COL 106 Lecture 13

Topic: Heap Operations

Announcement: Assignment 2 released.
Deadline: Sep 18, 23:00

Recap:

- HeapUp, HeapDown
- BuildHeap in $\Theta(n \log n)$ time.

Today:

- Faster BuildHeap
- Huffman Coding – intro

Build Heap (l) l: list of keys

Obvious algorithm: Enqueue each key in l, one by one

Running time : $O(n \log n)$

Claim: Running time is $\Omega(n \log n)$ (in the worst case).

$n, n-1, n-2, \ldots, n/2, \ldots, > 1$

↓

Enqueueing these takes $\Omega(\log n)$ time each.

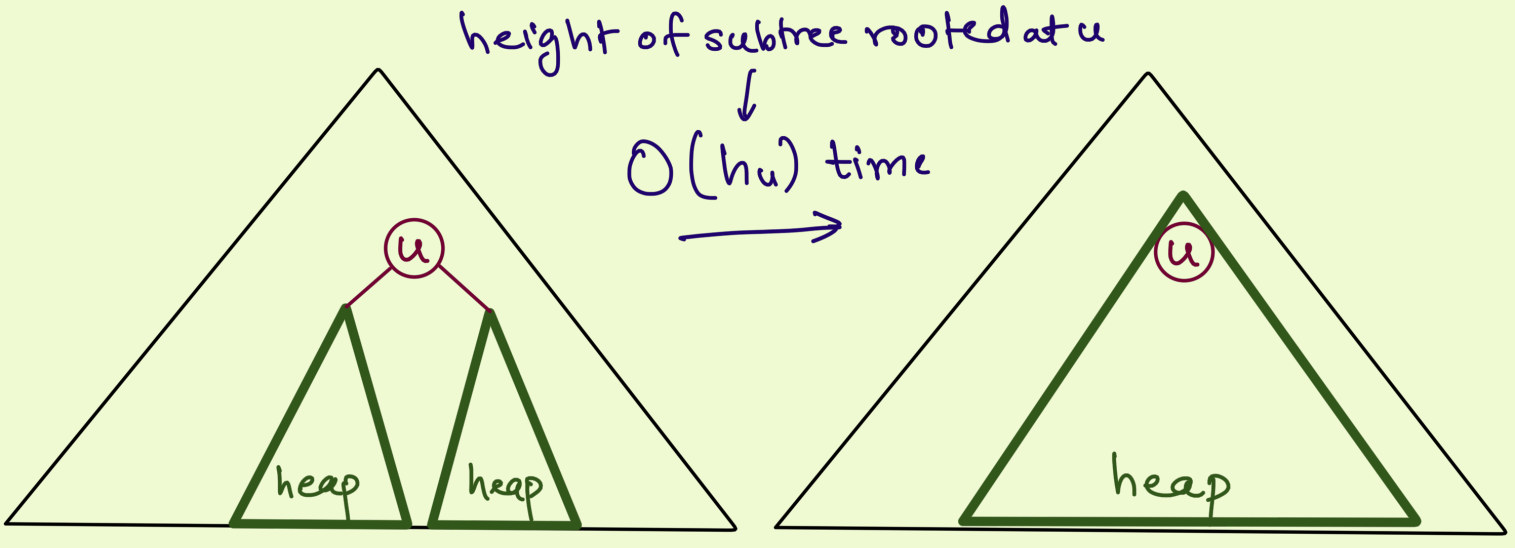
Recall HeapDown (u)

Preconditions :

① Almost complete binary tree

② Left and right subtrees of u are both heaps.

Postcondition : Subtree rooted at u is a heap.



Fast BuildHeap (l)

For $u = len(l) - 1$ to 0 :

HeapDown (u).

Loop invariant

← $\forall v > u$, subtree rooted at v is a heap

← $\forall v \geq u$, subtree rooted at v is a heap.

Correctness → follows from loop invariant and pre and post conditions of HeapDown.

Running Time: $O(\sum_u h_u)$

Time Complexity of Fast BuildHeap

Claim: Fast BuildHeap runs in $O(n)$ time.

