Huffman encoding

Encoding

- Consider problem of designing a binary code where each symbol is represented by a unique binary string.
- Suppose that the alphabet consists of four symbols (A, B, C, D).
- Fixed-length code requires 2 bits per symbol (00, 01, 10, 11).
- Given the frequencies for each symbol, using *variable*-length code can compress data considerably by giving frequent symbols short codewords and infrequent symbols long codewords.

Symbol	Frequency
A	70 million
B	3 million
C	20 million
D	37 million

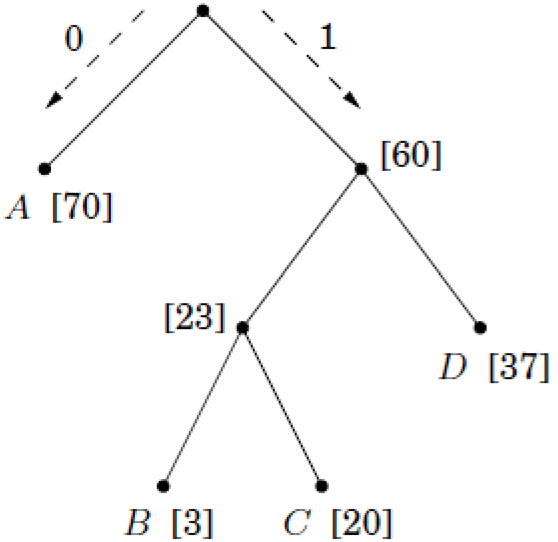
Symbol	Codeword
A	0
B	100
C	101
D	11

Prefix-free encoding

- If codewords are {0, 01, 11, 001}, the decoding of strings like 001 is ambiguous.
- *Prefix-free* encoding - no codeword can be a prefix of another codeword.
- Decoding process can be represented with a binary tree whose leaves are the given characters.
- An optimal code is represented by a *full* binary tree.

Symbol	Frequency
A	70 million
B	3 million
C	20 million
D	37 million

Symbol	Codeword
A	0
B	100
C	101
D	11

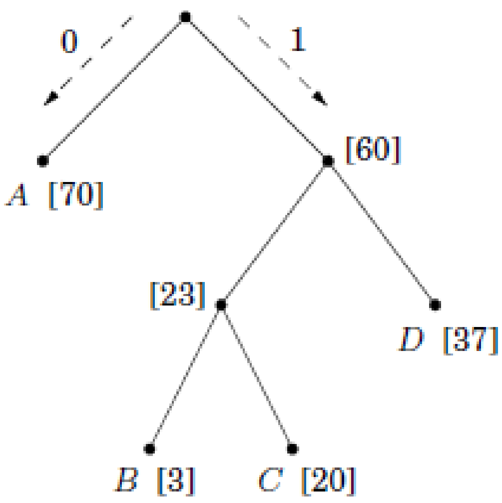


Cost of tree

- How do we find the optimal coding tree, given the frequencies $f_1, f_2, \ldots, f_n$ of n symbols?
- (1) We want a tree which minimizes the overall length of the encoding,

Symbol	Frequency
A	70 million
B	3 million
C	20 million
D	37 million

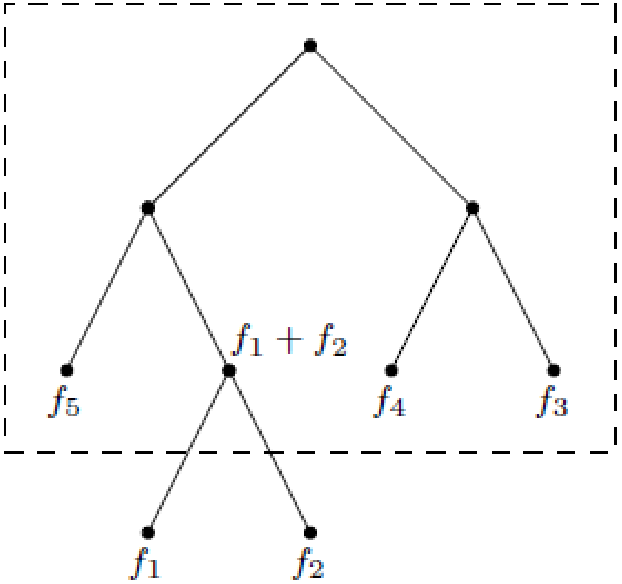
$$\text{cost of tree} = \sum_{i=1}^{n} f_i \cdot (\text{depth of } i\text{th symbol in tree})$$



- (The number of bits required for a symbol is its depth in the tree.)
- Define the frequency of an *internal* node as the sum of the frequencies of its descendant leaves - the number of times the internal node is visited during decoding.
- (2) So the total number of bits = cost of tree = the sum of the frequencies of all leaves and internal nodes except the root.

