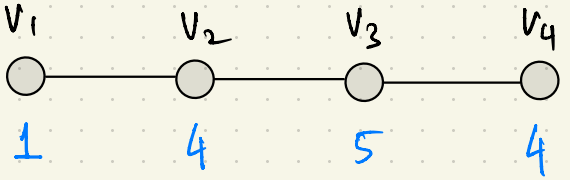
# A GREEDY APPROACH

# A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.

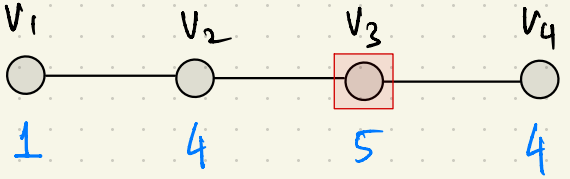
# A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



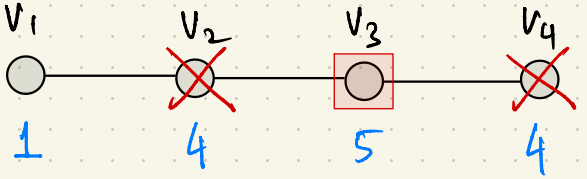
# A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



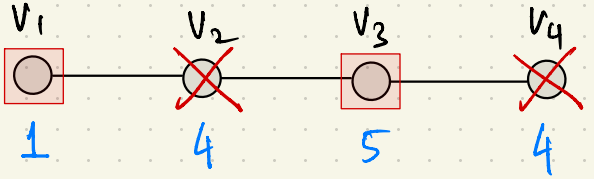
# A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



# A GREEDY APPROACH

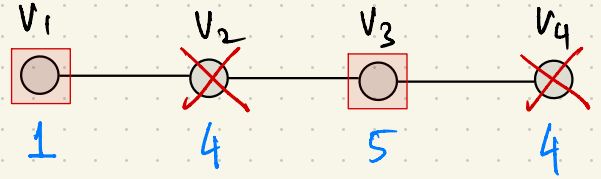
Among feasible vertices, select one with maximum weight.



