

COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 20

DYNAMIC PROGRAMMING I:

WEIGHTED INDEPENDENT SET

SEPT 09, 2026

|

ROHIT VAISH

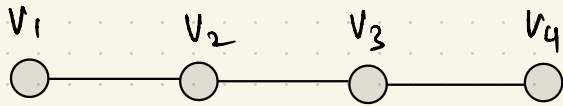
MAX WEIGHT INDEPENDENT SET ON PATHS

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

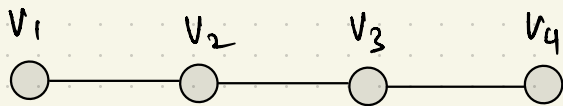
input: a path graph $G = (V, E)$ with nonnegative weights on vertices $\{w_v\}_{v \in V}$



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 $\sum_{v \in S} w_v$

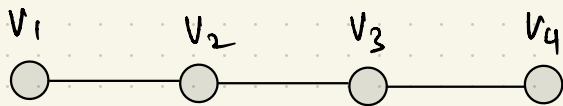


MAX WEIGHT INDEPENDENT SET ON PATHS

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

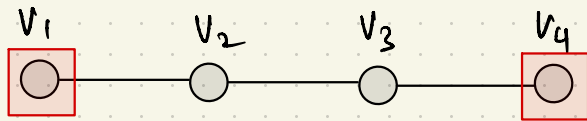


MAX WEIGHT INDEPENDENT SET ON PATHS

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

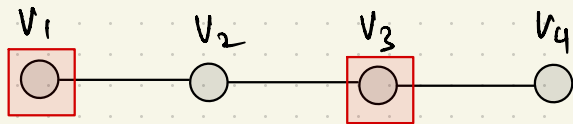


MAX WEIGHT INDEPENDENT SET ON PATHS

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

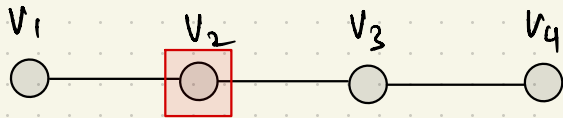


MAX WEIGHT INDEPENDENT SET ON PATHS

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

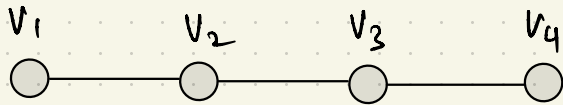


MAX WEIGHT INDEPENDENT SET ON PATHS

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



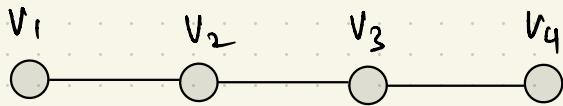
How many independent sets?

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

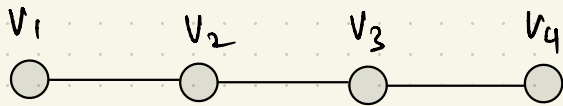
$\emptyset$, four singletons, $\{v_1, v_3\}$, $\{v_2, v_4\}$, $\{v_1, v_4\}$

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 😞

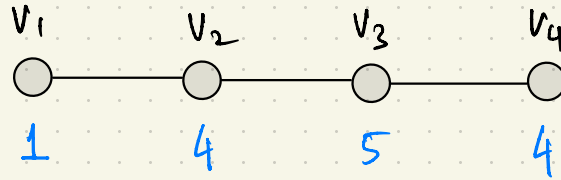
A GREEDY APPROACH

A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.

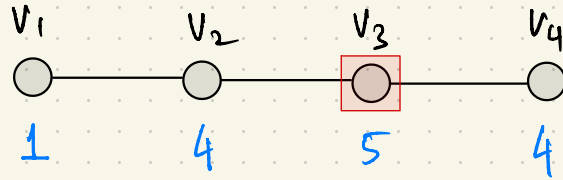
A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



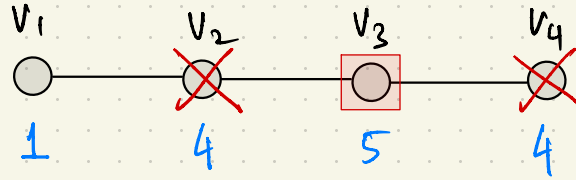
A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



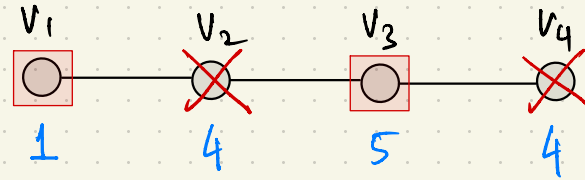
A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



A GREEDY APPROACH

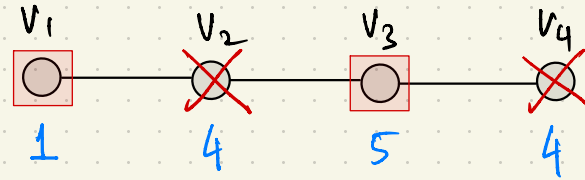
Among feasible vertices, select one with maximum weight.



