L19: Greedy Algorithms, Huffman Coding
Alex Steiger
CompSci 201: Spring 2024
3/25/2024

Person in CS: Kathleen Booth

• 1922 – 2022
• British Mathematician, PhD in 1950
• Worked to design the first assembly language for early computer designs in the 1950s
• May have been the first woman to write a book on programming
• Early interest in neural networks

Logistics, Coming up

• Today: Monday 3/25
  • Project P4: Autocomplete due

• This Wednesday 3/27
  • APT 7 (tree recursion problems) due

• Monday 4/1
  • Nothing due—P5 the week after
Today's agenda

• Solve HeightLabel APT

• Introduce Greedy Algorithms

• Huffman Coding (PS: Huffman)
  • Uses trees and greedy algorithms
  • Back to more tree-based data structures on Wed

---

Balance and Trees

Balanced

For each node, left and right subtrees have roughly equal number of nodes.

Unbalanced

One subtree has many more nodes than the other.

---

Recurrence relation and runtime for traversing a balanced tree

• $T(n)$ time for $\text{count}(\text{tree})$ with $n$ nodes (balanced)

```java
public int count(TreeNode tree) {
    if (tree == null) {
        return 0;
    }
    return 1 + count(tree.left) + count(tree.right);
}
```

• $T(n) = 2T(n/2) + O(1)$
• $= O(n)$
Recurrence relation and runtime for traversing unbalanced tree

- $T(n)$ time for $count(tree)$ with $n$ nodes (unbalanced)

```java
public int count(TreeNode tree) {
    if (tree == null) {
        return 0;
    }
    return 1 + count(tree.left) + count(tree.right);
}
```

- $T(n) = T(1) + T(n-1) + O(1)$
- $= O(1) + T(n-1) + O(1)$
- $= O(n)$

Balance Binary Search Tree Runtime (add, contains)

Balanced

Unbalanced

$T(n) = T(n/2) + O(1)$
$= O(log(n))$

$T(n) = T(n-1) + O(1)$
$= O(n)$

We will return to this problem later!

HeightLabel APT

https://www2.cs.duke.edu/csed/newapt/heightlabel.html

- Create a new tree from a tree parameter
  - Same shape, nodes labeled with height
  - Use new TreeNode. With what values ...
Solving HeightLabel in Pictures

Base case: when null, 0
Recursive case: height of node is 1 + max(height of node.left, height of node.right)

Live Coding HeightLabel

Solving HeightLabel in Code

private int height(TreeNode t) {
    if (t == null) return 0;
    return 1 + Math.max(height(t.left), height(t.right));
}

public class HeightLabel {
    public TreeNode rewrite(TreeNode t) {
        // Keeps track of what tree is returned
        return null;
    }
}

public TreeNode rewrite(TreeNode t) {
    if (t == null) return null;
    return new TreeNode(height(t),
                        rewrite(t.left),
                        rewrite(t.right));
}
Rewire runtime?

• Recurrence of this correct code? \( T(n) = \)

\[ \begin{align*}
2T(n/2) + O(n) & \quad \text{Balanced tree} \\
T(n-1) + O(n) & \quad \text{Unbalanced}
\end{align*} \]

```java
public TreeNode rewire(TreeNode t) {
    if (t == null) return null;
    return new TreeNode(rewire(t.left), rewire(t.right));
}
```

```java
private int height(TreeNode t) {
    if (t == null) return 0;
    return 1 + Math.max(height(t.left), height(t.right));
}
```

HeightLabel Complexity

• Balanced?
  - \( T(N) = 2T(n/2) + O(n) \rightarrow O(N \log N) \)

• Unbalanced,
  - \( T(N) = T(N-1) + O(N) \rightarrow O(N^2) \)

• Doable in \( O(N) \) time? Yes, if we don't call height
  - Balanced: \( T(N) = 2T(N/2) + O(1) \)
  - Unbalanced: \( T(N) = T(N-1) + O(1) \)

