http://www.cs.unh.edu/~ruml/cs758
DP

- Counting
- 0-1 Knapsack
- Time Complexity
- Break

More DP
You are late for a meeting that is held on the floor above your current location. You can climb the staircase one step at a time, two steps at a time, or, with great effort, three steps at a time. As you are rushing upstairs, you have a sudden flash of insight into how to count the number of ways of climbing a staircase of \( n \) steps. What is the algorithm?
Given $n$ objects with integer weights $w_i$ and values $v_i$, what is the most valuable subset that weighs at most $W$?
0-1 Knapsack

Given \( n \) objects with integer weights \( w_i \) and values \( v_i \), what is the most valuable subset that weighs at most \( W \)?

Give an algorithm that runs in \( O(nW) \) time.
Given $n$ objects with integer weights $w_i$ and values $v_i$, what is the most valuable subset that weighs at most $W$?

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

Will greedy work? What if items can be divided?
what is the length of the input?
what is the length of the input?

pseudo-polynomial time: polynomial if the magnitude of the input numbers is polynomial in the input size.
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 radix sort?
Break

DP
- Counting
- 0-1 Knapsack
- Time Complexity
- Break

More DP

- asst 5
- asst 6
- midterm
More DP

- Increasing Subseq
- EOLQs
Given a sequence of length $n$ consisting of numbers, give an $O(n^2)$ algorithm that finds the longest (not necessarily contiguous) subsequence that consists of monotonically increasing values.
Given a sequence of length $n$ consisting of numbers, give an $O(n^2)$ algorithm that finds the longest (not necessarily contiguous) subsequence that consists of monotonically increasing values.

BTW, there is an $O(n \log n)$ algorithm
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*