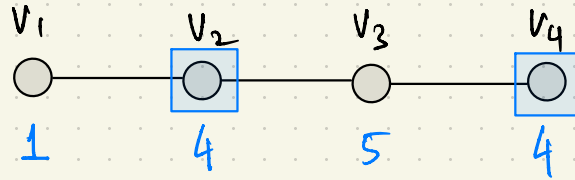
$$wt = 6$$

A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



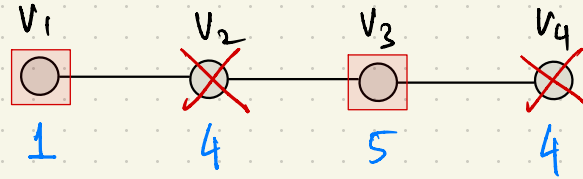
$$wt = 6$$



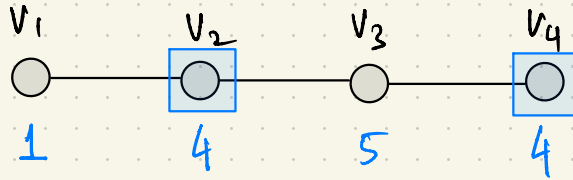
$$wt = 8$$

A GREEDY APPROACH

Among feasible vertices, select one with maximum weight.



$$wt = 6$$



$$wt = 8$$

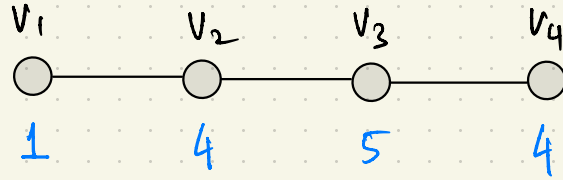
A DIVIDE & CONQUER APPROACH

A DIVIDE & CONQUER APPROACH

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

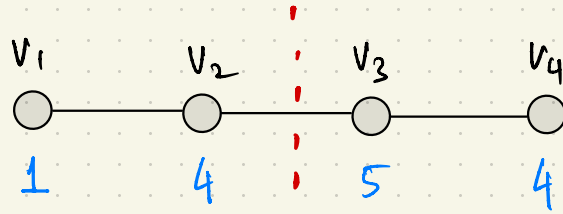
A DIVIDE & CONQUER APPROACH

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



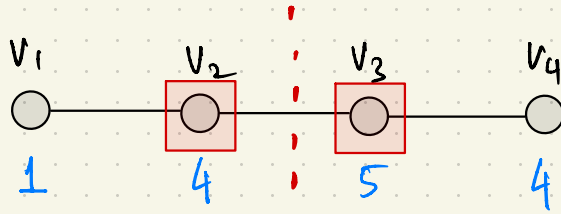
A DIVIDE & CONQUER APPROACH

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



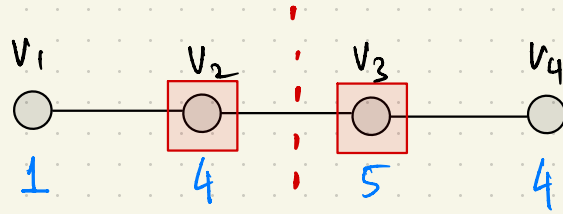
A DIVIDE & CONQUER APPROACH

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



A DIVIDE & CONQUER APPROACH

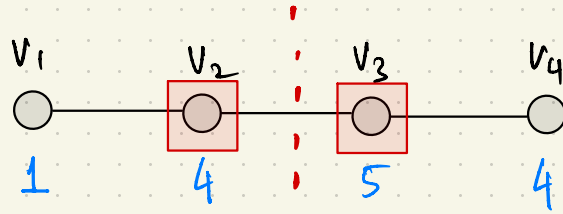
Recursively compute ind. set for left and right halves and combine the solutions