HeightLabel in \( O(N) \) time

• If recursion works, subtrees store heights!

```java
public TreeNode rewire(TreeNode t) {
    if (t == null) return null;
    TreeNode leftTree = rewire(t.left);
    TreeNode rightTree = rewire(t.right);
    int leftHeight = 0;
    int rightHeight = 0;
    if (leftTree == null) {
        leftHeight = leftTree.info;
    }
    if (rightTree == null) {
        rightHeight = rightTree.info;
    }
    return new TreeNode(t.info, leftTree, rightTree);
}
```
Hi, Alexander. When you submit this form, the owner will see your name and email address.

* Required

1

NetID *

solutions

2

Here is the recursive height helper method we wrote (1 indexes height). Which recurrence relation best describes the runtime complexity of height as a function of N = the number of
nodes in the tree t **assuming the tree is balanced**?

```
1  private int height(TreeNode t) {
2      if (t == null) return 0;
3      return 1 + Math.max(height(t.left),
4                             height(t.right));
5  }
```

- T(N) = 2T(N/2) + O(1)
- T(N) = 2T(N/2) + O(N)
- T(N) = 2T(N-1) + O(1)
- T(N) = T(N-1) + O(1)

Here is another version of the height helper method. Will it work correctly?

```
1  private int height(TreeNode t) {
2      int value = 1 + Math.max(height(t.left), height(t.right));
3      if (t == null) return 0;
4      return value;
5  }
```
Yes, this is equivalent to the previous approach

No, this will not compile

No, this computes different height values

No, this generates a runtime exception

Here is an equivalent version of the rewire method that uses the height helper method. Which recurrence relation best describes the runtime complexity of rewire as a function of N = the number of nodes in the tree t assuming the tree is balanced? You may assume that the height method is O(N).

```java
8     public TreeNode rewire(TreeNode t) {
9         if (t == null) return null;
10        int myHeight = height(t);
11        TreeNode lchild = rewire(t.left);
12        TreeNode rchild = rewire(t.right);
13        return new TreeNode(myHeight, lchild, rchild);
14    }
```

- T(N) = 2T(N/2) + O(1)
- T(N) = 2T(N/2) + O(N)

The height of the tree, because we start at the root

1, because a leaf node will be the first created (and this is 1-indexed)
This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Microsoft Forms | AI-Powered surveys, quizzes and polls  Create my own form
Privacy and cookies | Terms of use
Greedy Algorithms for Discrete Optimization

Optimization

- Find the solution that maximizes or minimizes some objective

- Example: **Knapsack**
  - Find the bundle of items with maximum value without exceeding a budget.
  - What should you buy if you have $10?
  - (Only one of each item)

<table>
<thead>
<tr>
<th>Items</th>
<th>Value</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>$10</td>
<td></td>
</tr>
</tbody>
</table>

Greedily Searching for Optima

- Start with a partial solution. In each iteration make a step toward a complete solution.

  - **Greedy principle**: In each iteration, make the step that "best improves" the solution (e.g., the lowest cost or highest value step).

  - Knapsack example:
    - Partial solution is a set of items you can afford
    - Greedy step: Add the item with best value per cost ratio that you can afford with remaining money
Local Optima vs Global Optima?

Greedy algorithms do not always guarantee to find the best overall solution, called global optima.

<table>
<thead>
<tr>
<th>Items</th>
<th>Value</th>
<th>Cost</th>
<th>Value/Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>2</td>
<td>$1</td>
<td>2</td>
</tr>
<tr>
<td>banana</td>
<td>1</td>
<td>$1</td>
<td>1</td>
</tr>
<tr>
<td>pizza</td>
<td>12</td>
<td>$10</td>
<td>1.2</td>
</tr>
</tbody>
</table>

0. Start with $10
1. Buy apple, best value/cost, $8 remaining
2. Buy banana (can’t afford pizza), $7 remaining
3. Done: Can’t afford any more items.