Greedy algorithm

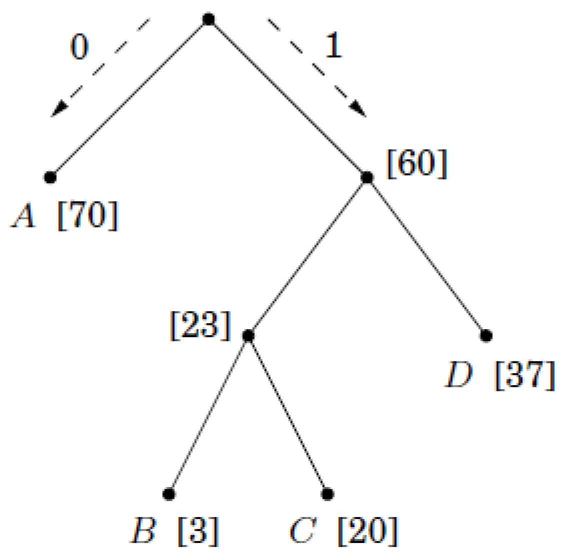
- By (1), the two symbols with the smallest frequencies must be at the bottom of the optimal tree, as children of the lowest internal node.
- (This internal node has two children since the tree is *full).*
- Otherwise, swapping these two symbols with whatever is lowest in the tree would improve the encoding.
- Assume f_1 and f_2 are the two smallest frequencies.
- By (2), any tree in which f_1 ,f_2 are sibling-leaves has cost $f_1 + f_2$ plus the cost for a tree with n-1 leaves of frequencies $(f_1 + f_2)$, $f_3, f_4, \dots f_n$



```
procedure Huffman(f)
Input:  An array f[1···n] of frequencies
Output:  An encoding tree with n leaves

let H be a priority queue of integers, ordered by f
for i = 1 to n:  insert(H, i)
for k = n + 1 to 2n − 1:
  i = deletemin(H),  j = deletemin(H)
  create a node numbered k with children i, j
  f[k] = f[i] + f[j]
  insert(H, k)
```