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

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

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

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

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

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

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

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

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

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

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

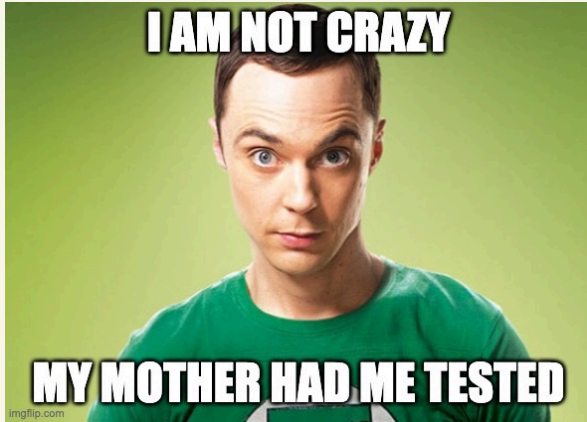
Idea: Try both possibilities and return the better solution.

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

OPTIMAL SUBSTRUCTURE

Idea: Try both possibilities and return the better solution.

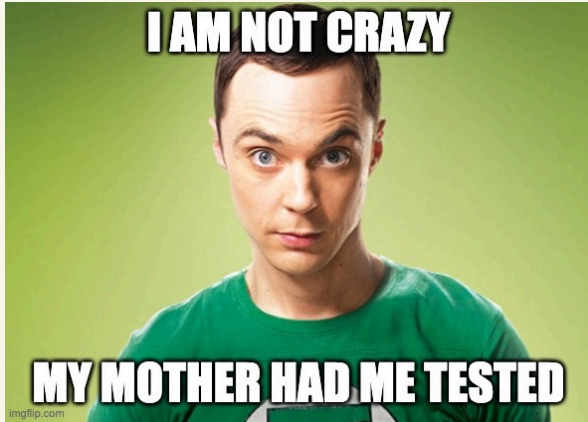
You: "Are you crazy? Isn't this just brute force?"



OPTIMAL SUBSTRUCTURE

Idea: Try both possibilities and return the better solution.

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

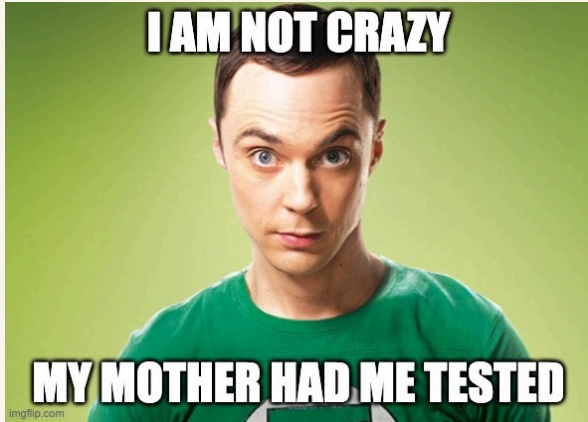


This is **recursively organized** brute force.

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.

A RECURSIVE ALGORITHM

A RECURSIVE ALGORITHM

input: a path graph G with vertices $v_1, v_2, \dots, v_n$, nonnegative weights $\{w_i\}$

output: a maximum-weight independent set of G

A RECURSIVE ALGORITHM

input: a path graph G with vertices $v_1, v_2, \dots, v_n$, nonnegative weights $\{w_i\}$

output: a maximum-weight independent set of G

if $n = 1$

return $\{v_1\}$

A RECURSIVE ALGORITHM

input: a path graph G with vertices $v_1, v_2, \dots, v_n$, nonnegative weights $\{w_i\}$

output: a maximum-weight independent set of G

if $n = 1$

return $\{v_1\}$

// recursion when $n \geq 2$

A RECURSIVE ALGORITHM

input: a path graph G with vertices $v_1, v_2, \dots, v_n$, nonnegative weights $\{w_i\}$

output: a maximum-weight independent set of G

if $n = 1$

return $\{v_1\}$

// recursion when $n \geq 2$

$S_1 :=$ recursively compute an MWIS of $G \setminus \{v_n\}$

$S_2 :=$ recursively compute an MWIS of $G \setminus \{v_{n-1}, v_n\}$

A RECURSIVE ALGORITHM

input: a path graph G with vertices $v_1, v_2, \dots, v_n$, nonnegative weights $\{w_i\}$

output: a maximum-weight independent set of G

if $n = 1$

return $\{v_1\}$

// recursion when $n \geq 2$

$S_1 :=$ recursively compute an MWIS of $G \setminus \{v_n\}$

$S_2 :=$ recursively compute an MWIS of $G \setminus \{v_{n-1}, v_n\}$

return S_1 or $S_2 \cup \{v_n\}$, whichever has higher weight

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

Proof: Exercise, by induction on number of vertices n .

