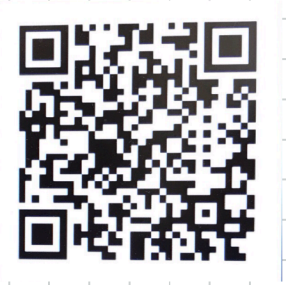
CS 170

Lecture 8

- use iClicker

Sanjam Garg.

join.iclicker.com/RGWR



Greedy Algorithms

Goal: Optimize some task involving multiple decisions/steps.

Greedy: We take whatever seems optimal right now; hopefully the final solution is also optimal. → elegant algorithm design.

When? local to global connection that ensures overall optimal solution.

Task Scheduling Problem

Input: n jobs with start & end times
$[s_1, t_1] \dots [s_n, t_n]$

Task: Schedule as many non-overlapping tasks as possible

Claim: Greedy non-overlapping first time finish is optimal!

Proof:
Greedy = $[s_1, t_1] \dots [s_r, t_r]$
Optimal = $[s_1', t_1'] \dots [s_l', t_l']$

Claim I: $r \le l$; follows from the fact that l is max. number of jobs that can be scheduled.

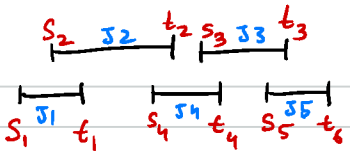
Claim II: $\forall\, i = 0 \dots r$ $H_i = [s_1, t_1] \dots [s_i, t_i] [s_{i+1}', t_{i+1}'] \dots [s_l', t_l']$ is optimal.

- $H_0 = [s_1', t_1'] \dots [s_l', t_l']$ → done ◻
- $i \to i+1$ H_i → is optimal
  $H_{i+1} = [s_1, t_1] \dots [s_i, t_i] [s_{i+1}', t_{i+1}'] \dots [s_l', t_l']$
  prove that this is optimal ←
  $H_{i+1} = [s_1, t_1] \dots [s_i, t_i] [s_{i+1}, t_{i+1}] [s_{i+2}', t_{i+2}'] \dots [s_l', t_l']$
  $t_i < s_{i+1} < t_{i+1} \le t_{i+1}' < s_{i+2}'$

Claim III: $r = l$
$H_r = [s_1, t_1] \dots [s_r, t_r] [s_{r+1}', t_{r+1}'] \dots [s_l', t_l']$

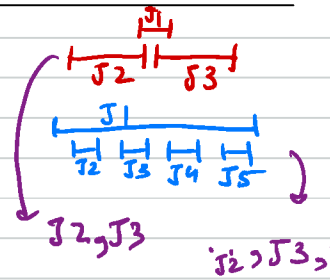
greedy would stop only if $r = l$ ◻

Example:

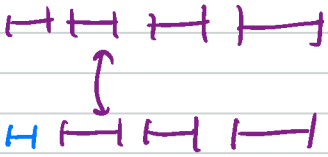


optimal → J1, J4, J5

Strategies	Good idea?
shortest first	
first start time	
first finish time	



optimal



Algorithm: Sort jobs by finish time so $t_1 \le t_2 \ldots\ldots t_n$ → $O(n \log n)$

set of jobs that are scheduled → $A = \phi$, $t^* = -\infty$

for $j = 1$ to n

if $t^* < s_j$ → $O(n)$

$A = A \cup \{[s_j, t_j]\}$

$t^* = t_j$

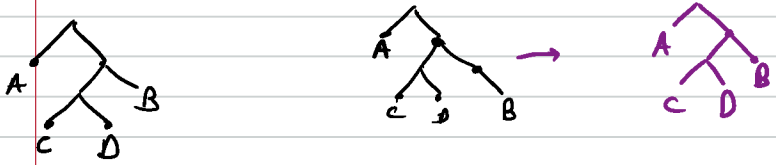
Return A

Compression (Huffman Codes)

Input: Encode some text with T letters from an alphabet Γ with n letters and frequencies $\{f_i : i \in \Gamma\}$

$$\text{cost}(T) = \sum_i f_i \times \#\text{bits needed to represent } i \in T$$

Prefix-freeness: No codeword can be a prefix of another.



