RB-Tree Augmentation

- Augment x with Size(x), where
  - Size(x) = size of subtree rooted at x
  - Size(NIL) = 0

OS-Rank

OS-RANK(x, y)
// Different from text (recursive version)
// Find the rank of x in the subtree rooted at y
1. r = size[left[y]] + 1
2. if x = y then return r
3. else if (key[x] < key[y]) then
   4. return OS-RANK(x, left[y])
5. else return r + OS-RANK(x, right[y])

Time Complexity O(log n)

OS-Select

OS-SELECT(x, i) // page 304
// Select the node with rank i
// in the subtree rooted at x
1. r = size[left[x]] + 1
2. if i = r then
   3. return x
4. elseif i < r then
   5. return OS-SELECT(left[x], i)
6. else return OS-SELECT(right[x], i-r)

Time Complexity O(log n)
Augmented Data Structures

- Why is it needed?
  - Because basic data structures not enough for all operations
  - Storing extra information helps execute special operations more efficiently.
- Can any data structure be augmented?
  - Yes. Any data structure can be augmented.
- Can a data structure be augmented with any additional information?
  - Theoretically, yes.
- How to choose which additional information to store.
  - Only if we can maintain the additional information efficiently under all operations. That means, with additional information, we need to perform old and new operations efficiently maintain the additional information efficiently.

How to augment data structures

1. choose an underlying data structure
2. determine additional information to be maintained in the underlying data structure,
3. develop new operations,
4. verify that the additional information can be maintained for the modifying operations on the underlying data structure.

Augmenting RB-Trees

Theorem 14.1, page 309
Let f be a field that augments a red-black tree T with n nodes, and f(x) can be computed using only the information in nodes x, left[x], and right[x], including f[left[x]] and f[right[x]].
Then, we can maintain f(x) during insertion and deletion without asymptotically affecting the O(lgn) performance of these operations.

For example,
size[x] = size[left[x]] + size[right[x]] + 1
rank[x] = ?
Examples of augmenting information for RB-Trees

- Parent
- Height
- Any associative function on all previous values or all succeeding values.
- Next
- Previous

Greedy Algorithms

- Given a set of activities \((s, f)\), we want to schedule the maximum number of non-overlapping activities.
- **Greedy-Activity-Selector** \((s, f)\)
  1. \(n = \text{length}(s)\)
  2. \(S = \{a_1\}\)
  3. \(i = 1\)
  4. for \(m = 2\) to \(n\) do
  5. \(\text{if } s_m \text{ is not before } f_i \text{ then}\)
  6. \(S = S \cup \{a_m\}\)
  7. \(i = m\)
  8. return \(S\)

Example

- \([1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14] -- Sorted by finish times\)
- \([1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]\)
- \([1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]\)
- \([1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]\)
- \([1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]\)
- \([1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]\)
Why does it work?

**THEOREM**
Let \( A \) be a set of activities and let \( a_1 \) be the activity with the earliest finish time. Then activity \( a_1 \) is in some maximum-sized subset of non-overlapping activities.

**PROOF**
Let \( S' \) be a solution that does not contain \( a_1 \). Let \( a'_1 \) be the activity with the earliest finish time in \( S' \). Then replacing \( a'_1 \) by \( a_1 \) gives a solution \( S \) of the same size.

Why are we allowed to replace? Why is it of the same size?

Then apply induction! How?

---

Greedy Algorithms – Huffman Coding

**Huffman Coding Problem**
Example: Release 29.1 of 15-Feb-2005 of TrEMBL Protein Database contains 1,614,107 sequence entries, comprising 505,947,503 amino acids. There are 20 possible amino acids. What is the minimum number of bits to store the compressed database?

~2.5 G bits or 300MB.

**How to improve this?**
In terms of frequencies are not the same.

- Ala (A) 7.72
- Gin (Q) 3.91
- Leu (L) 9.56
- Ser (S) 6.98
- Arg (R) 5.24
- Glu (E) 6.54
- Lys (K) 5.96
- Thr (T) 5.52
- Asn (N) 4.28
- Gly (G) 6.90
- Met (M) 2.36
- Trp (W) 1.18
- Asp (D) 5.28
- His (H) 2.26
- Phe (F) 4.06
- Tyr (Y) 3.13
- Cys (C) 1.60
- Ile (I) 5.88
- Pro (P) 4.87
- Val (V) 6.66

---

Huffman Coding

**IDEA 1: Use ASCII Code**
Each need at least 8 bits,
Total = 16 M bits = 2 MB

**IDEA 2: Use 4-Bit Codes**
Each need at least 4 bits,
Total = 8 M bits = 1 MB

How to Decode?
Need Unique decoding!
Easy for Ideas 1 & 2.
What about Idea 3?

---

Huffman Coding

**IDEA 3: Use Variable Length Codes**

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Code Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>000</td>
</tr>
<tr>
<td>10</td>
<td>001</td>
</tr>
<tr>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>00011</td>
<td>00010</td>
</tr>
<tr>
<td>00010</td>
<td>00001</td>
</tr>
<tr>
<td>00001</td>
<td>00000</td>
</tr>
</tbody>
</table>

2 million characters in file.

How to encode?
Need Unique decoding!
Easy for Ideas 1 & 2.
What about Idea 3?

---

Frequencies

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Code Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>000</td>
</tr>
<tr>
<td>10</td>
<td>001</td>
</tr>
<tr>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>00011</td>
<td>00010</td>
</tr>
<tr>
<td>00010</td>
<td>00001</td>
</tr>
<tr>
<td>00001</td>
<td>00000</td>
</tr>
</tbody>
</table>

2 million characters in file.

Length: ?
Expected length: ?

New approach to frequency times the code length, i.e.,
\[(22x2) + (22x2) + (18x3) + (22x1) + (6x5) + (4x6) + (4x5) + (2x5) = 2 M bits \]

2.5 million bits = 2.5 MB

---

Percentage

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Code Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>000</td>
</tr>
<tr>
<td>10</td>
<td>001</td>
</tr>
<tr>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>00011</td>
<td>00010</td>
</tr>
<tr>
<td>00010</td>
<td>00001</td>
</tr>
<tr>
<td>00001</td>
<td>00000</td>
</tr>
</tbody>
</table>

2 million characters in file.

Length: ?
Expected length: ?

New approach to frequency times the code length, i.e.,
\[(22x2) + (22x2) + (18x3) + (22x1) + (6x5) + (4x6) + (4x5) + (2x5) = 2 M bits \]

2.5 million bits = 2.5 MB
Huffman Coding
- Idea: Use shorter codes for more frequent amino acids and longer codes for less frequent ones.

Greedy Algorithms – Other examples
- Minimum Spanning Trees (Kruskal’s & Prim’s)
- Matroid Problems
- Several scheduling problems

Dynamic Programming
- Activity Problem Revisited: Given a set of activities \((s, f)\), we want to schedule the maximum number of non-overlapping activities.
- New Approach:
  \(A_i\) = Best solution for intervals \((a_1, \ldots, a_i)\) that includes interval \(a_i\).
  \(B_i\) = Best solution for intervals \((a_1, \ldots, a_i)\) that does not include interval \(a_i\).
- Does it solve the problem to compute \(A_i\) and \(B_i\)?
- How to compute \(A_i\) and \(B_i\)?