Plan.

* Knapsack Problem
* Announcements
* Approximation Scheme
* Dynamic Program.
Knapsack Problem

* Given:
  - list of $n$ items.
    - Each item $i \in \{1, \ldots, n\}$ has weight $w_i$ and value $v_i$
  - weight bound $W$

* Find: Subset $S \subseteq \{1, \ldots, n\}$ maximizing

$$\sum_{i \in S} v_i$$

subject to
$$\sum_{i \in S} w_i \leq W.$$
$W = 16$
\[ W = 16 \]
\[ S = \{ 5, 7, 4 \} \]
\[ V_s = 11 \]
\[ W_s = 15 \]
\[ W = 16 \]

\[ S = \{ 5 \} \]

\[ V_s = 11 \]
\[ W_s = 15 \]

\[ S = \{ 1, 2, 6 \} \]

\[ V_s = 15 \]
\[ W_s = 11 \]
Two facts about Knapsack

(1) Knapsack is NP-Hard

(2) Knapsack is solvable by dynamic programming in time $O(n^2 \cdot v^*)$

where $v^* = \max_{i \in \{1 \ldots n\}} v_i$
Two facts about Knapsack

(1) Knapsack is NP-Hard

(2) Knapsack is solvable by dynamic programming in time \( O(n^2 v^*) \)

where \( v^* = \max_{i \in \mathcal{I}} v_i \)

Knapsack is "weakly" NP-Hard
Weak NP-Hardness

Hardness arises due to solving problem exactly on large values.
Weak NP-Hardness

- Hardness arises due to solving problem exactly on large value.

Idea:

- Reduce Knapsack on large values to Knapsack on small values.
Weak NP-Hardness

- Hardness arises due to solving problem exactly on large value.

Idea:

- Reduce Knapsack on large values to Knapsack on small values.

- Solution on "noisy instance" w/ small values approximates optimal solution on original instance.
Announcements

* HW9 out now.
* Fall 2024 TA Application due tonight!
* Student Survey about CIS courses WIC/URMC
DP Exact Algorithm.

* pseudo-polynomial time

Dynamic Program
DP Exact Algorithm

- pseudo-polynomial time

**Dynamic Program**

\[ v^* = \max_{i \in \mathbb{Z}_+} v_i \]

\[ n v^* \] (i.e. \( V_{\text{Max}} \))

\[ O(n^2 v^*) \] \( \text{RT} \)

\[ \text{\# n entries per value} \]
DP Exact Algorithm

* pseudo-polynomial time

Dynamic Program

\[ v^* = \max_{i \in \{1, \ldots, n\}} V_i \]

\( V^* \) (i.e. \( V_{\text{Max}} \))

\[ |T| \leq n \text{ entries per value} \]

\[ O(n^2 v^*) \text{ RT} \]

* Reduction, solve a "coarse" instance using exact DP
Reduce Knapsack to "Coarse" Knapsack.

\[
\begin{align*}
\text{Maximize} & \quad \sum_{i \in \mathcal{S}} v_i \\
\text{subject to} & \quad \sum_{i \in \mathcal{S}} w_i \leq W
\end{align*}
\]
Reduce Knapsack to "Coarse"-Knapsack.

\[
\begin{align*}
\text{max} \quad & \sum_{i \in \mathcal{I}} v_i \\
\text{s.t.} \quad & \sum_{i \in \mathcal{I}} w_i \leq W
\end{align*}
\]

\[
\Rightarrow
\begin{align*}
\text{max} \quad & \sum_{i \in \mathcal{I}} v_i \\
\text{s.t.} \quad & \sum_{i \in \mathcal{I}} w_i \leq W
\end{align*}
\]

Properties

\[
\hat{V} = \max_{i \in \mathcal{I}} \tilde{V}_i \quad \Rightarrow \quad V^* = \max_{i \in \mathcal{I}} V_i
\]
Reduce Knapsack to "Coarse"-Knapsack.

\[
\begin{align*}
\max_{i \in \{1, \ldots, n\}} & \sum_{i \in S} v_i \\
\text{s.t.} & \sum_{i \in S} w_i \leq W
\end{align*}
\rightarrow
\begin{align*}
\max_{i \in \{1, \ldots, n\}} & \sum_{i \in S} v_i \\
\text{s.t.} & \sum_{i \in S} w_i \leq W
\end{align*}
\]

