Let’s build a tree for “engineering useless rings”

Step two: Create leaf nodes and then build the tree.

<table>
<thead>
<tr>
<th>char</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>5</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
</tr>
<tr>
<td>s</td>
<td>4</td>
</tr>
<tr>
<td>g</td>
<td>3</td>
</tr>
<tr>
<td>i</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>2</td>
</tr>
<tr>
<td>l</td>
<td>1</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
</tr>
</tbody>
</table>

```
001001011100...
```
How to Encode Message?

- Encode the tree
- Encode the message
- Decoding is similar
CompSci 161
Winter 2023 Lecture 20:
Greedy Algorithms:
Fractional Knapsack and
Comparison of Techniques
Fractional Knapsack

- Decide \( x_i : 0 \leq x_i \leq w_i \)
- Require \( \sum_i x_i \leq W \)
- Goal: \( \max \sum_i b_i \left( \frac{x_i}{w_i} \right) = \max \sum_i \left( \frac{b_i}{w_i} \right) x_i \)

Suppose \( W = 10 \) and

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight:</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Benefit:</td>
<td>12</td>
<td>32</td>
<td>40</td>
<td>30</td>
<td>50</td>
</tr>
</tbody>
</table>
Greedy Algorithm for Fractional Knapsack

- Sort by benefit per unit wt: $\frac{b_i}{w_i}$ (smaller $i$: larger $\frac{b_i}{w_i}$)
- For each item in order
  - Take all (if possible) or remaining carrying capacity

- Suppose FSOC solution exists better than ours
- What do we know because of that?

  $i < K \quad x_i < w_i \quad \text{and} \quad x_K > 0$

  but $\frac{b_i}{w_i} > \frac{b_K}{w_K}$

- How do we improve this “better” solution?

  Decrease $x_K$ increase $x_i$. By $\min(w_i - x_i, x_K)$
Review Conversations

Why don’t these work for fractional knapsack?
- Sort by weight
- Sort by benefit

Why don’t these work for 0-1 Knapsack?
- Sort by weight
- Sort by benefit
- Sort by benefit per unit weight