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

Proof: Exercise, by induction on number of vertices n .

Claim: The recursive algorithm for MWIS takes exponential time.

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

Proof: Exercise, by induction on number of vertices n .

Claim: The recursive algorithm for MWIS takes exponential time.

Proof: $T(n) = T(n-1) + T(n-2) + O(1)$

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

Proof: Exercise, by induction on number of vertices n .

Claim: The recursive algorithm for MWIS takes exponential time.

Proof: $T(n) = T(n-1) + T(n-2) + O(1)$

branching factor = 2 ... but very little progress

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

Proof: Exercise, by induction on number of vertices n .

Claim: The recursive algorithm for MWIS takes exponential time.

Proof: $T(n) = T(n-1) + T(n-2) + O(1)$

branching factor = 2 ... but very little progress

$$T(n) \geq \text{Fibonacci}(n) \sim \text{const.}^n$$

A RECURSIVE ALGORITHM

Claim: The recursive algorithm for MWIS is correct.

Proof: Exercise, by induction on number of vertices n .

Claim: The recursive algorithm for MWIS takes exponential time.

Proof: $T(n) = T(n-1) + T(n-2) + O(1)$

branching factor = 2 ... but very little progress

$T(n) \geq \text{Fibonacci}(n) \sim \text{const.}^n$ No better than brute force



Among the exponentially-many recursive calls,
how many **distinct** subproblems are considered?

Among the exponentially-many recursive calls,
how many **distinct** subproblems are considered?

$$\Theta(n)$$

Among the exponentially-many recursive calls,
how many **distinct** subproblems are considered?

$$\Theta(n)$$

The prefixes of graph G $v_1 - v_2 - v_3 \dots v_i - v_{i+1} \dots v_n$

ELIMINATING REDUNDANCY

The first time we solve a subproblem, cache its solution in a global table for $O(1)$ time lookup later on.

ELIMINATING REDUNDANCY

The first time we solve a subproblem, cache its solution in a global table for $O(1)$ time lookup later on.

(i.e., **memoization**)

BLENDING CACHE INTO PSEUDOCODE

BLENDING CACHE INTO PSEUDOCODE

let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

BLENDING CACHE INTO PSEUDOCODE

let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

BLENDING CACHE INTO PSEUDOCODE

let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$

BLENDING CACHE INTO PSEUDOCODE

let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$

// main loop

for $i = 2, 3, \dots, n$:

BLENDING CACHE INTO PSEUDOCODE

let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$

// main loop

for $i = 2, 3, \dots, n$:

$$A[i] := \max \{ A[i-1], A[i-2] + w_i \}.$$

BLENDING CACHE INTO PSEUDOCODE

Let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$

Running time: ?

// main loop

for $i = 2, 3, \dots, n$:

$$A[i] := \max \{ A[i-1], A[i-2] + w_i \}.$$

BLENDING CACHE INTO PSEUDOCODE

Let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$

Running time: $O(n)$

// main loop

for $i = 2, 3, \dots, n$:

$$A[i] := \max \{ A[i-1], A[i-2] + w_i \}.$$

BLENDING CACHE INTO PSEUDOCODE

Let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$

// main loop

for $i = 2, 3, \dots, n$:

$$A[i] := \max \{ A[i-1], A[i-2] + w_i \}.$$

Running time: $O(n)$

Correctness: same as recursive algorithm

BLENDING CACHE INTO PSEUDOCODE

let $G_i :=$ first i vertices of graph G and corresponding edges

$$v_1 - v_2 - \dots - v_{i-1} - v_i$$

Plan: populate array A left to right with $A[i] =$ value of MWIS of G_i .

// initialization

$$A[0] = 0$$

$$A[1] = w_1$$



Computes the **weight** of Ind Set,
but **not the set itself**

// main loop

for $i = 2, 3, \dots, n$:

$$A[i] := \max \{ A[i-1], A[i-2] + w_i \}.$$

RECONSTRUCTION ALGORITHM

RECONSTRUCTION ALGORITHM

Idea: Trace back through array A to reconstruct optimal solution.

RECONSTRUCTION ALGORITHM

Idea: Trace back through array A to reconstruct optimal solution.