Properties

* \( \bar{v} = \max_{i \in \{1, \ldots, n\}} \bar{v}_i \) \( \ll \) \( \bar{v} = \max_{i \in \{1, \ldots, n\}} v_i \)

* \( \exists b \in \mathbb{R} \) \( \text{s.t.} \) \( b \cdot \sum_{i \in S} v_i \ll \sum_{i \in S} v_i \)
Reduction

$$\forall i \in \mathbb{Z}^+ \quad \nu_i = \left\lceil \frac{\nu_i}{b} \right\rceil$$
Reduction

\[ \hat{v}_i = \left\lfloor \frac{v_i}{b} \right\rfloor \quad \forall i \in \mathbb{Z}_1 \ldots n \]

Approximation

\[ v_i \leq b \cdot \hat{v}_i = b \cdot \left\lfloor \frac{v_i}{b} \right\rfloor \leq v_i + b \]
Reduction

\[ v_i = \left\lfloor \frac{v_i}{b} \right\rfloor \]

Leave weights & W the same.

\[ \forall i \in \mathbb{Z} \quad \text{and} \quad v_i \in \mathbb{Z} \]

Approximation

\[ v_i \leq b \cdot \hat{v}_i = b \cdot \left\lfloor \frac{v_i}{b} \right\rfloor \leq v_i + b \]

Claim. Suppose \( S \) is maximum \( b \)-"coarse" solution, and \( S^* \) is maximum to original Knapsack instance.

Then,

\[ \sum_{i \in S} v_i \geq \sum_{j \in S^*} v_j - bn \]
Claim. Suppose $S$ is maximum $b$-"coarse" solution, and $S^*$ is maximum to original Knapsack instance. Then,

$$\sum_{i \in S} v_i \geq \sum_{j \in S^*} v_j - bn$$

Pf.

$$\sum_{i \in S^*} \tilde{v}_i \leq \sum_{i \in S} \tilde{v}_i \quad \text{if by maximum } S.$$
Claim. Suppose $S$ is maximum $b$-"coarse" solution, and $S^*$ is maximum to original Knapsack instance. Then,

$$\sum_{i \in S} v_i \geq \sum_{j \in S^*} v_j - bn$$

Pf.

1. $\sum_{i \in S^*} v_i \leq \sum_{i \in S} v_i$ \hspace{1cm} // by maximum $S$.

By the inequalities established:

$$\sum_{i \in S^*} v_i \leq b \cdot \sum_{i \in S^*} \tilde{v}_i \leq b \cdot \sum_{i \in S} \tilde{v}_i \leq \sum_{i \in S} v_i + b |S|$$

by @

by @

by @
So, if we can solve the b-course Knapsack problem, we can obtain approximation.

\[ V_{\text{approx}} = \sum_{i \in S} V_i \]
\[ V_{\text{exact}} = \sum_{j \in S^*} V_j \]
So, if we can solve the \( b \)-coarse Knapsack problem, we can obtain approximation.

\[
    V_{\text{approx}} = \sum_{i \in S} V_i
\]

\[
    V_{\text{exact}} = \sum_{j \in S^*} V_j
\]

**Approximation Ratio**

\[
    \frac{V_{\text{approx}}}{V_{\text{exact}}} \geq \frac{V_{\text{exact}} - bn}{V_{\text{exact}}} = 1 - \frac{bn}{V_{\text{exact}}}
\]
So, if we can solve the $b$-coarse Knapsack problem, we can obtain approximation.

$$V_{\text{approx}} = \sum_{i \in S} V_i$$

$$V_{\text{exact}} = \sum_{j \in S^*} V_j$$

**Approximation Ratio**

$$\frac{V_{\text{approx}}}{V_{\text{exact}}} \geq \left(1 - \frac{b_n}{V_{\text{exact}}}\right)^{n} = 1 - \frac{b_n}{V_{\text{exact}}}$$

$$r \geq 1 - \varepsilon \quad \text{for all} \quad \varepsilon \geq \frac{b_n}{\sum_{j \in S^*} V_j}$$
Polynomial-Time Approximation Scheme

An $r$-approximation algorithm for any $r < 1$. 
Polynomial-Time Approximation Scheme

An $r$-approximation algorithm for any $r < 1$.

