Backtracking (CA 16 in [5a])
(10.5 Textbook)

Backtracking: Systematic way to search for a solution (optimal one) to a problem

Solution Space needs to be defined so that it contains at least one optimal solution.

Consider the knapsack Problem

\[
\text{Maximize } \sum_{i=1}^{n} p_i x_i \\
\text{subject to } \sum_{i=1}^{n} w_i x_i \leq C
\]

\( x_i \in \{0, 1\} + i \)

\( x_i = 0 \): Object \( i \) does not end in knapsack
\( x_i = 1 \): \( i \) ends in knapsack

<table>
<thead>
<tr>
<th>Objects</th>
<th>Profit</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textbook</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Computer</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>Stereo</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>Pen</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Beer</td>
<td>?</td>
<td>2</td>
</tr>
</tbody>
</table>
GC: From the remaining objects, select the one with max profit that fits into the knapsack.

\[ n = 3 \]

\[
\begin{array}{ccc}
W & 10 & 10 & 10 \\
C & 105 & 105 & 105 \\
p & 20 & 15 & 15 \\
\end{array}
\]

GC: From the remaining objects, select the one that has minimum weight that also fits into knapsack.

\[ n = 2 \]

\[
\begin{array}{ccc}
W & 10 & 20 \\
C & 25 & 25 \\
p & 5 & 5 \\
\end{array}
\]

GC: From the remaining objects, select the one with maximum \( p_i/w_i \) that fits into the knapsack.

\[ n = 3 \]

\[
\begin{array}{ccc}
W & 20 & 15 & 15 \\
C & 30 & 30 & 30 \\
p & 40 & 25 & 25 \\
\end{array}
\]
Solution space for 0/1 knapsack with n=3

We need to organize solution space so that it can be searched easily via depth-first search.
Every node will be determined to be feasible or not during the search. Backtracking visits nodes in depth-first search order, but it only visits feasible nodes. When visiting a node, it calls it E-node.

----- Visiting of E-nodes

The nodes from root to E-nodes are called live nodes.
0/1 Knapsack m=3

<table>
<thead>
<tr>
<th>w</th>
<th>20</th>
<th>15</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>40</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

**EXAMPLE**

\[ c=30 \]

\[ \text{Remaining Capacity} \]
\[ \text{cp: Current Profit} \]

**E-node**

<table>
<thead>
<tr>
<th>A</th>
<th>( w=10 )</th>
<th>( cp=90 )</th>
<th>( \text{Can Move} )</th>
<th>( \text{To} )</th>
<th>( \text{Line} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>( w=10 )</td>
<td>( cp=90 )</td>
<td>E</td>
<td>(A,B)</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>( w=10 )</td>
<td>( cp=90 )</td>
<td>K</td>
<td>(A,B,E)</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>( w=10 )</td>
<td>( cp=90 )</td>
<td>-</td>
<td>(A,B,E,K)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>( w=30 )</td>
<td>( cp=30 )</td>
<td>For G</td>
<td>(A,C)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>( w=15 )</td>
<td>( cp=25 )</td>
<td>Lor M</td>
<td>(A,C,F)</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>( w=20 )</td>
<td>( cp=50 )</td>
<td>-</td>
<td>(A,C,F,G)</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>( w=15 )</td>
<td>( cp=25 )</td>
<td>-</td>
<td>(A,C,F,M)</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>( w=30 )</td>
<td>( cp=0 )</td>
<td>Non O</td>
<td>(A,C,G)</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>( w=15 )</td>
<td>( cp=15 )</td>
<td>-</td>
<td>(A,C,G,H)</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>( w=30 )</td>
<td>( cp=0 )</td>
<td>done!!</td>
<td>(A,C,G,O)</td>
<td></td>
</tr>
</tbody>
</table>

**Best Solution**

**Best Profit**

So far

Improvement

So
Space: One path in the tree

Time: Worst case is $2^n$

Bounding function:

- In the above example (previous page) we only test for feasibility, weight used $\leq$ capacity.

- We could use something more sophisticated (to eliminate nodes like $G_{1,1,0}$)

Visit a node if the current profit + max profit from remaining elements $\geq$ "just an estimate" of

\[ P_1 = 40 \quad P_2 = 25 \quad P_3 = 25 \]
\[ w_1 = 20 \quad w_6 = 15 \quad w_3 = 15 \]

remaining capacity is 30

add object 1 + $\frac{2}{3}$ of object 2

max possible profit $\leq 40 + \frac{2}{3} \cdot 25$
Branch and Bound: Another way to systematically search a solution space.

- All feasible nodes considered so far are added to a list.
- One of the nodes in the list is deleted (according to some criteria) and called E-node.
- All the children (feasible ones) are added to the list.
- This process is continued till the list is empty.

Selection of node from list

- FIFO: Extract in same order in which they were put in (breadth-first search)

- Max Profit so far
- Max Remaining:
- Select the node with max profit so far or max remaining capacity.
0/1 Knapsack

n = 3  C = 30
w = 20  15  15
p = 40  25  25

FIFO: Extraction

E_NODE   LIST
A    (B, C)
B    (C, E)
C    (E, F, G)
E    (F, G, K)
F    (G, K, L, M)
G    (K, L, M, N, O)
K    (L, M, N, O)

Note: Procedure could be made to stop adding nodes to the list when they are leaves.
**Extraction: Max Profit so far**

0/1 Knapsack n=3  c=30

\[ \begin{align*}
\text{w} & : 20 \ 15 \ 15 \\
\text{p} & : 40 \ 25 \ 25
\end{align*} \]

E-node List (node, profit)

A  ( (B,40), (C,0))
B  ( (E,40), (C,0))
E  ( (K,40), (C,0))
K  ( (C,0))
C  ( (F,25), (G,0))
F  ( (L,50), (M,25)/(G,0))

Profit 50 <= L  ( (M,25), (G,0))
M  ( (G,0))
G  ( (N,25), (D,0))
N  ( (0,0))
O  ---

**Note:** No need to place leaves in list. Procedure can be changed to avoid this.
USE OF BOUNDING FUNCTION

SEE THE NODE'S PROMISE

Find upper bound on the max profit that can possibly be obtained by expanding the node.

Consider knapsack objects in decreasing order of \( \frac{P_i}{W_i} \)

Fill up remaining capacity in that order. If an object partially fits, then do only part of the object

<table>
<thead>
<tr>
<th>Object</th>
<th>( P_i )</th>
<th>( W_i )</th>
<th>( \frac{P_i}{W_i} )</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
<td>0.5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>20</td>
<td>0.25</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>50</td>
<td>0.2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>

Remaining capacity: 40

\[ \frac{40}{5} = \frac{1}{5} \text{ of object 3} \]

\[ \text{MRP} = \text{max Remaining Profit} = \frac{2}{12} \]

Explore node if \( \frac{P_i + \text{MRP}}{W_i} > \text{Best Profit so far} \)
Total number of visitors to this page, since 4/1/2001, is **000522**