Key point:

vertex v_i belongs to
MWIS of G_i $\iff w_i +$
MWIS of $G_{i-2} \geq$ MWIS of G_{i-1}

RECONSTRUCTION ALGORITHM

Idea: Trace back through array A to reconstruct optimal solution.

Key point:

vertex v_i belongs to
MWIS of G_i $\iff w_i +$
MWIS of $G_{i-2} \geq$ MWIS of G_{i-1}

$S := \emptyset$

While $i \geq 1$

if $A[i-1] \geq A[i-2] + w_i$
decrease i by 1

else

add v_i to S , decrease i by 2

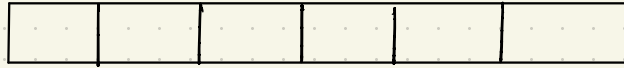
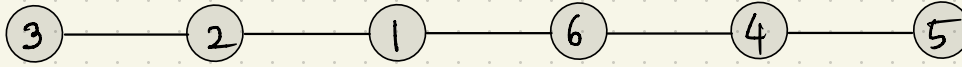
return S

// scan from right to left

// $v_i \notin$ MWIS

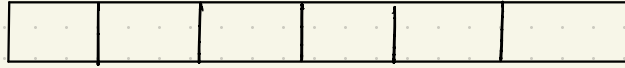
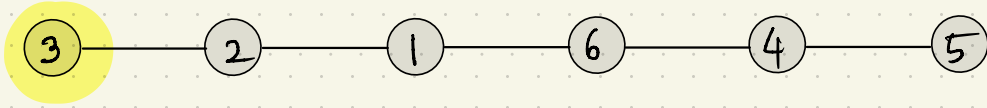
// $v_i \in$ MWIS

EXAMPLE

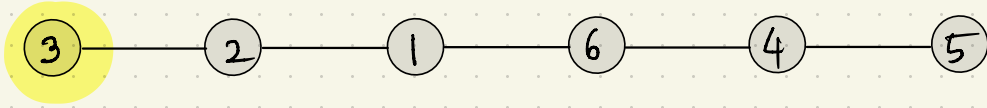


A

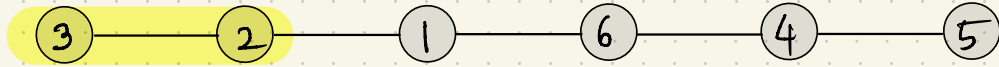
EXAMPLE



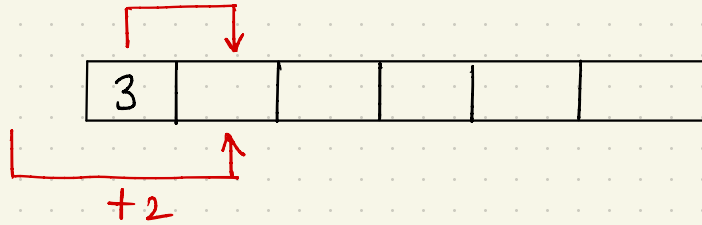
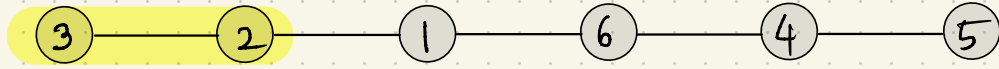
EXAMPLE



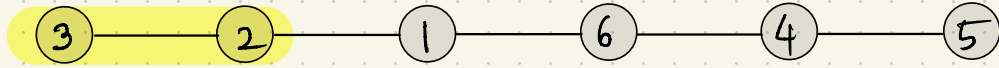
EXAMPLE



EXAMPLE



EXAMPLE



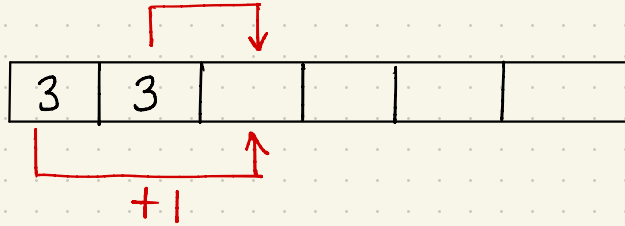
3	3				
---	---	--	--	--	--

EXAMPLE



3	3				
---	---	--	--	--	--

EXAMPLE

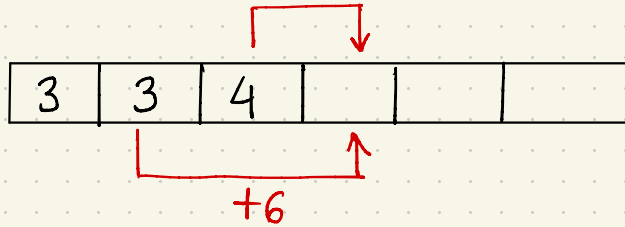
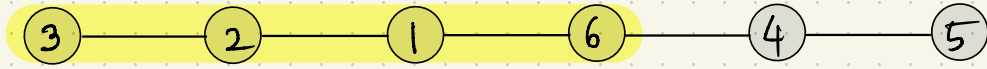


