

Honors Algorithms
G22.3520-001 Fall 2007

Lecture 9
Read: CLRS 16

Greedy Algorithms

Example

The Line Breaking Problem

- Given a sequence of words
- Form a paragraph, breaking lines as necessary
- Assumptions:
 - fixed width spacing
 - need at least one space between words
 - words have lengths l_1, \dots, l_n , and lines have length B , with each $l_i \leq B$
 - the l_i 's and B already include a “trailing space character”

A solution is a sequence (p_1, p_2, \dots, p_k) , meaning that p_i words go on the i th line, for $i = 1, \dots, k$

Different ways of measuring the “cost” of a solution

One simple cost metric: # of lines ($= k$)

A “greedy” strategy: choose p_1 maximal such that words $1, \dots, p_1$ fit on first line, then choose p_2 maximal such that words $p_1 + 1, \dots, p_1 + p_2$ fit on second line, and so on ...

Running time: $O(n)$

Claim

This greedy strategy minimizes the number of lines

Proof of claim. Prove by induction on k the statement:

- if the greedy algorithm finds an k -line solution, then there is no solution with $< k$ lines

$k = 1$: clear

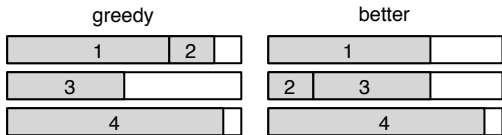
$k > 1$:

- Suppose greedy algorithm finds an k -line solution (p_1, \dots, p_k)
- Consider another solution $(q_1, \dots, q_{k'})$
- We must have $q_1 \leq p_1$, since p_1 was chosen as large as possible by the greedy algorithm

- Modify the solution $(q_1, \dots, q_{k'})$, moving $p_1 - q_1$ words from the second line to the first
- This yields a solution of length $k'' \leq k' - 1$ to the problem instance $(\ell_{p_1+1}, \dots, \ell_n)$
- However, (p_2, \dots, p_k) is a greedy solution to this problem instance
- By induction, $k - 1 \leq k'' \leq k' - 1$, and so $k \leq k'$ QED

Different cost functions may not allow a greedy algorithm

- “1 norm” — sum of extra spaces on all but the last line
 - greedy algorithm also works here
- “max norm” — maximum of extra spaces on all but last line
 - greedy algorithm *does not* work here



We can minimize the max-norm cost using dynamic programming

For $i = 1 \dots n$, define $C(i) =$ minimum max-norm cost to format l_i, \dots, l_n

For $i = 1 \dots n$ and $j = i \dots n$, let $S_{ij} := \sum_{t=i}^j l_t$

For $i = 1 \dots n$, we have

$$C(i) = \begin{cases} 0 & \text{if } S_{in} \leq B \\ \min_{\substack{i \leq j < n \\ S_{ij} \leq B}} \left\{ \max(B - S_{ij}, C(j+1)) \right\} & \text{o/w} \end{cases}$$

Subproblem graph:



Number of nodes = $O(n)$

Number of edges per node = $O(n)$

Running time = $O(n^2)$

Example: Huffman Encoding Problem

Let w_1, \dots, w_n be non-negative weights

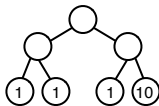
Let T be a binary tree, with each w_i labeling some leaf of of depth d_i

Define $\text{Cost}(T) := \sum_i w_i d_i$

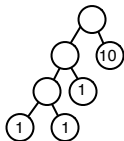
Problem: given w_1, \dots, w_n , find a minimal cost T

Without loss of generality, we may assume T is a *full* binary tree, i.e., each non-leaf has two children

Example:



Cost = 26



Cost = 10 + 2 + 3 + 3 = 18

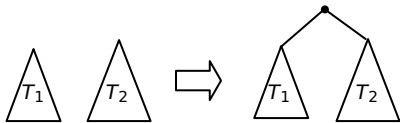
Application: optimal prefix-free binary encoding