↑ full binary tree

Def #1 node has 0 or 2 children

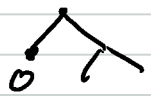
Example: $\Gamma = \{A, B, C, D\}$ $T = 100$

Γ	Frequency	Code1	Code2	Code3
A	80	00	0	0
B	10	10	1	11
C	5	01	10	100
D	5	11	11	101
Cost		200	110	130

In code2 how do we decode 10?

is it ① BA or ② C

Heaviest First

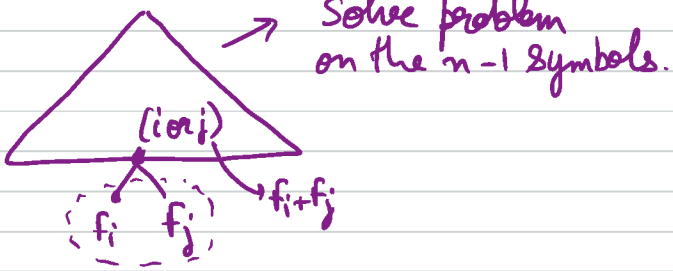


Lightest First

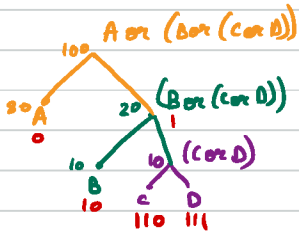
Algorithmic intuition.

List symbols in increasing order of frequency $f_1 \longrightarrow \dots \longrightarrow f_n$

- Find lowest two frequencies f_i, f_j
- Remove f_i, f_j & add symbol (i or j) with frequency $f_i + f_j$
- iterate



Example

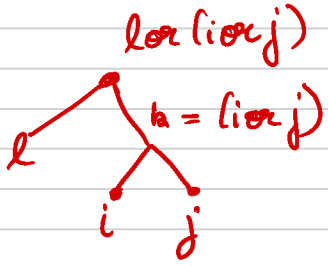


Algorithm:

Huffman (f)

Input: $f[1 \dots n]$ the frequencies.
Output: encoding tree with n leaves.

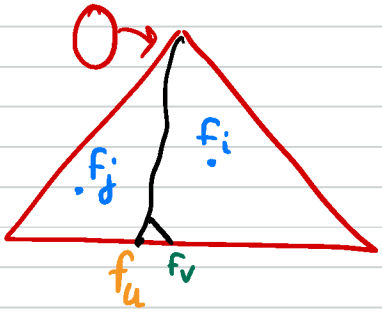
```
H = priority queue
For i = 1 ... n : Insert (i, f[i])
For k = n+1 .... 2n-1
        i = Delmin(H)    j = Delmin(H)
        create node k with children i, j
        f[k] = f[i] + f[j]
        Insert ( k, f[k])
```



$\rightarrow O(n \log n)$

Claim: $\exists$ an optimal tree T where f_i, f_j (the lowest two frequencies) are the deepest siblings.

Proof:



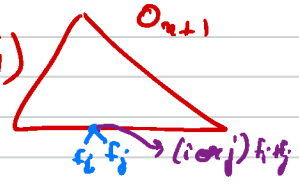
- Let O be an optimal tree
- Let f_u be the deepest node in O
- f_u must have a sibling f_v
- f_i, f_j are somewhere
- Swap f_i with f_u, f_j with f_v get a new tree M
- $f_i \& f_j \leq f_u \& f_v$ $\quad f_i \leq f_u \quad f_j \leq f_v$
- $cost(M) \leq cost(O)$
- $cost(M) = cost(O)$ □

Lemma: Huffman finds the optimal tree

$n=2$:  $\rightarrow O(T) = f_i + f_j$

$n \rightarrow n+1$

- By the claim that we saw, $\exists$ an optimal solution O_{n+1} with $f_i \& f_j$ as the deepest nodes.
- $cost(O_{n+1}) = cost(O_n') + f_i + f_j$ — (1)



* Huffman: H_{n+1} if places $f_i \& f_j$ as siblings

H_n on other symbols & (i or j) with $f_i + f_j$

- $cost(H_n) = cost(O_n)$ — (2)
- $cost(H_{n+1}) = cost(H_n) + f_i + f_j$ — (3)

$\overset{(2)}{=} cost(O_n) + f_i + f_j$

$\overset{(1)}{=} cost(O_{n+1})$