EXAMPLE

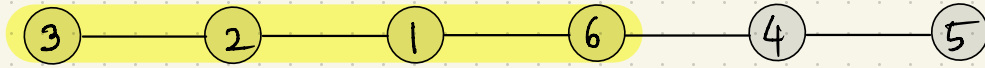


3	3	4			
---	---	---	--	--	--

EXAMPLE

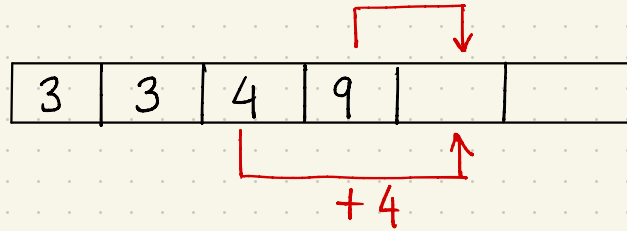
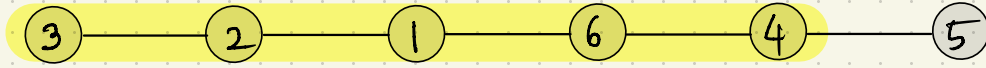


EXAMPLE

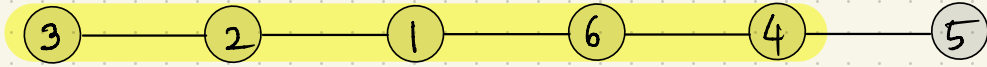


3	3	4	9		
---	---	---	---	--	--

EXAMPLE

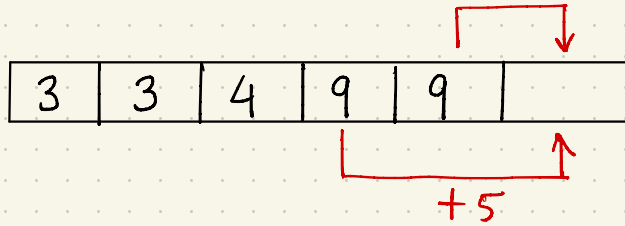
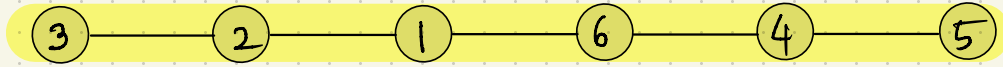


EXAMPLE

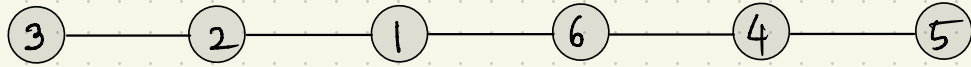


3	3	4	9	9	
---	---	---	---	---	--

EXAMPLE

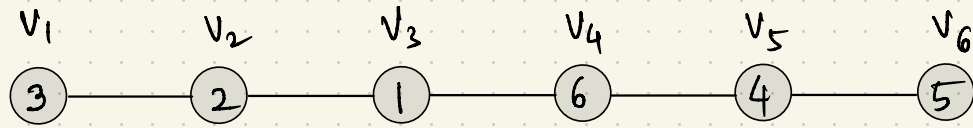


EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

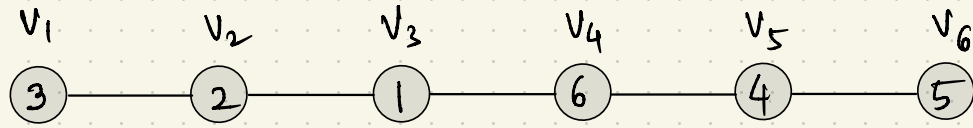
EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