Given any $\varepsilon > 0$, algorithm produces a $(1 - \varepsilon)$-approximate solution to \textsc{Knapsack} in time $O(poly(n) \cdot f(1/\varepsilon))$ for some $f(\cdot)$. 
Polynomial-Time Approximation Scheme

An $r$-approximation algorithm for any $r < 1$. Given any $\varepsilon > 0$, algorithm produces a $(1-\varepsilon)$-approximate solution to Knapsack in time $O(poly(n) \cdot f(1/\varepsilon))$ for some $f(\cdot)$. Fully PTAS $O(poly(n, 1/\varepsilon))$. 
Given $\varepsilon$, choose $\varepsilon \cdot V_{\text{exact}}$ at least the max value
(pre-process to remove infeasible values)
Given $\epsilon$, choose $\nu \leq \frac{\epsilon \cdot V^*}{n}$.

\[
\Rightarrow \quad \nu \leq \frac{\epsilon \cdot V^*}{n} = \left\lfloor \frac{n}{\epsilon} \right\rfloor .
\]

Max course value

\[
(1-\epsilon) - \text{Approximate Knapsack} \left( W, w_1, \ldots, w_n, v_1, \ldots, v_n \right)
\]

\[
\nu = \epsilon \cdot \max_{i \in \{1, \ldots, n\}} \frac{v_i}{\nu}
\]

\[
\tilde{v}_i = \left\lfloor \frac{v_i}{\nu} \right\rfloor \quad \text{for all } i \in \{1, \ldots, n\}
\]

Return $S \leftarrow \text{Knapsack} \left( W, w_1, \ldots, w_n, \tilde{v}_1, \ldots, \tilde{v}_n \right)$
Dynamic Program for Knapsack

* 2D Table $T[i, U]$

minimum weight $\sum_{j \in S_i} w_j$ where $S_i \subseteq \{1, \ldots, i\}$ and $\sum_{j \in S_i} v_j \geq U$
Dynamic Program for Knapsack

* 2D table \( T[i, U] \)

minimum weight \( \sum_{j \in S_i} w_j \) where \( S_i \leq i \) and
\[ \sum_{j \in S_i} v_j \geq U \]

- \( i \) tracks subset of elements considered.
- \( U \) tracks value

Eventually return max value \( U \) that doesn't exceed weight \( W \)
Dynamic Program for Knapsack

2D Table \( T[i, U] \)

Minimum weight \( \sum_{j \in S_{i,u}} w_j \) where \( S_{i,u} \leq \sum_{j \in S_{i,u}} v_j \) and \( \sum_{j \in S_{i,u}} v_j \geq U \)

Cases:

1. \( i \not\in S_{i,u} \) \( \Rightarrow \ T[i, U] = T[i-1, U] \)

2. \( i \in S_{i,u} \) \( \Rightarrow \ T[i, U] = w_i + T[i-1, U-v_i] \)

Need to handle case where \( < 0 \).
Knapsack DP Values

Initialize $T[i, u] = +\infty$ for $i \in \{1, \ldots, n\}$,$
   \forall u \in \{0, \ldots, W\}$. 

For $i = 1, \ldots, n$ 
   $T[i, u_0] = 0$ for $u_0 \leq 0$. 

For $i = 1, \ldots, n$. 

For $U = 1, \ldots, \sum_{j=1}^{i} v_j$ 

$T[i, u] = \min \left\{ T[i-1, u], \right.$
   \hfill $w_i + T[i-1, u-v_n] \left. \right\}$
   $+\infty$ for infeasible $U$. 

Return

$max U \text{ s.t. } T[n, u] \leq W$. 

Claim. DP returns the correct value.

Claim. DP runs in $O(n^2 \sqrt{n})$ time.

$\Rightarrow$ Theorem. Knapsack has an FPTAS.

For any $\epsilon > 0$,

$(1-\epsilon)$-approximate algorithm running in $O(n^3/\epsilon)$ time.