Greedy Algorithms

Proceed by series of locally optimal decisions.

E.g. MST algorithms

- repeatedly select current lowest cost connection

Dijkstra

- repeatedly identify next nearest vertex to source

Ideal case

local optimality \rightarrow \text{global optimality}

Sometimes

local optimality \rightarrow \text{approx. global optimality}

Sometimes

\# local optimality \rightarrow \text{who knows?}
Non-stable list merging problem.

Given \( L_1, L_2, \ldots, L_n \) of lists of lengths \( l_1, l_2, \ldots, l_n \) in what order should they be merged where merge of \( L_i, L_j \) costs \( d + l_i \)?

This means merge tree of lowest cost:

\[
\begin{array}{c}
\text{cost lower left} \\
\text{cost lower right} \\
\text{cost higher left} \\
\text{cost higher right} \\
\end{array}
\]

Tree not necessarily balanced.

Solution: always merge two remaining shortest lists

(after merging, put merged list back in the list collection).

Locally: min-cost merge.

**Theorem:** This procedure also yields min-cost merge order.

Proof: See pages 5-6 below.

Diagram:

- Lower cost node
- Swap
- Savings \( (l_a - l_b) \), height difference
- Non-minicost merge
Huffman coding.

Encode symbols \( s_1, s_2, \ldots, s_n \) in binary where \( s_i \) occurs with frequency \( f_i \), so that no code is a prefix (an initial part) of another.

i.e. can represent code using a tree:

```
       0
      / \ 1
     /   
    0    1
   / \   /
  0  0  1 1
 /       /  
S_1 S_2 S_3 S_4
```

Binary string always decodes:

0 1 0 0 1 1 1 0 0 0

No need for spaces.

\( s_2, s_1, s_3, s_4, s_1, s_5 \)

Task: Let \( d_i \) be length of code for \( s_i \).

Find a code s.t.

\[ \sum d_i \] is minimized

i.e. weighted code length is minimized.
Solution: Treat $f_i$ as a list length.

For another way of looking at merge cost is

$E_{d_i} \cdot l_i$ where $d_i$ is no. of merges i th list is part of.

Before minimized $E_{d_i} \cdot l_i$.

Now want to minimize $E_{d_i} \cdot f_i$.

Method: Combine two symbols of minimum frequency $f_a, f_b$

$\frac{f_a + f_b}{f_a \cdot f_b}$ put new symbol of frequency

$f_a \cdot f_b$ break into the collection of symbols.

Theorem: Produces min-cost code.

Proof: Suppose not.

Diagram:

Reduces cost by

$(f_{a-b} - f_c) \cdot \text{height difference}$

Construction note: Maintain min heap of current $f$ values.

Each action: 2 deletions, one insertion.

Repeat $n$ times: $O(n \log n)$ time.

Initial $n$ costs: $O(n)$ time.
Optimality of the MST Algorithm

Suppose $l' < l$. Then swap of $l, l'$ subtrees gives an improved tree:

For cost reduction is $(l - l')$, height difference

For note that

$$\text{Total cost} = \sum_{\text{items } e} \left( \# \text{ of mergs } e \text{ is part of} \right)$$

Optimal tree:

- Each at deepest level - must have the lowest cost leaves.
- Also cut unchanged if swap nodes on the same level.

Thus can ensure the two lowest cost leaves are siblings in deepest level.
Claim \( \text{Opt Cost}\left(\ell_1, \ell_2, \ldots, \ell_n\right) = \ell_1 \ell_2 \times \text{Opt Cost}\left(\ell_1, \ell_2, \ell_3, \ldots, \ell_n\right) \).

Proof

**LHS \& RHS:**

Take \( \text{Opt tree for } \ell_1, \ell_2, \ell_3, \ldots, \ell_n \).
Add children \( \ell_1, \ell_2 \) to \( \ell_1, \ell_2 \).
\( \uparrow \) cost by \( \ell_1 + \ell_2 \).
Hence \( \text{LHS} \& \text{RHS} \).

**RHS \& LHS:**

Take \( \text{Opt tree for } \ell_1, \ell_2, \ell_3, \ldots, \ell_n \).
Remove \( \ell_1, \ell_2 \) and label parent with \( \ell_1, \ell_2 \).
\( \uparrow \) cost by \( \ell_1 + \ell_2 \).
Hence \( \text{RHS} \& \text{LHS} \).

It follows that the greedy alg is building the Opt tree bottom up.
Bin Packing

Items of \( n \) integer sizes \( s_1, s_2, \ldots, s_n \) want to pack into as few bins as possible of size \( b \).

**Greedy Solution:** Sort items in \( \mathcal{V} \) order. Process in this order. Put each item in fullest bin in which it will fit.

**Claim Not Optimal:**

**Sizes:** \( \frac{4}{19}, \frac{4}{19}, \frac{24}{19}, \frac{24}{19}, \frac{24}{19}, \frac{24}{19} \)

**Greedy:** 3 bins  

**Opt:** 2 bins:

\[
\begin{array}{|c|c|}
\hline
\frac{4}{19} & \frac{24}{19} \\
\frac{4}{19} & \frac{24}{19} \\
\hline
\end{array}
\quad
\begin{array}{|c|c|}
\hline
\frac{24}{19} & \frac{24}{19} \\
\frac{24}{19} & \frac{24}{19} \\
\hline
\end{array}
\]

**Lemma:** Suppose that \( \text{Opt} \) uses \( b \) bins.

Let the \( k \) smallest items have size \( s' = s_{n-b+1} \).

Then Greedy uses at most \( k + \left\lfloor \frac{k s'}{b} \right\rfloor \) bins.

Then Greedy uses at most \( 2k+1 \) bins.

**Proof:** There are at most \( k \) items of size \( \geq \frac{b}{2} \) (as would not fit in \( b \) bins). But Greedy uses \( 15 \) bins.

At most one bin is \( \frac{b}{2} \) full at any time.

Let \( \frac{x}{y} = \frac{\sum_{i=1}^{k} s_i}{k} \). Then \( \sum_{i=1}^{k} s_i > \frac{b-1}{2} b \).

But \( \sum_{i=1}^{k} s_i < kb \). Hence \( \frac{b-1}{2} \leq k \) or \( b \leq 2k-1 \).