Reconstruction $S = \{ \quad \}$

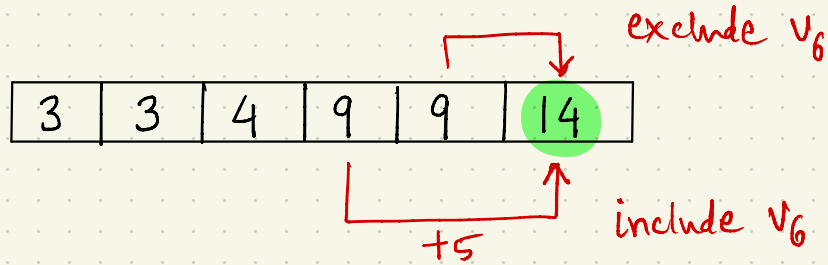
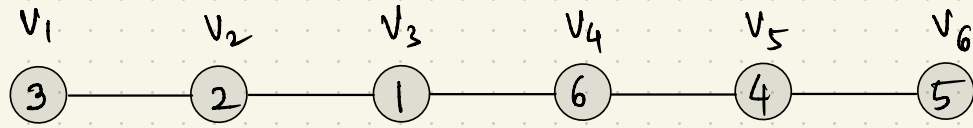
EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

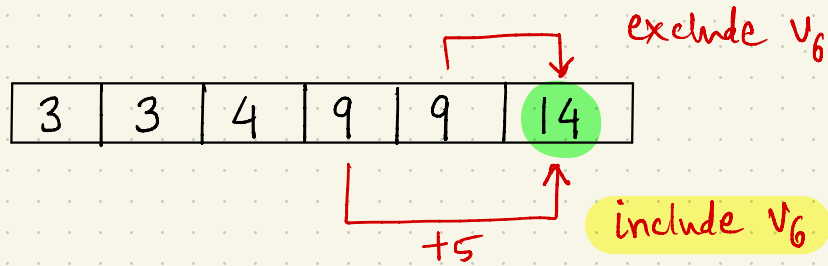
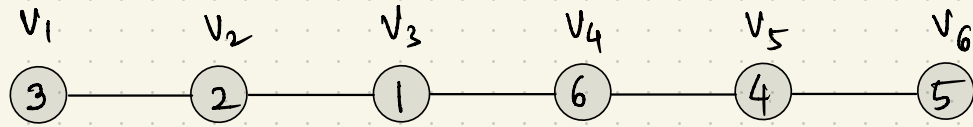
Reconstruction $S = \{ \quad \}$

EXAMPLE



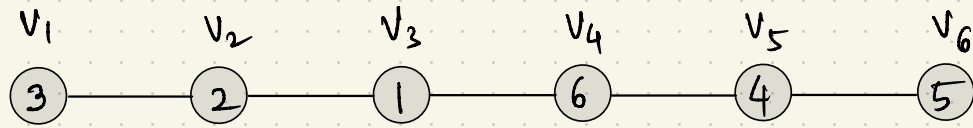
Reconstruction $S = \{ \quad \}$

EXAMPLE



Reconstruction $S = \{ \quad v_6 \}$

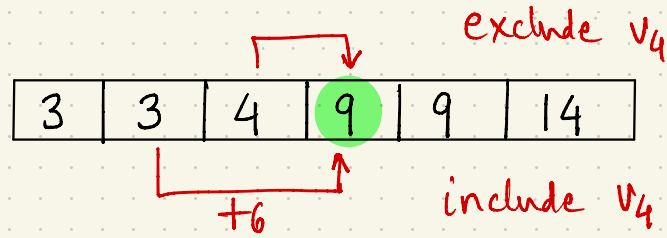
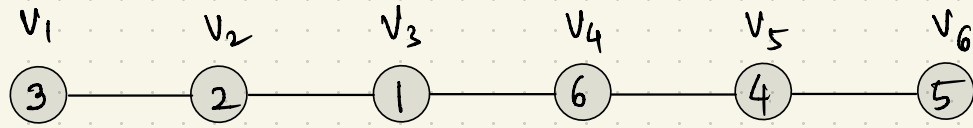
EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

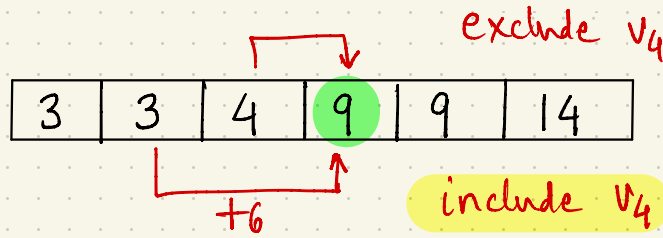
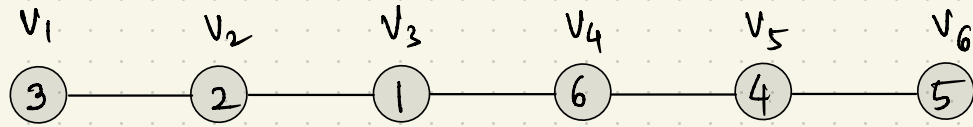
Reconstruction $S = \{ \quad v_6 \}$

