EE360C ALGORITHMS - FALL 2018 - 11-12:30 - THU OCT 18

Huffman Alg: Given $\overbrace{\text{symbols}}^{\text{set } C}$ and their $\overbrace{\text{frequencies}}^{f(x)}$, generate $\underbrace{\text{optimal}}_{\text{minimizes } ABL(T)}$ $\underbrace{\text{prefix code}}_{\text{tree } T}$.

freq. of x : $\dfrac{\#\text{of occurances in file}}{\text{length of file}}$

$$ABL(T) = \sum_{x \in C} f(x) \cdot d_T(x)$$

$ABL(T)$ ↑ avg bits per letter

$d_T(x)$ ↑ depth of x in T = size of the encoding of x

Greedy: Start from lower frequency symbols at the bottom of the tree.

Huff(C):

If $C = \{x, y\}$, return 

Let x, y be two lowest frequency symbols in C.

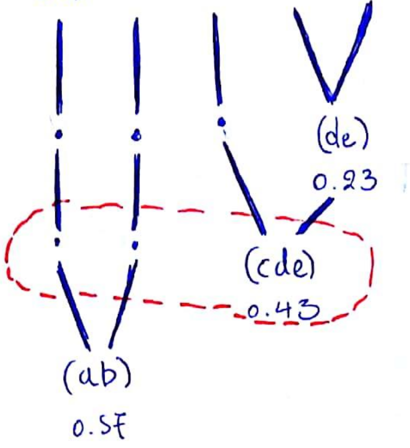
Let $C' = \underbrace{(C \setminus \{x,y\}) \cup \{z\}}_{\text{remove } x,y \text{ and add } z}$ where $f(z) = f(x) + f(y)$

Recursively compute $T = \text{Huff}(C')$

Modify T by adding x and y as children of z (which ceases to be a leaf)

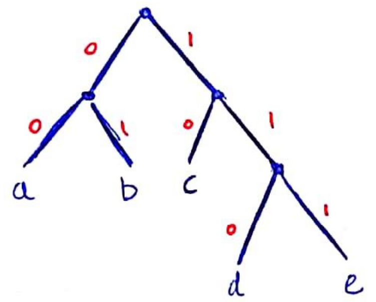
Ex: $C = \{a, b, c, d, e\}$

freq: 0.32 0.25 0.2 0.18 0.05



At this point, we have:
a, with freq. 0.32
b, — " — 0.25
(cde), — " — 0.43
The two lowest are a and b
so we combine these next.

Unroll recursion to find T:



Running Time:

- n recursive calls
- with heap: each recursive call does $O(\log n)$ work.

Overall: $O(n \log n)$

Divide & Conquer

unordered array

merge-sort (A):

$$O(n)\begin{cases}\text{if } |A| = 1, \text{ return } A\\ \text{let } L = \text{left half of } A\\ \text{let } R = \text{right half of } A\end{cases}$$

$T(n/2)$ let LS = merge-sort (L)

$T(n/2)$ let RS = merge-sort (R)

return (merge (LS, RS))

given 2 sorted lists, merge into 1 sorted list. needs $O(n)$ time.

For simplicity assume $|A|$ is a power of 2.

If $T(n)$ = running time of merge sort on $|A| = n$.

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

Next class we will learn how to solve recursions like this one!