But just buying the pizza has value 12, which is the (only) global optimum.

Total value of items = 3

Why Learn Greedy Algorithms?

1. Sometimes a greedy algorithm is optimal (always returns global optima). Examples:
   • Huffman Compression (today, Project 5)
   • Computing shortest paths in networks/graphs
2. Sometimes the greedy algorithm is not optimal, but still works well in practice
3. A greedy algorithm is typically easy to start with for optimization problems.

Aside: What is Machine Learning?
Aside continued – How do you “learn a model” greedily?

- Often (in deep learning) represent a model with a neural network.
- Learn model: optimize parameters of network on data.

- How to optimize the parameters?
  - Greedy algorithm called gradient descent.
  - At each step, make a small change that best improves model performance.

Huffman Coding

Topic of Project 5: Huffman

Huffman Compression

Representing data with bits: Preferably fewer bits

- Zip
- Unicode
- JPEG
- MP3

Huffman compression used in all of these and more!
Encoding

- Eventually, everything stored as bit sequence: 01100101...

- **Fixed length encoding**
  - Each value has a unique bit sequence of the same length stored in a table.
  - With $N$ unique values to encode, need $\lceil \log_2(N) \rceil$ bits per value.
  - E.g., with 8 characters, need 3 bits per character.

Optimizing Encoding?

- Suppose we have three characters {a, b, c}:
  - a appears 1,000,000 times
  - b and c appear 50,000 times each

- Fixed length encoding uses 2,200,000 bits:
  - $\lceil \log_2(3) \rceil = 2$ bits per character
  - 2 bit/char * 1,100,000 chars = 2,200,000 bits

- **Variable length encoding**: Use fewer bits to encode more common values, more bits to encode less common values.
  - What if we encode: a = 1, b = 10, c = 11?
  - Only uses 1,200,000 bits.

Decoding Fixed Length

- **Fixed Length with length k**
  - Every k bits, look up in table
  - 001 001 010 110
    - 001 → o
    - 001 → o
    - 010 → p
    - 110 → s
Decoding Variable Length

• What if we use
  • a = 1
  • b = 10
  • c = 11

• How would we decode 1011?
  • “baa” or “bc”?

• Problem: Encoding of a (1) is a prefix of the encoding for c (11). Ambiguous!

Prefix Property: Encoding as a Tree

Huffman Coding

• Greedy algorithm for building an optimal variable-length encoding tree.

• High level idea:
  • Start with the leaves/values you want to encode with weights = frequency. Then repeat until all leaves are in single tree:
    • Greedy step: Choose the lowest-weight nodes to connect as children to a new node with weight = sum of children.

• Implementation? Priority queue!
Visualizing the Algorithm

Encoding the text "go go gophers"

<table>
<thead>
<tr>
<th>char</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>00</td>
</tr>
<tr>
<td>o</td>
<td>01</td>
</tr>
<tr>
<td>g</td>
<td>1110</td>
</tr>
<tr>
<td>o</td>
<td>1101</td>
</tr>
<tr>
<td>g</td>
<td>101</td>
</tr>
<tr>
<td>e</td>
<td>1111</td>
</tr>
<tr>
<td>r</td>
<td>1100</td>
</tr>
<tr>
<td>h</td>
<td>100</td>
</tr>
</tbody>
</table>

P5 Outline

1. Write Decompress first
   - Takes a compressed file (we give you some)
   - Reads Huffman tree from bits
   - Uses tree to decode bits to text

2. Write Compress second
   - Count frequencies of values/characters
   - Greedy algorithm to build Huffman tree
   - Save tree and file encoded as bits

Diameter Problem

leetc ode.com/problems/diameter-of-binary-tree

Calculate the diameter of a binary tree, the length of the longest path (maybe through root, maybe not, can’t visit any node twice).