

# CS3383 Unit 2: Greedy

David Bremner

February 15, 2024



# Outline

## Greedy

Properties of (optimal) Huffman trees

Huffman algorithm

# Lecture background

- ▶ CLRS4 §15.3
- ▶ `https://jeffe.cs.illinois.edu/teaching/algorithms/book/04-greedy.pdf`
  - ▶ Huffman Coding is covered in §4.4
- ▶ DPV 5.2

# Lecture background

- ▶ CLRS4 §15.3
- ▶ <https://jeffe.cs.illinois.edu/teaching/algorithms/book/04-greedy.pdf>
  - ▶ Huffman Coding is covered in §4.4
- ▶ DPV 5.2

## From Data Structures

- ▶ heaps
- ▶ priority queues

# Prefix codes

Symbol	Freq	Codeword
A	70	0
B	3	001
C	20	01
D	37	11

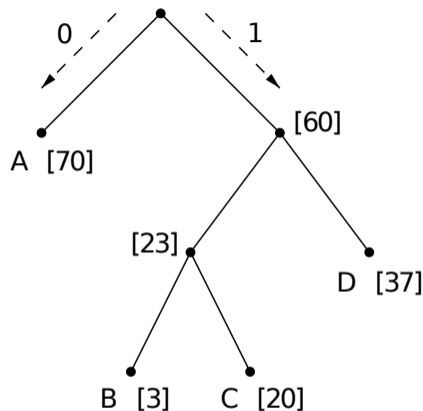
- ▶ avoiding ambiguous bitstreams: what is 001?

# Prefix codes

Symbol	Freq	Codeword
A	70	0
B	3	100
C	20	101
D	37	11

# Huffman coding

Symbol	Codeword
A	0
B	100
C	101
D	11



(Avg cost)

$$\text{cost}(T) = \sum_{i=1}^n f_i \text{depth}_i$$

# Huffman trees are full

## Lemma (Full Trees)

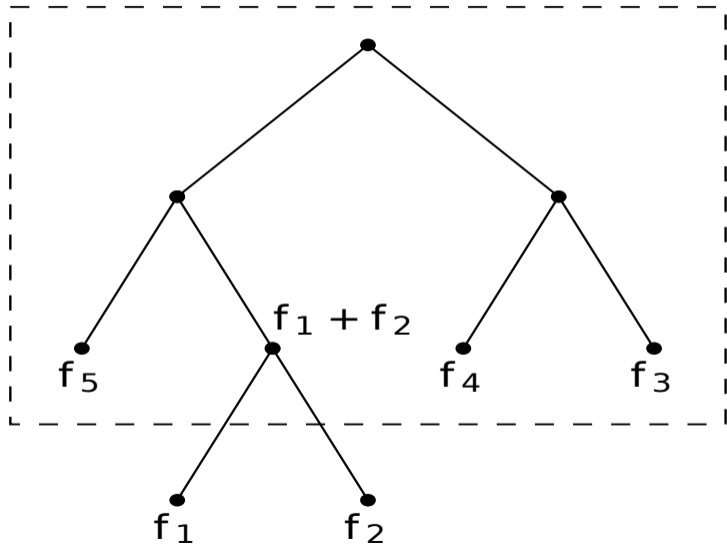
*In an optimal Huffman tree every node has zero or two children.*

## Proof.

If not, consider a node with one child. The corresponding bit in the code can be deleted without changing the property of being a prefix code. □



# Lightest leaves are deepest





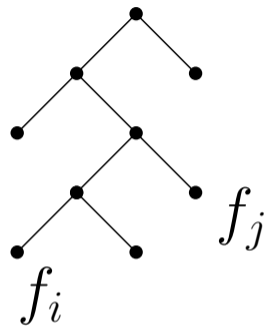
# The two lightest leaves are deepest

## Lemma (Lightest siblings)

*There exists an optimal tree where the lightest leaves are siblings on the deepest level.*

## setup for contradiction

Let  $T$  be an optimal Huffman tree with  $\text{cost}(T) = \sum_k f_k d_k$ . Let  $f_i$  be a leaf on the deepest level. Suppose some other leaf  $f_j$  exists with  $f_j < f_i$  and  $d_j < d_i$ .



# The two lightest leaves are deepest

## Lemma (Lightest siblings)

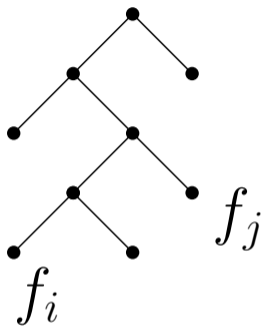
*There exists an optimal tree where the lightest leaves are siblings on the deepest level.*

Let  $T'$  be the tree with  $f_i$  and  $f_j$  swapped. The cost of  $T'$  is

$$\text{cost}(T') = \left( \sum_k f_k d_k \right) - f_j d_j - f_i d_i + f_j d_i + f_i d_j$$

(residual)

$$= \text{cost}(T) - [(d_j - d_i) \times (f_j - f_i)]$$





# Huffman demo

```
def huffman(f):
    tree=[]; H=[]; n = len(f)
    for i in range(0,n):
        heappush(H,(f[i],i))
        tree.append((f[i],None,None))
    for k in range(n,2*n-1):
        (f1,index1) = heappop(H)
        (f2,index2) = heappop(H)
        f3 = f1 + f2
        heappush(H,(f3,k))
        tree.append((f3,index1,index2))
    return tree
```

```
214
  84
    37
      47
        23
          3
            20
              24
                130
                  60
                    70
```

