Greedy Algorithms Ch 16

(How to prove optimality?)

Consider variation of 0-1-Knapsack:

No bars of gold \( \Rightarrow \) bags of gold dust.

\[
\begin{array}{ccc}
\text{Bag} & \text{Value} & \text{Weight} \\
1 & $100 & 20 \text{ lbs} \\
2 & $60 & 10 \text{ lbs} \\
3 & $120 & 30 \text{ lbs}
\end{array}
\]

\[1, 2, \ldots, n\] bags of gold dust. Bag \( i \) has gold type \( i \).

Each bag has weight \( w_i > 0 \) and value \( v_i > 0 \).

Knapsack has weight capacity \( W \leq \sum w_i \) (can't take all).

Can take a fraction \( x_i \) of bag with gold type \( i \) where \( 0 \leq x_i \leq 1 \).

\[ \Rightarrow \text{Gold type } i \text{ contributes weight } w_i x_i, \text{ value } v_i x_i \]

Goal:

Find \( x_i \) for \( i = 1 \) to \( n \) that maximizes \( \sum_{i=1}^{n} v_i x_i \) s.t. \( \sum_{i=1}^{n} w_i x_i \leq W \)

\( \text{Go to} \)

Fractional Knapsack (Applications)
Soln?
1. Sort bags in decreasing order of 
weight \( \frac{v_i}{w_i} \)
2. Continuously take gold of highest 
worth until 
W reached.

*Note: Always take as much as 
possible! Greedy works!*

Frac-knapsack \((T, W)\) // \(T\) is set of \(n\) gold types/bags.
// \(W\) is knapsack weight capacity.

1. for \(i = 1\) to \(n\) 
   \(x_i = 0\)
   \(G_i = \frac{v_i}{w_i} \) //worth of type \(i\)
2. Sort \(G\) in descending order 
3. \(m = 0\) //weight to take of current type 
   \(c = 0\) //current weight of knapsack 
   while \((c < W)\) 
     if \(w_i + c \leq W\) //taking all of \(i\) does not exceed \(W\) 
       \(m = w_i\) //take all
     else 
       \(m = W - c\) //take as much as will fit 
     \(x_i = m/w_i\)
     \(c = c + m\)

*More concise: if \(w_i \leq W - c\) \(m = w_i\) \(\Rightarrow m = \min(w_i, W - c)\) 
else \(w_i > W - c \Rightarrow m = W - c\)*
\[ \text{Ex: } W = 50 \]

<table>
<thead>
<tr>
<th>Items</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 lbs</td>
<td>$60</td>
<td></td>
</tr>
<tr>
<td>20 lbs</td>
<td>$100</td>
<td></td>
</tr>
<tr>
<td>30 lbs</td>
<td>$120</td>
<td></td>
</tr>
</tbody>
</table>

\[
\frac{W_i}{W_j} = 6 \quad 5 \quad 4
\]

Running Time? \( n \) types, weight capacity \( W \).

Sort \( G \): \( O(n \log n) \)

While loop: \( O(n) \); take from all \( n \) types.

\( = O(W) = O(n) \); if each type has weight 1

Big Picture:

<table>
<thead>
<tr>
<th>Divide and Conquer</th>
<th>Dyn. Prog.</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MergeSort</td>
<td>Coin Change</td>
<td>Coin Change w/ 5s</td>
</tr>
<tr>
<td>Binary Search</td>
<td>LCS</td>
<td>Frac Knapsack</td>
</tr>
<tr>
<td>Selection</td>
<td>O(1) Knapsack</td>
<td></td>
</tr>
<tr>
<td>Min/Max from HW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Proving optimality of greedy is not trivial!

Proof is similar to Inductive Proof.

2 Properties:

1. Greedy Choice Property - some optimal solution contains the first choice made by greedy
2. Base Case: first choice is optimal.
(2) Optimal substructure property - optimal soln to whole problem contains optimal solutions to subproblems.

Inductive Step: If some sub-solution is optimal, next solution is also optimal.

Which property does Greedy Alg of choosing highest \( \frac{v_i}{w_i} \) for 01-Knapsack lack?

\[
\begin{array}{c|c|c|c}
  & 1 & 2 & 3 \\
\hline
 w & 10 \text{ lbs} & 20 \text{ lbs} & 30 \text{ lbs} \\
 v & 60 & 100 & 120 \\
\end{array}
\]

\[\frac{v_i}{w_i} = \frac{6}{10} = 0.6 \] \[\frac{v_i}{w_i} = \frac{5}{20} = 0.25 \] \[\frac{v_i}{w_i} = \frac{4}{30} = \frac{2}{15} \]

Greedy first choice: 1.

Opt: 2, 3 \( \Rightarrow \$220 \)

Proof of optimality for Greedy:

Prove (1) + (2) via proof by contradiction:

- Make a claim (what we want to prove). \( \Rightarrow \) Claim
- Assume the opposite of claim. \( \Rightarrow \) Assume \( \neg \)Claim
- Show that a contradiction occurs, e.g. to our false claim, \( \neg \) some other assumption necessary for our false claim, something obviously wrong (\( \neg \Leftrightarrow 0 \))

\( \vdash \) Claim must be true.
(1) Greedy choice property.

? What is the first choice made by greedy alg?

→ 1st choice: take as much as possible, \( x_i \), of type 1 \( \left( \frac{v_i}{w_i} \text{ is max over all types} \right) \).

(1) Claim: Greedy Knapsack has greedy choice property.

Claim: There is some optimal solution that contains \( x_i \), of type 1 \( \left( \frac{v_i}{w_i} \text{ is max over all i} \right) \).

Proof: (What is 7 Claim?) Every optimal solution contains \( \leq x_i \), of type 1.

Suppose by way of contradiction (bwoc) that every optimal solution contains \( \leq x_i \), of type 1.

\[ S = \text{an optimal solution with } \leq x_i, \text{ of type 1}. \]

Main Idea: Show that we can get a better solution than \( S \) if we take \( x_i \), of type 1.