Lecture 8

CS 170

Sanjam Garg.

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:
$$\text{Greedy} = [s_1, t_1] \; \dots \; [s_r, t_r]$$
$$\text{Optimal} = [s_1', t_1'] \; \dots \; [s_\ell', t_\ell']$$

H H H...H

(i) $r \le \ell$

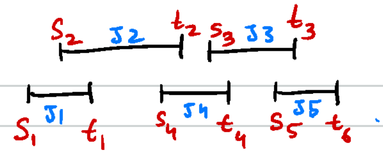
(ii) $\forall\, i = 0 \dots r \quad H_i = [s_1, t_1] \dots [s_i, t_i]\,[s_{i+1}', t_{i+1}'] \dots [s_\ell', t_\ell']$

$H_0 \equiv \text{Optimal}$

$H_r \equiv \text{Greedy} \,|\, \boxed{\text{optimal}}$

are optimal.

Example:

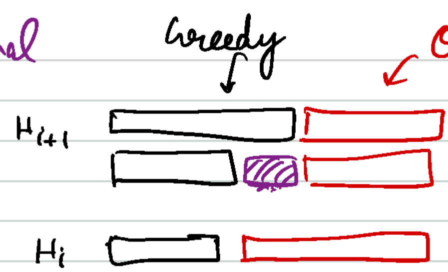


J2 & J3, J1 & J4, J1 & J4 & J5

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

Base: H_0 is optimal

IS: $H_i \to H_{i+1}$



→ (i) non-overlapping.
(ii) has ℓ jobs.
] ⇒ H_{i+1} is optimal

$$t_i < s_{i+1} < t_{i+1} \le t_{i+1}' < s_{i+2}'$$

(iii) $[s_1, t_1] \; \dots \; [s_r, t_r]\,[s_{r+1}', t_{r+1}'] \; \dots \; [s_\ell', t_\ell']$

Greedy could go on!

$\Rightarrow \underline{r = \ell}$ $\square$

Algorithm:

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

scheduled jobs ← $A = \phi$, $t^* = -\infty$ → finish time of the last scheduled job

for $j = 1$ to n

  if $t^* < s_j$

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

    $t^* = t_j$ → $O(n)$

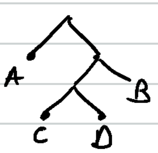
Return A

Compression (Huffman Codes)

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

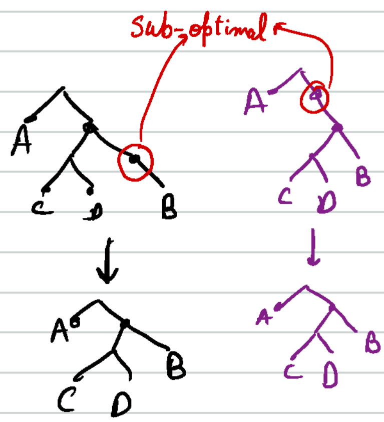
$$\text{Cost}(T) = \sum_i f_i \times \#|\text{bits needed to represent } i \in T|$$

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



↑ full binary tree

Def: Each node has 0 or 2 children

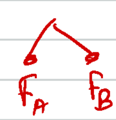


Example: $\Gamma = \{A, B, C, D\}$ $T = 100$

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 ⓐ BA or ⓑ C

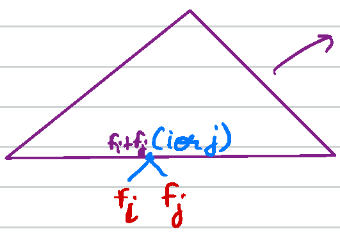
Strategy	
Heaviest First	 ✗
Lightest First	 ✓

Algorithmic intuition.

List symbols in increasing order of frequency $f_1 \; - - - \; f_n$

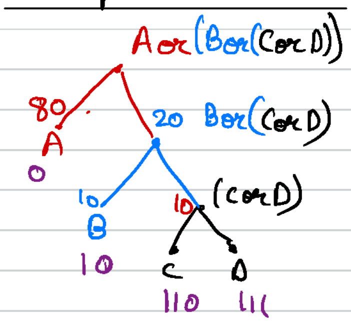
- Find lowest two frequencies f_i, f_j.
- Remove f_i, f_j & add symbol (i or j) with frequency $f_i + f_j$
- iterate

Pictorially



solve problem on smaller instance of $n-1$ symbols.

Example execution



Algorithm:

Huffman (f)

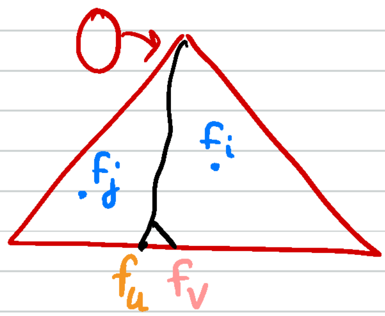
```
Input:  f[1 ... n]   the frequencies.
Output: encoding tree with n leaves.

    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])
```

Running time

$O(n \log n)$

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

Proof:



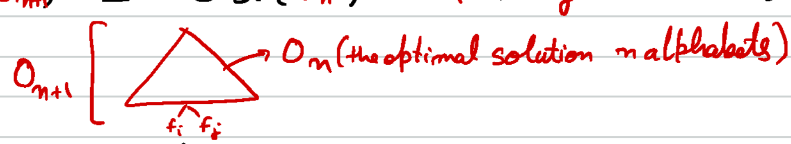
- Let f_u be the deepest node in an optimal tree O
- Since the tree is a full tree f_u has a sibling f_v
- The tree has O has f_i and f_j somewhere
- Swap f_i with f_u & f_j with f_v to get a new tree M
- Observe that $cost(M) \le cost(O)$ as $f_i \le f_u$ & $f_j \le f_v$
- However, O is optimal & $cost(O)$ is the smallest
- Thus, $cost(M) = cost(O)$ $\square$

Lemma: Huffman finds the optimal tree

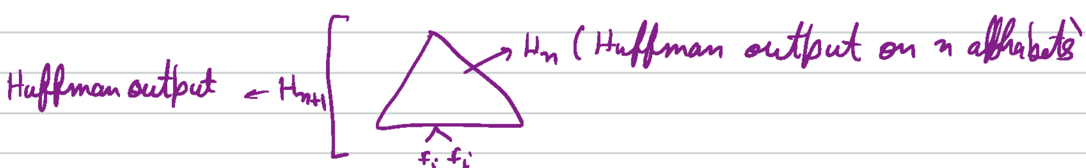
$n = 2$:

$n \to n+1$:

- By claim on last slide, $\exists$ an optimal solution O_{n+1} with f_i, f_j as the deepest leaves.
- $cost(O_{n+1}) = cost(O_n) + f_i + f_j$ ——— (1)



- Huffman proceeds by also placing f_i, f_j as leaves.



- $cost(H_{n+1}) = cost(H_n) + f_i + f_j$ ——— (2)
- By IH, $cost(H_n) = cost(O_n)$ ——— (3)
- By (1), (2), and (3) $cost(H_{n+1}) = cost(O_{n+1})$