Recall the definition of **Knapsack**

**Input**
- Items 1, ..., n \([n]\)
- Weights \(w_1, \ldots, w_n\) positive integers
- Values \(v_1, \ldots, v_n\) positive integers
- \(W_{\text{max}}\)

For \(S \subseteq \{1, \ldots, n\}\) define
\[
W(S) = \sum_{i \in S} w_i \\
V(S) = \sum_{i \in S} v_i
\]

**Output**
- \(S_{\text{opt}} \subseteq \{1, \ldots, n\}: V(S_{\text{opt}}) = \max_{S \subseteq \{1, \ldots, n\}} V(S)\)
- \(W(S) \leq W_{\text{max}}\)
This is also pseudo-polynomial in worst case.

Running time poly \((n, \log \text{max}, \text{max})\)

Finds Sort

Dynamic Programming Algorithm

\(0 (n \log \text{max})\)
THEOREM: Dynamic programming algorithm which, on input an instance of KNAPSACK and \( \varepsilon, 0 < \varepsilon < 1 \) finds \( S_{\text{approx}} \) such that \( V(S_{\text{approx}}) \geq (1-\varepsilon)V(S_{\text{opt}}) \).

Running time \( \text{poly}(n, \log W_{\text{max}}, \log v_{\text{max}}, \frac{1}{\varepsilon}) \)

Actually, \( O\left(n^2 \log v_{\text{max}} \frac{1}{\varepsilon}\right) \) (plus calculations with \( \log v_{\text{max}}, \log W_{\text{max}}, \log \frac{1}{\varepsilon} \) bits).
PROOF:

We have to give an algorithm
We have to prove its running time
We have to prove approximation performance guarantee

The algorithm will be a dynamic programming algorithm on the exact same instance of KNAPSACK. But with scaled down values $v_i$

We need to scale down the values enough to make the table of algorithm $B$ small enough so that the running time is polynomial.

But as we are scaling down, we are losing in accuracy. We cannot scale down too much since we want to prove a good approximation.
scaled-down knapsack input

Items $1, \ldots, n$

weights $w_1, \ldots, w_n$

Values $\hat{V}_i = \left[ \frac{V_i}{\alpha} \right]$

$W_{\text{max}}$

Run Dynamic Programming Algorithm (B) on scaled-down input

Output (will be) $\hat{S}_{\text{opt}} \subseteq [n]: V(\hat{S}_{\text{opt}}) = \max_{S \subseteq [n]} \hat{V}(S)$

$W(S) \leq W_{\text{max}}$

$S_{\text{approx}} = \hat{S}_{\text{opt}}$

Looking for suitable $\alpha$:

try 1: $\alpha = V_{\text{max}}$

wait, this would be too much, would make all $\hat{V}_i = 0$

try 2: $\alpha = \frac{V_{\text{max}}}{\eta}$

is better, still keeps the table polynomial

try 3: $\alpha = \frac{V_{\text{max}} \varepsilon}{\eta}$

also still keeps the table polynomial
For the approximation performance guarantee we must compare

\[ V(S_{\text{approx}}) = \sum_{i \in S_{\text{approx}}} V_i \quad \text{to} \quad V(S_{\text{opt}}) = \sum_{i \in S_{\text{opt}}} V_i \]

But we only know how to compare \( V_i \) and \( \hat{V}_i \):

\[ \hat{V}_i = \left\lfloor \frac{V_i}{\alpha} \right\rfloor \Rightarrow \frac{V_i}{\alpha} - 1 \leq \hat{V}_i \leq \frac{V_i}{\alpha} \]

We will also need (would have done this calculation at the end but I write it here to save space).

If \( \alpha = \frac{V_{\text{max}}}{n} \) then \( an = V_{\text{max}} \varepsilon \).

And since \( V_{\text{max}} \leq V(S_{\text{opt}}) \) we have \( an \leq EV(S_{\text{opt}}) \).
We are ready to prove the performance guarantee.

\[ V(S_{\text{approx}}) = V(S_{\text{opt}}) \]

\[ = \sum_{i \in \hat{S}_{\text{opt}}} V_i \]

\[ \geq \alpha \sum_{i \in \hat{S}_{\text{opt}}} \hat{V}_i \quad \text{by (1)} \]

because \( \hat{S}_{\text{opt}} \) is optimal under values \( \hat{V}_i \)

\[ \geq \alpha \sum_{i \in \hat{S}_{\text{opt}}} \left( \frac{V_i}{\alpha} - 1 \right) \quad \text{by (2)} \]

\[ \geq \sum_{i \in \hat{S}_{\text{opt}}} V_i - \alpha \sum_{i \in \hat{S}_{\text{opt}}} 1 \]

\[ \geq V(S_{\text{opt}}) - \alpha n \]

\[ \geq V(S_{\text{opt}}) - \varepsilon V(S_{\text{opt}}) \quad \text{by (3)} \]

\[ = (1 - \varepsilon) V(S_{\text{opt}}) \]

**END OF PROOF**
Exercise: Complete the details of Dynamic Programming Algorithm in page 2.

Note: We chose $\alpha = \frac{V_{\text{max}} \varepsilon}{n}$ in page 5 but did not use the particular choice of $\alpha$ when I was working the performance guarantee in page 7. Realize that it is actually until the very end at the point $*$ of page 7 that we really make the choice of $\alpha$.

Question: Suppose we choose $\alpha = \frac{V_{\text{max}} \varepsilon}{n^2}$

How does this affect the running time? the performance guarantee?

Question: How small can you make $\varepsilon$? Can it be a function of $n$? What kind of function?

Comment on running time/performance guarantee.