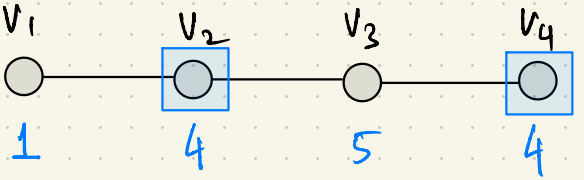

wt = 6

# A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



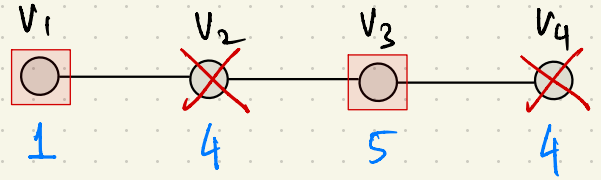

wt = 6



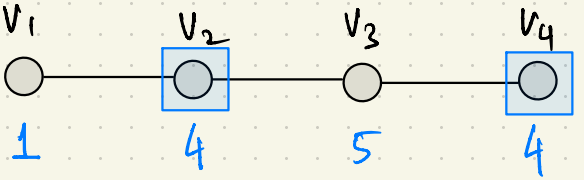

wt = 8

# A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.




wt = 6 ☹️




wt = 8

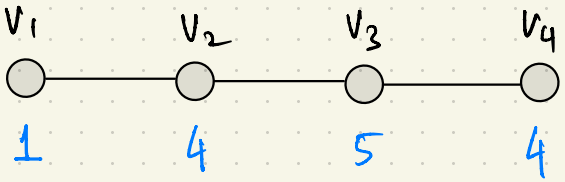
# A DIVIDE & CONQUER APPROACH

# A DIVIDE & CONQUER APPROACH

Recursively compute ind. set for left and right halves and combine the solutions

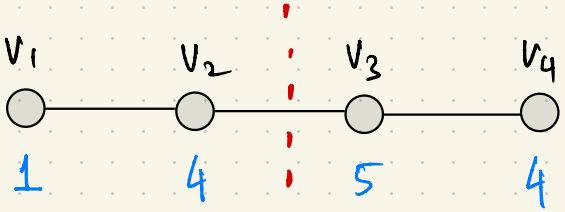
# A DIVIDE & CONQUER APPROACH

Recursively compute ind. set for left and right halves and combine the solutions



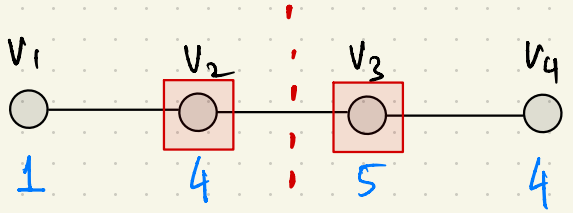
# A DIVIDE & CONQUER APPROACH

Recursively compute ind. set for left and right halves and combine the solutions



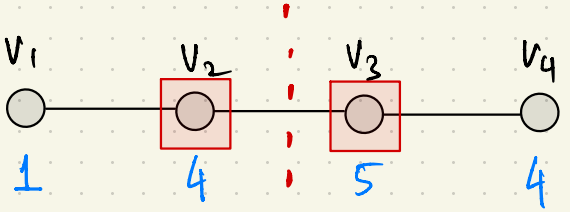
# A DIVIDE & CONQUER APPROACH

Recursively compute ind. set for left and right halves and combine the solutions



# A DIVIDE & CONQUER APPROACH

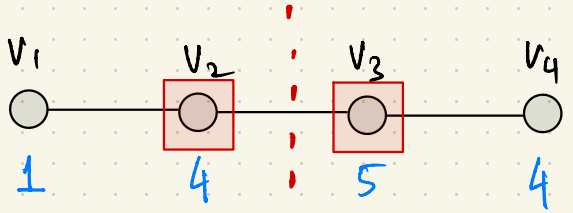
Recursively compute ind. set for left and right halves and combine the solutions




☹ Not clear how to resolve conflicts.

# A DIVIDE & CONQUER APPROACH

Recursively compute ind. set for left and right halves and combine the solutions




☹️ Not clear how to resolve conflicts.

🤔 $O(n^2)$ algorithm with four recursive calls.

# OPTIMAL SUBSTRUCTURE

# OPTIMAL SUBSTRUCTURE

Idea : Optimal solution of overall problem built up from optimal solution of subproblems in a prescribed way.

# OPTIMAL SUBSTRUCTURE

**Idea:** Optimal solution of overall problem built up from optimal solution of subproblems in a prescribed way.

**Hope:** Need to consider only a few subproblems — manageable!

# OPTIMAL SUBSTRUCTURE

Notation : Path graph $G = (V, E)$ $v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

# OPTIMAL SUBSTRUCTURE

Notation : Path graph $G = (V, E)$ $v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

# OPTIMAL SUBSTRUCTURE

Notation : Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

Case analysis : Either $v_n \notin S$ or $v_n \in S$.

# OPTIMAL SUBSTRUCTURE

Notation : Path graph $G = (V, E)$ $\quad v_1 - v_2 - \ldots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

Case analysis : Either $v_n \notin S$ or $v_n \in S$.

Case I : $v_n \notin S$

Case II : $v_n \in S$

# OPTIMAL SUBSTRUCTURE

Notation : Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

Case analysis : Either $v_n \notin S$ or $v_n \in S$.

Case I : $v_n \notin S$

Let $G' := G \setminus \{v_n\}$.

Case II : $v_n \in S$

# OPTIMAL SUBSTRUCTURE

**Notation:** Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

**Case analysis:** Either $v_n \notin S$ or $v_n \in S$.

**Case I:** $v_n \notin S$

Let $G' := G \setminus \{v_n\}$. Then, $S$ is also an independent set of $G'$.

**Case II:** $v_n \in S$

# OPTIMAL SUBSTRUCTURE

Notation : Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

Case analysis : Either $v_n \notin S$ or $v_n \in S$.

Case I : $v_n \notin S$

Let $G' := G \setminus \{v_n\}$. Then, $S$ is also an independent set of $G'$.

$\Rightarrow$ $S$ must be a max wt ind. set of $G'$.

Case II : $v_n \in S$

# OPTIMAL SUBSTRUCTURE

**Notation:** Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

**Case analysis:** Either $v_n \notin S$ or $v_n \in S$.

**Case I:** $v_n \notin S$

Let $G' := G \setminus \{v_n\}$. Then, $S$ is also an independent set of $G'$.

$\Rightarrow$ $S$ must be a max wt ind. set of $G'$.

**Case II:** $v_n \in S$

Then, $v_{n-1} \notin S$.

# OPTIMAL SUBSTRUCTURE

**Notation:** Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

**Case analysis:** Either $v_n \notin S$ or $v_n \in S$.

**Case I:** $v_n \notin S$

Let $G' := G \setminus \{v_n\}$. Then, $S$ is also an independent set of $G'$.

$\Rightarrow$ $S$ must be a max wt ind. set of $G'$.

**Case II:** $v_n \in S$

Then, $v_{n-1} \notin S$. Let $G'' := G \setminus \{v_{n-1}, v_n\}$.

# OPTIMAL SUBSTRUCTURE

**Notation:** Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

**Case analysis:** Either $v_n \notin S$ or $v_n \in S$.

**Case I:** $v_n \notin S$

Let $G' := G \setminus \{v_n\}$. Then, $S$ is also an independent set of $G'$.

$\Rightarrow$ $S$ must be a max wt ind. set of $G'$.

**Case II:** $v_n \in S$

Then, $v_{n-1} \notin S$. Let $G'' := G \setminus \{v_{n-1}, v_n\}$.

Then, $S \setminus \{v_n\}$ is an independent set of $G''$.

# OPTIMAL SUBSTRUCTURE

**Notation:** Path graph $G = (V, E)$ $\quad v_1 - v_2 - \dots - v_{n-2} - v_{n-1} - v_n$

Let $S \subseteq V$ be a max weight independent set (MWIS).

