COL 351 Lecture 19 2023/02/22

Topic : Huffman Coding
Optimum Spanning Trees

Announcement:

Quiz 2 Feb 28 19:00-20:00 LH 111

Syllabus: Upto Huffman Coding (inclusive)
and Tutorials 1-5.

Huffman Coding Problem

Input : Finite set $S = \{(a_1, p_1), \ldots (a_n, p_n)\}$
($\{a_1, \ldots, a_n\}$: set of "letters" / alphabet)
A +ve real number p_i for each a_i.

Output: A binary tree T with leaves $a_1, \ldots, a_n$
that minimizes

$$\sum_{i=1}^{n} p_i \cdot \text{depth}(a_i).$$

Claim: $\exists$ an opt tree in which the two least frequent letters are siblings.

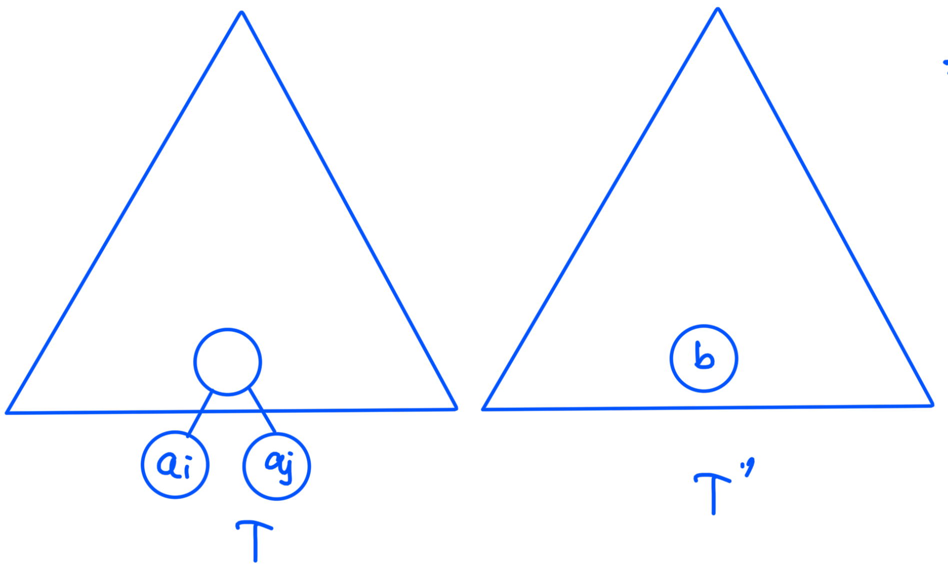
HuffmanCoding(S) $\quad (S = \{(a_1, p_1), \ldots, (a_n, p_n)\})$

0. (Base case) If $n = 1$, return the tree with a single node a_1.
1. Find the two least frequent letters, say a_i, a_j.
2. $S' \leftarrow (S \setminus \{(a_i, p_i), (a_j, p_j)\}) \cup \{(b, p_i + p_j)\}$, where b is a fresh letter not among $a_1 \ldots a_n$.
3. $T' \leftarrow$ HuffmanCoding(S').
4. Attach a_i, a_j to T' as children of b, to get tree T.
5. Return T.

Claim: Let T be any solution to S in which a_i, a_j are siblings. Let T' be the solution to S' formed by labelling the parent of a_i, a_j in T by b, and cutting off a_i, a_j. Then T is opt for S iff T' is opt for S'.

Proof: [If].

$$\begin{aligned}
&\text{cost}(T) - \text{cost}(T') \\
&= p_i\,\text{depth}_T(a_i) + p_j\,\text{depth}_T(a_j) \\
&\quad - (p_i + p_j)\,\text{depth}_{T'}(b) \\
&= p_i + p_j
\end{aligned}$$



Suppose T is not opt for S. Let O be an opt tree for S, in which a_i, a_j are siblings. Remove a_i, a_j from O, call their parent b, call the resulting tree O'.

Then $\text{cost}(O) - \text{cost}(O') = p_i + p_j$

$\text{cost}(O) < \text{cost}(T) \quad \therefore \quad \text{cost}(O') < \text{cost}(T') \longrightarrow$ contradiction to optimality of T'.

[Only if] : Left as exercise.

Running Time of Huffman Coding algorithm:

$O(n^2)$ for "obvious" implementation

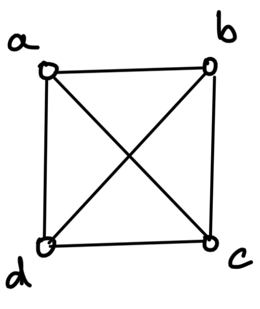
$O(n \log n)$ using a min-heap.

Optimum Spanning Trees.

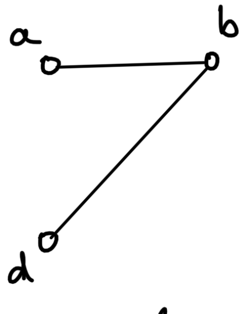
Input: Connected undirected graph $G = (V, E)$

Weight (possibly -ve) w_e for each edge $e \in E$.

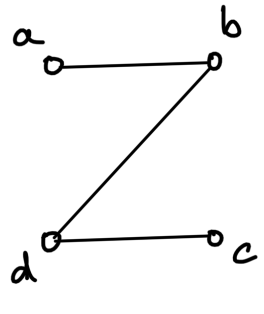
Output: A spanning tree of G with min weight.



G



tree but not a spanning tree of G



spanning tree of G

T : spanning tree of G.

$e \in T$, $e' \notin T$

Question: When is $T' = (T \setminus \{e\}) \cup \{e'\}$ a spanning tree?

T' is a spanning tree iff endpoints of e' are in different connected components of $T \setminus \{e\}$

(*) T' is a spanning tree iff e belongs to the unique cycle in $T \cup \{e'\}$.

If, additionally, $w_{e'} < w_e$, then T' is better than T.

Kruskal's Algorithm:

1. $T \leftarrow \emptyset$.
2. Sort edges in nondecreasing order of weight, say $e_1, e_2, \ldots, e_m$ $(w_{e_1} \leq w_{e_2} \leq \ldots \leq w_{e_m})$.
3. For $i = 1$ to m: (Invariant: T is acyclic)
   - If the endpoints of e_i are in different components of T: (use BFT / DFT)

     then $T \leftarrow T \cup \{e_i\}$.

   (Else discard e_i).