- w_i represents probability of symbol σ_i
- path in tree represents a bit string encoding
- $\text{Cost}(T)$ is expected encoding length
- prefix-free property allows for unambiguous parsing of strings
- Example: $\Pr[A] = \Pr[B] = \Pr[C] = 1/13$, $\Pr[D] = 10/13$
Encoding: $A \Rightarrow 000$, $B \Rightarrow 001$, $C \Rightarrow 01$, $D \Rightarrow 1$

For a tree T , define its *weight* to be the sum of weights of its leaves

Greedy Algorithm:

- Start with a forest of n leaves
- Repeat $n - 1$ times:
 - Take two trees T_1, T_2 in the forest of least weight, and join them:



Implement using a heap. Running time: $O(n \log n)$

Theorem

This greedy algorithm produces a least-cost tree.

Lemma 1

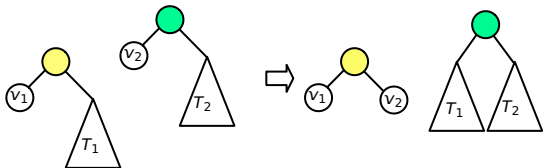
Let T be a full binary tree with weights $w(v)$ assigned to leaves v . Suppose v_1, v_2 are leaves of smallest weight. We can construct a new tree T' from T such that

1. the leaves of T' and T are the same,
2. v_1 and v_2 are siblings in T' , and
3. $\text{Cost}(T') \leq \text{Cost}(T)$.

Proof of Lemma 1. Assume v_1, v_2 not siblings in T

Let $d_i :=$ depth of v_i in T for $i = 1, 2$

Assume $d_1 \geq d_2$ and let $\Delta := d_1 - d_2$

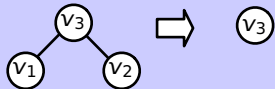


Moving v_2 down *increases* cost by $\Delta \cdot w(v_2)$

All leaves in T_1 have weight $\geq w(v_2)$ (because v_1, v_2 have least weight), and so moving T_1 up *decreases* cost by at least $\Delta \cdot w(v_2)$

Lemma 2

Let T be a full binary tree with weights $w(v)$ assigned to leaves v . Suppose v_1, v_2 are leaves that are siblings in T with parent v_3 , and that we create a new tree \tilde{T} by deleting v_1 and v_2 and set $w(v_3) := w(v_1) + w(v_2)$:



Then $\text{Cost}(\tilde{T}) = \text{Cost}(T) - w(v_1) - w(v_2)$.

Proof. Let $d = \text{depth of } v_3 \text{ in } T$

v_1 and v_2 contribute $(d + 1)(w(v_1) + w(v_2))$ to $\text{Cost}(T)$

v_3 contributes $d(w(v_1) + w(v_2))$ to $\text{Cost}(\tilde{T})$

Proof of Theorem.

Induction on n

If $n \leq 2$, clear; assume $n > 2$ and assume theorem holds for all values less than n

Let T be the tree produced by the greedy algorithm, and let X be any tree with same weights as T

Want to show: $\text{Cost}(T) \leq \text{Cost}(X)$

Consider the first step of the greedy algorithm, which joined two leaves v_1, v_2 of smallest weight

v_1 and v_2 are siblings in T

Apply Lemma 1 to X , obtaining a new tree X' in which v_1 and v_2 are siblings, and $\text{Cost}(X') \leq \text{Cost}(X)$

Apply Lemma 2 to both T and X' , removing v_1 and v_2 , obtaining trees \tilde{T} and \tilde{X}' such that

$$\text{Cost}(\tilde{T}) = \text{Cost}(T) - w(v_1) - w(v_2)$$

$$\text{Cost}(\tilde{X}') = \text{Cost}(X') - w(v_1) - w(v_2)$$

\tilde{T} is also a tree that would be produced the greedy algorithm, and so by induction, $\text{Cost}(\tilde{T}) \leq \text{Cost}(\tilde{X}')$

It follows that

$$\text{Cost}(T) \leq \text{Cost}(X') \leq \text{Cost}(X) \quad \text{QED}$$