EXAMPLE



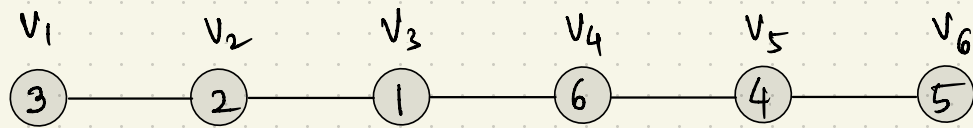
Reconstruction $S = \{ \quad v_6 \}$

EXAMPLE



Reconstruction $S = \{ v_4, v_6 \}$

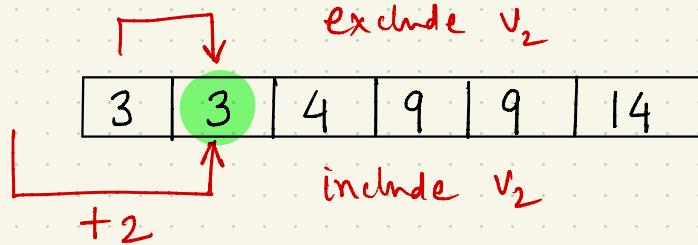
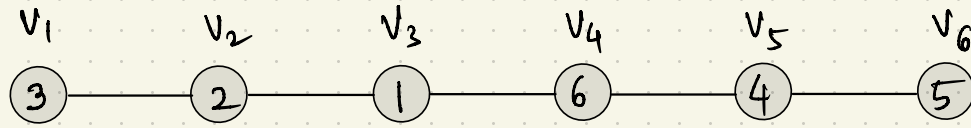
EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

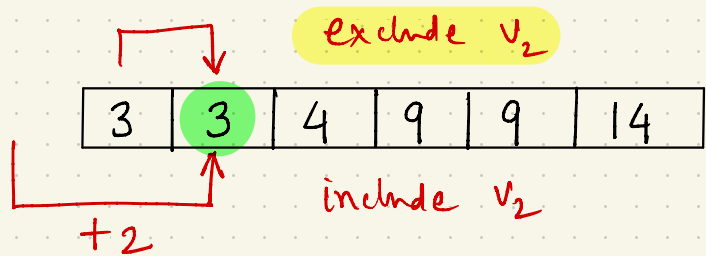
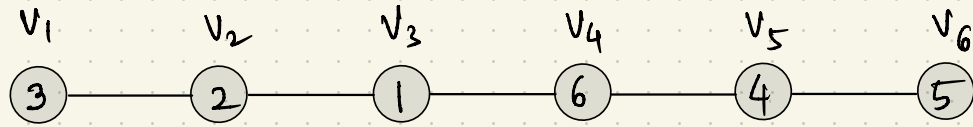
Reconstruction $S = \{ v_4, v_6 \}$

EXAMPLE



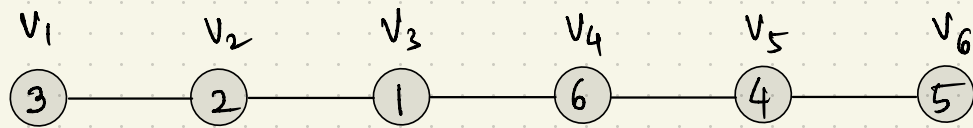
Reconstruction $S = \{ v_4, v_6 \}$

EXAMPLE



Reconstruction $S = \{ \cancel{v_2} \ v_4 \ v_6 \}$

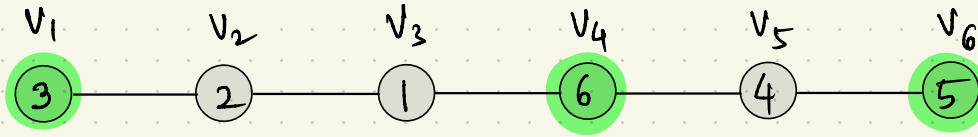
EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

Reconstruction $S = \{v_1 \ v_4 \ v_6\}$

EXAMPLE



3	3	4	9	9	14
---	---	---	---	---	----

Reconstruction $S = \{v_1 \ v_4 \ v_6\}$