**Case analysis:** Either $v_n \notin S$ or $v_n \in S$.

**Case I:** $v_n \notin S$

Let $G' := G \setminus \{v_n\}$. Then, $S$ is also an independent set of $G'$.

$\Rightarrow$ $S$ must be a max wt ind. set of $G'$.

**Case II:** $v_n \in S$

Then, $v_{n-1} \notin S$. Let $G'' := G \setminus \{v_{n-1}, v_n\}$.

Then, $S \setminus \{v_n\}$ is an independent set of $G''$.

$\Rightarrow$ $S \setminus \{v_n\}$ must be a max wt ind. set of $G''$.

# OPTIMAL SUBSTRUCTURE

if $v_n \notin S$

$\Rightarrow$ $S$ must be a max wt ind. set of $G' := G \setminus \{v_n\}$.

if $v_n \in S$

$\Rightarrow$ $S \setminus \{v_n\}$ must be a max wt ind. set of $G'' := G \setminus \{v_{n-1}, v_n\}$.

# OPTIMAL SUBSTRUCTURE

if $v_n \notin S$

$\Rightarrow$ $S$ must be a max wt ind. set of $G' := G \setminus \{v_n\}$.

if $v_n \in S$

$\Rightarrow$ $S \setminus \{v_n\}$ must be a max wt ind. set of $G'' := G \setminus \{v_{n-1}, v_n\}$.

👎 We don't know which case we are in.

👍 Only need to think about two subproblems!

# OPTIMAL SUBSTRUCTURE

Idea: Try both possibilities and return the better solution.

# OPTIMAL SUBSTRUCTURE

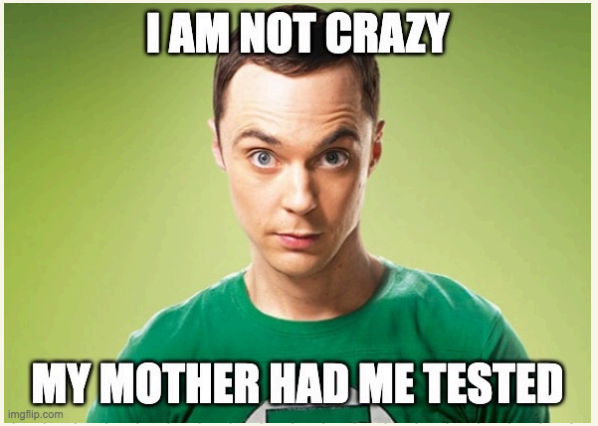
Idea: Try both possibilities and return the better solution.

You: "Are you crazy? Isn't this just brute force?"

# OPTIMAL SUBSTRUCTURE

Idea: Try both possibilities and return the better solution.

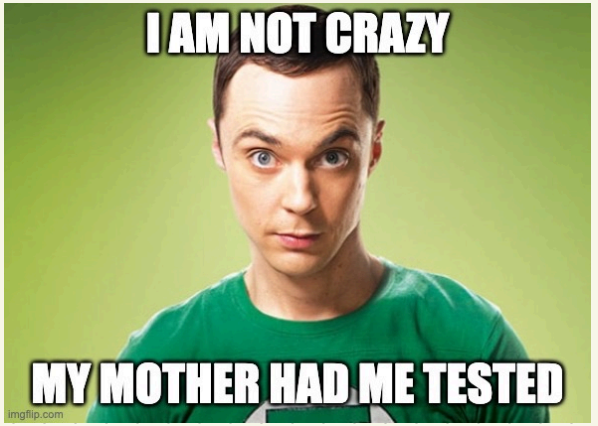
You: "Are you crazy? Isn't this just brute force?"



# OPTIMAL SUBSTRUCTURE

Idea: Try both possibilities and return the better solution.

You: "Are you crazy? Isn't this just brute force?"

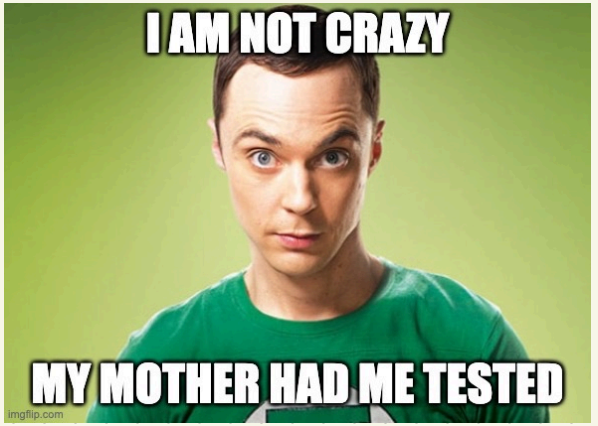



This is recursively organized brute force.

# OPTIMAL SUBSTRUCTURE

Idea: Try both possibilities and return the better solution.

You: "Are you crazy? Isn't this just brute force?"




This is recursively organized brute force.

By eliminating redundant computations, we can solve the problem in linear time.