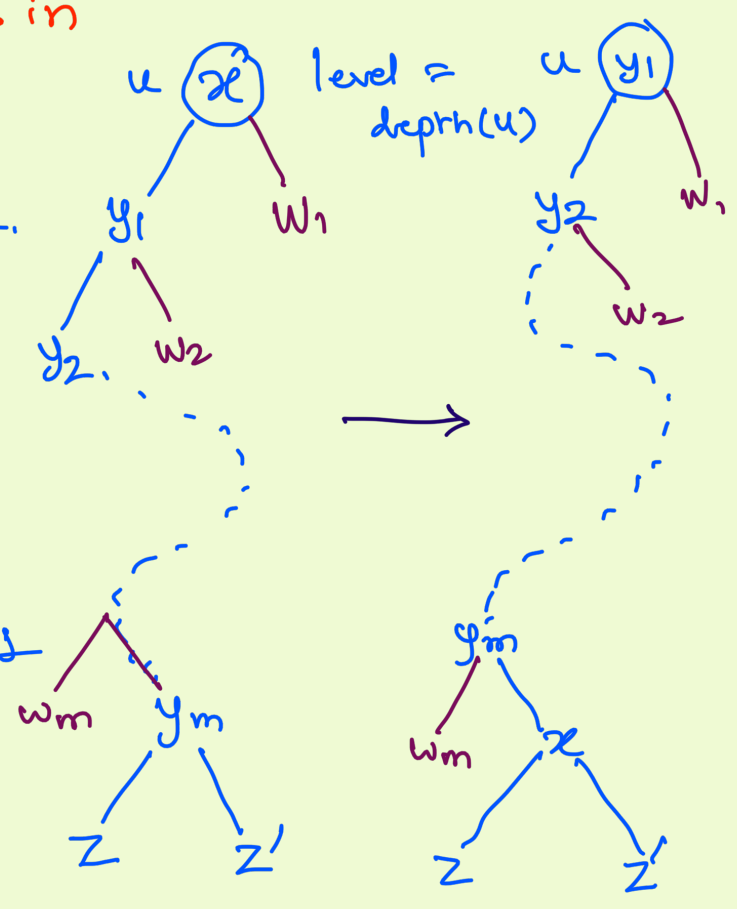
Proof: Consider layers l, $l+1$.

exchanges by HeapDown (u) between layers l and $l+1$

≤ 1

$= 0$ if depth (u) $> l$

$\} < 2^{l+1}$



exchanges between levels l and $l+1$ made by Build Heap $\leq$ # nodes in levels $0, \ldots, l$.

$$= 1 + 2 + \cdots + 2^l < 2^{l+1}$$

Total # exchanges made by Build Heap

$$< \sum_{l=0}^{h-2} 2^{l+1} < 2^h$$

But $h = \lceil \log_2(n+1) \rceil$

$\therefore$ Total # exchanges $< 2^h = O(n)$. ∎

Heap Operations : Summary

- Heap Up (u) — $O(\text{depth}(u))$
  - → Enqueue — $O(\log n)$
- Heap Down — $O(h_u) = O(\log n - \text{depth}(u))$
  - → Extract Min — $O(\log n)$
  - → Change Key — $O(\log n)$
  - → Build Heap — $O(n)$

Find Min
Size
Is Empty
} — $O(1)$

Huffman Coding Problem

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

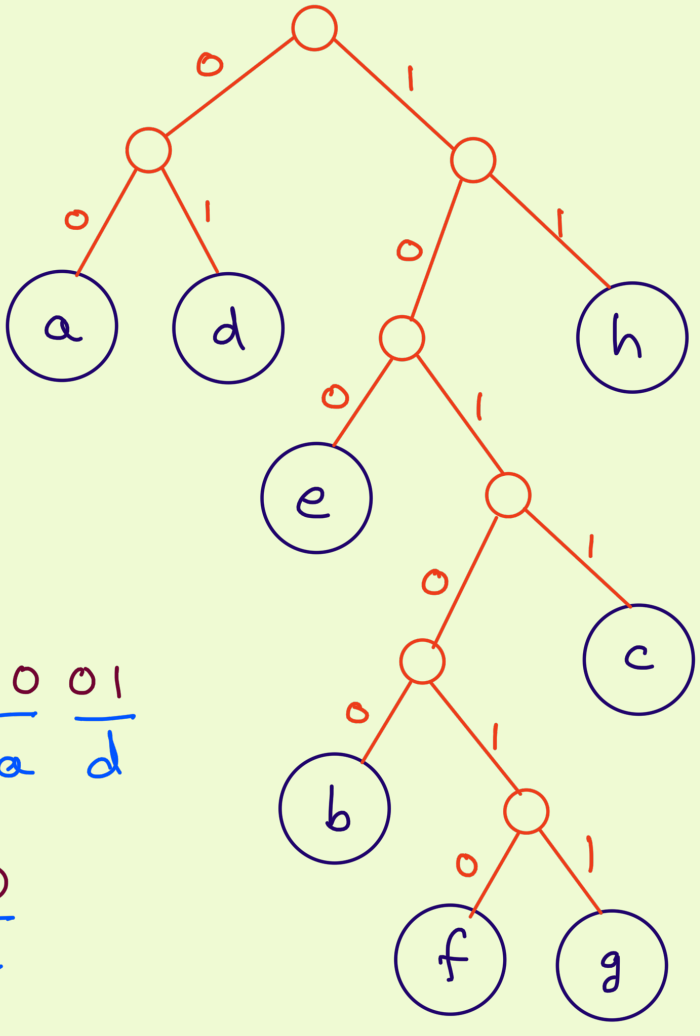
Output: A binary tree T with leaves $a_1, \ldots, a_n$
that minimizes

