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

- 0-1 Knapsack
- Time Complexity
- Break

3D DP
Given \( n \) objects with integer weights \( w_i \) and values \( v_i \), what is the most valuable subset that weighs at most \( W \)?
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?
Three-dimensional Dynamic Programming
The Context-Free Grammar Parsing Problem

Given:

1. A context-free grammar $G$ in Chomsky Normal Form
   - $\epsilon$-free
   - all rules like:
     - $A \rightarrow BC$
     - $A \rightarrow a$
     
     So $A \rightarrow BCD$ becomes $A \rightarrow BX$ and $X \rightarrow CD$

2. String of length $n$ of tokens $w_i$
Bottom-up dynamic programming:

\[
\pi[i, j, A] = \text{can } A \text{ span } i-j?
\]
\[
\pi[i, i, A] =
\]
Bottom-up dynamic programming:

\[ \pi[i, j, A] = \text{can } A \text{ span } i-j? \]
\[ \pi[i, i, A] = \text{true iff } A \to w_i \]
\[ \pi[i, j, A] = \]
Dynamic Programming

Bottom-up dynamic programming:

\[
\pi[i, j, A] = \text{can } A \text{ span } i-j?
\]
\[
\pi[i, i, A] = \text{true iff } A \rightarrow w_i
\]
\[
\pi[i, j, A] = \bigvee_{A \rightarrow BC} \bigvee_{i \leq k < j} \pi[i, k, B] \land \pi[k + 1, j, C]
\]
1. initialize $\pi$ to false everywhere
2. for $i$ from 1 to $n$
3.    foreach nonterminal $A$
4.        $\pi[i, i, A] \leftarrow \text{true if } A \rightarrow w_i$
5. for $\text{len}$ from 2 to $n$
6.    for $i$ from 1 to $n - (\text{len} - 1)$
7.        $j \leftarrow i + (\text{len} - 1)$
8.    for $k$ from $i$ to $j - 1$
9.        foreach rule $A \rightarrow BC$
10.        if $\pi[i, k, B]$ and $\pi[k + 1, j, C]$ then
11.            $\pi[i, j, A] \leftarrow \text{true}$
12. return $\pi[1, n, S]$
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!*