# Huffman example

Symbol	Freq	Codeword
A	70	0
B	3	100
C	20	101
D	37	11

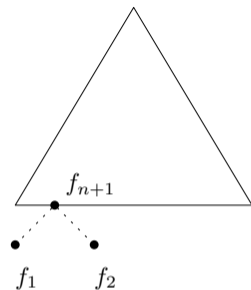
► first loop gives us

```
H = [(3, 0), (20, 1), (37, 2), (70, 3)]  
T = [(3, None, None), (20, None, None),  
      (37, None, None), (70, None, None)]
```

# Correctness of Huffman algorithm

## Theorem (Greedy Huffman Algorithm)

*huffman produces an optimal binary prefix code.*





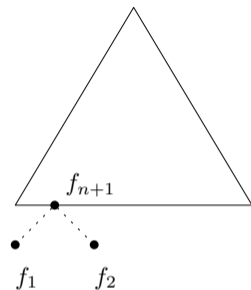
# Correctness of Huffman algorithm

## Theorem (Greedy Huffman Algorithm)

*huffman produces an optimal binary prefix code.*

## base case

If we have 1 or 2 symbols, any one bit code is optimal.



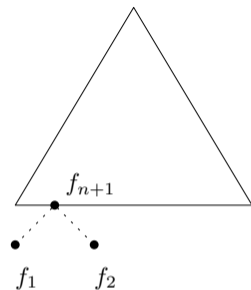
# Correctness of Huffman algorithm

## Theorem (Greedy Huffman Algorithm)

*huffman produces an optimal binary prefix code.*

## induction hypothesis

For all  $k < n$ , `huffman`( $[f_1 \dots f_k]$ ) produces an optimal Huffman tree.



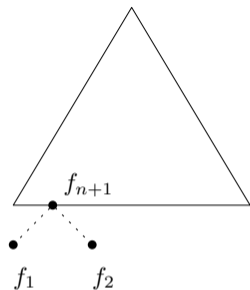
# Correctness of Huffman algorithm

## Theorem (Greedy Huffman Algorithm)

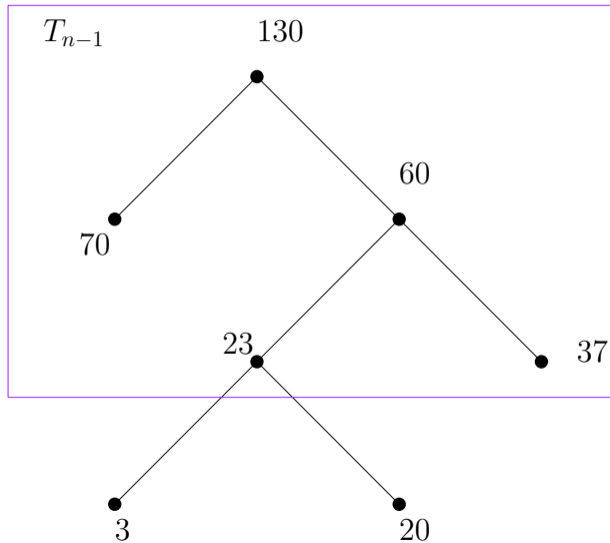
*huffman produces an optimal binary prefix code.*

## induction setup

Let  $f_1 \leq f_2 \leq \dots f_n$  be the original input. From *lightest siblings*, there exists some optimal  $T_n$  with  $f_1$  and  $f_2$  as deepest siblings. Let  $f_{n+1} = f_1 + f_2$ . Let  $T_{n-1} = \text{huffman}(f_3 \dots f_{n+1})$ .



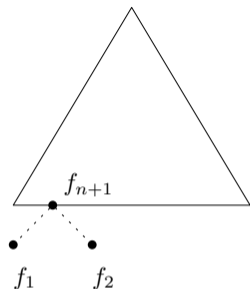
# Induction example



# Induction details

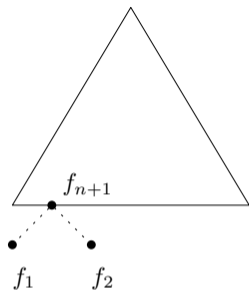
By induction,  $T_{n-1}$  is optimal. Now put  $f_1$  and  $f_2$  back as children of  $f_{n+1}$ , producing  $T_n$ . Let  $d_i$  denote the height of  $f_i$  in  $T_{n-1}$  (and/or  $T_n$ )

$$\begin{aligned}\text{cost}(T_n) &= \sum_{k=1}^n f_k d_k \\ &= f_1 d_1 + f_2 d_2 + \sum_{j=3}^{n+1} f_j d_j - f_{n+1} d_{n+1}\end{aligned}$$



# Induction details

$$\begin{aligned}\text{cost}(T_n) &= f_1 d_1 + f_2 d_2 + \sum_{k=3}^{n+1} f_k d_k - f_{n+1} d_{n+1} \\ &= \text{cost}(T_{n-1}) + f_1 d_1 + f_2 d_2 - f_{n+1} d_{n+1} \\ &= \text{cost}(T_{n-1}) + (f_1 + f_2) d_1 - f_{n+1} (d_1 - 1) \\ &= \text{cost}(T_{n-1}) + f_1 + f_2.\end{aligned}$$



# Induction details

Suppose  $\exists$  cheaper  $T'_n$  for  $f_1 \dots f_n$ . Removing  $f_1$  and  $f_2$  yields a tree  $T'_{n-1}$  for  $f_3 \dots f_{n+1}$ .

$$\text{cost}(T'_n) = \text{cost}(T'_{n-1}) + f_1 + f_2$$

$$\text{cost}(T'_{n-1}) + (f_1 + f_2) < \text{cost}(T_{n-1}) + (f_1 + f_2)$$

(contradiction)

$$\text{cost}(T'_{n-1}) < \text{cost}(T_{n-1})$$