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

Motivation

$S = \{a, b, c, d, e, f, g, h\}$

a — 00	e — 100
b — 10100	f — 101010
c — 1011	g — 101011
d — 01	h — 11



01 00 01 11 00 01 00 10100 00 01
d a d h a d d b a d

11 100 00 01 00 1011 11 100
h e a d a c h e

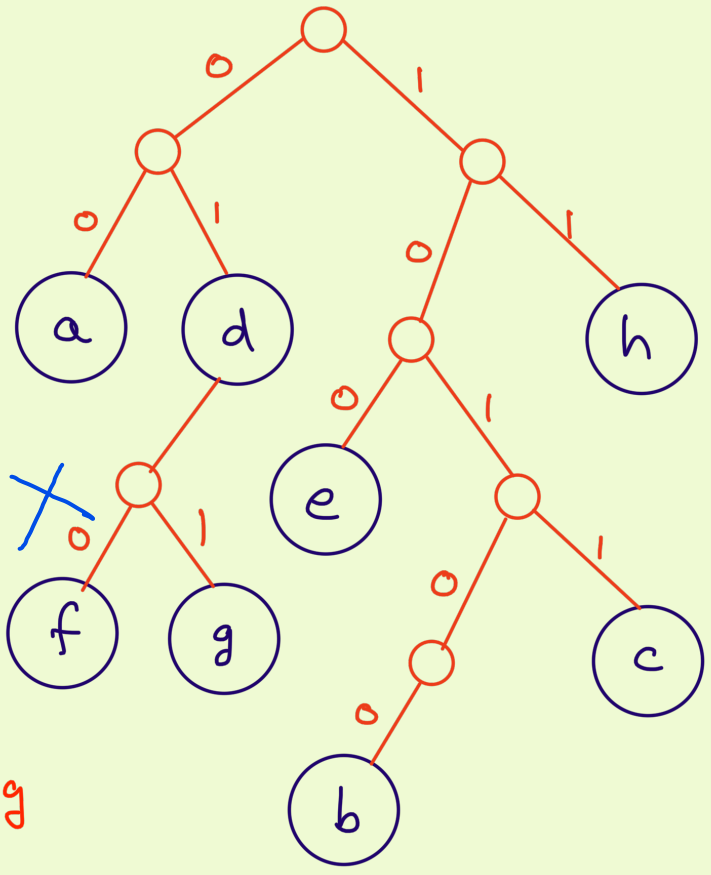
Prefix-Free Codes

0101 < g, dd

d — 01

g — 0101

f — 0100



Need: ~~∃~~ distinct letters a_i, a_j such that the bitstring representing a_i is a prefix of the bitstring representing a_j.

Objective function

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

p_i: prob. that a letter is a_i

depth(a_i): length of bit string encoding a_i

Expected # bits per letter

Claim: If $p_i < p_j$ then depth $(a_i) \geq$ depth (a_j) in the optimum binary tree.

Proof: By contradiction. Suppose $p_i < p_j$ and depth (a_i) < depth (a_j) in the optimum binary tree, say T.

Exchange the locations of a_i and a_j. Call the resulting tree T'.

Then T' is better than T → Contradiction. ∎