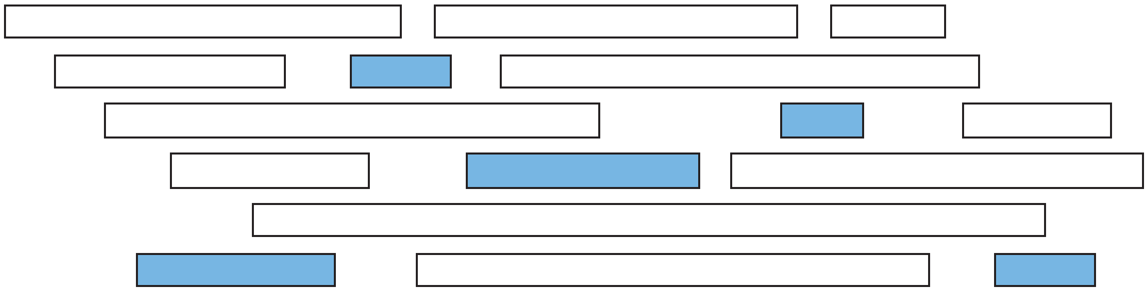
Symbol	Codeword
A	0
B	100
C	101
D	11



Scheduling problem

Scheduling classes

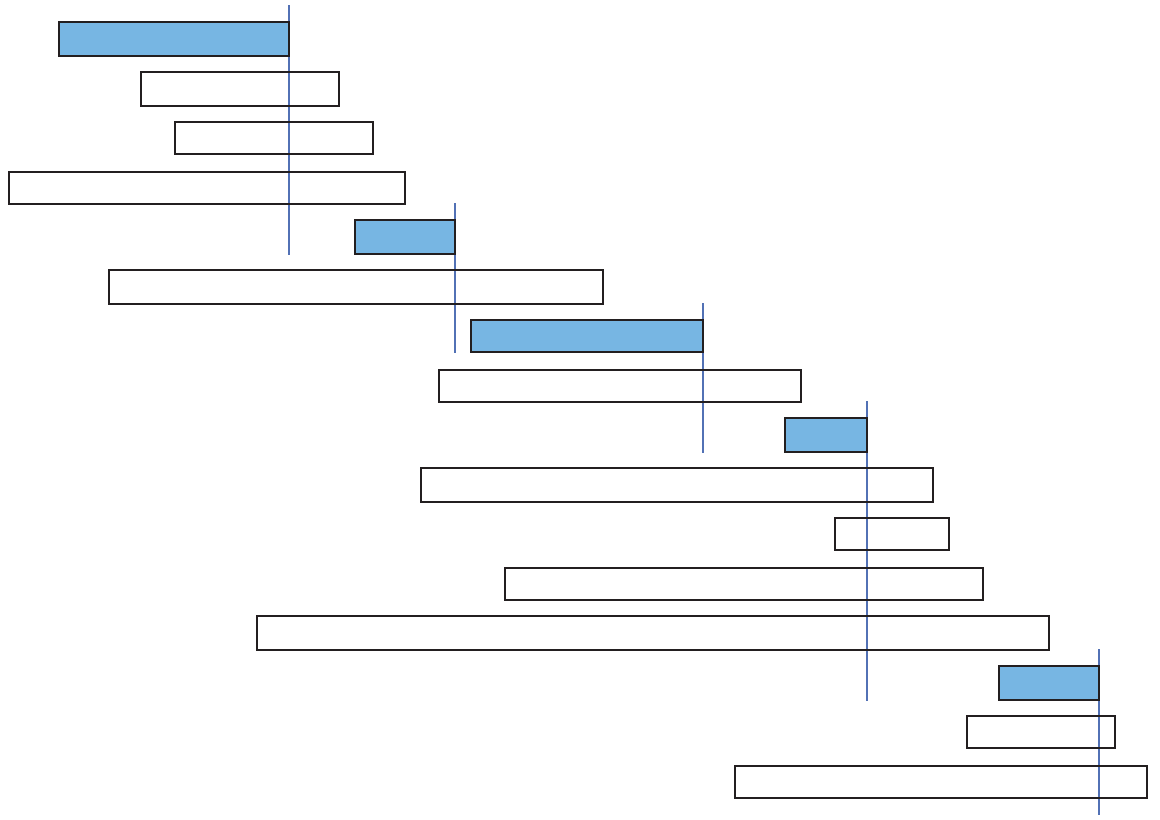
- Suppose you are given two arrays $S[1..\ n]$ and $F[1..\ n]$ listing the start and finish times of each class.
- Your task is to choose the largest possible subset $X \in \{1, 2,.., n\}$ so that for any pair $i, j \in X$, either $S[i] > F[j]$ *or* $S[j] > F[i]$.



A maximal conflict-free schedule for a set of classes.

Greedy algorithm

- Sort classes by finish times.
- Scan classes in the sorted order and choose the next class that does not conflict with the latest class so far.



The same classes sorted by finish times and the greedy schedule.

```
GreedySchedule(S[1..n], F[1..n]):
    sort F and permute S to match
    count ← 1
    X[count] ← 1
    for i ← 2 to n
        if S[i] > F[X[count]]
            count ← count + 1
            X[count] ← i
    return X[1..count]
```

- Running time?

Correctness

- To prove that this greedy algorithm actually gives us a maximal conflict-free schedule, we use an *exchange argument*.
- Note that there can be maximal conflict-free schedules different from the output of the algorithm.
- We only claim that at least one of the maximal schedules is the one that the greedy algorithm produces.

Correctness

- **Lemma : At least one maximal conflict-free schedule includes the class that finishes first.**
- Proof:
- Let f be the class that finishes first.
- Suppose we have a maximal conflict-free schedule X that does not include f.
- Let g be the first class in X to finish.
- Since f finishes before g does, f cannot conflict with any class in the set X -$\{g\}$.
- Thus, the schedule $X' = X \cup \{f\} - \{g\}$ is also conflict-free.
- Since X' has the same size as X, it is also maximal.

Correctness

- **Theorem : The greedy schedule is an optimal schedule.**
- Proof:
- Let f be the class that finishes first, and let L be the subset of classes that start after f finishes.
- The previous lemma implies that some optimal schedule contains f, so the best schedule that contains f is an optimal schedule.
- The best schedule that includes f must contain an optimal schedule for the classes that do not conflict with f, that is, an optimal schedule for L.
- The greedy algorithm chooses f and then, by the inductive hypothesis, computes an optimal schedule of classes from L.

Proving correctness of greedy algorithms

- *An inductive exchange argument.*
- Assume that there is an optimal solution that is different from the greedy solution.
- Find the ‘first’ difference between the two solutions.
- Argue that we can exchange the optimal choice for the greedy choice without degrading the solution.
- This argument implies by induction that there is an optimal solution that contains the entire greedy solution.