Knapsack

Knapsack

0-1 Knapsack

Example

Time Complexity

Dynamic Programming

0-1 Knapsack

Given n objects with integer weights w_i and values v_i, what is the most valuable subset that weighs $\leq W$

Give an algorithm that runs in $O(nW)$ time.

0-1 Knapsack

Given n objects with integer weights w_i and values v_i, what is the most valuable subset that weighs $\leq W$

Give an algorithm that runs in $O(nW)$ time.

Example

Calculate the best value you can store in a knapsack with $W = 7$, based on the following price table:

weight w_i	3	2	1	5	4
price p_i	9	7	3	9	10

How can we reconstruct the solution (decide which items to include to get the best price)?

Example

Calculate the best value you can store in a knapsack with $W = 7$, based on the following price table:

weight w_i	3	2	1	5	4
price p_i	9	7	3	9	10

How can we reconstruct the solution (decide which items to include to get the best price)?

Time Complexity

What is the length of the input?

pseudo-polynomial time: polynomial if the magnitude of the input numbers is polynomial in the input size

Does this apply to counting sort?

Time Complexity

What is the length of the input?

pseudo-polynomial time: polynomial if the magnitude of the input numbers is polynomial in the input size

Does this apply to counting sort?

Time Complexity

What is the length of the input?

pseudo-polynomial time: polynomial if the magnitude of the input numbers is polynomial in the input size

Does this apply to counting sort?

Dynamic Programming

Summary of Dynamic Programming

1. optimal substructure: global optimum uses optimal solutions of subproblems
2. ordering of subproblems: solve ‘smallest’ first, build larger solutions from smaller
3. ‘overlapping’ subproblems: polynomial number of subproblems, used multiple times
4. independent subproblems: optimal solution of one subproblem doesn’t affect optimality of another

- top-down: memoization
- bottom-up: compute table, then recover solution