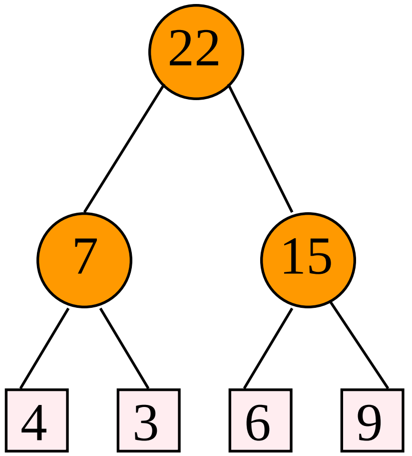
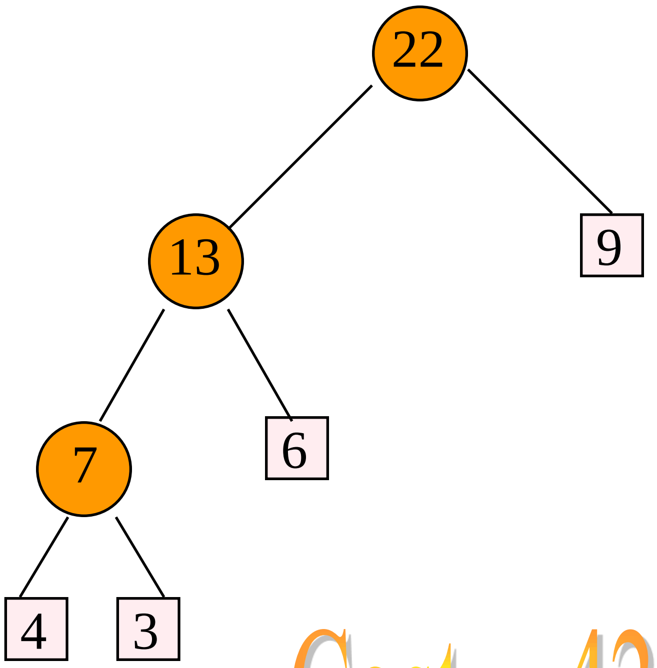
Optimal Merging Of Runs



Cost = 44



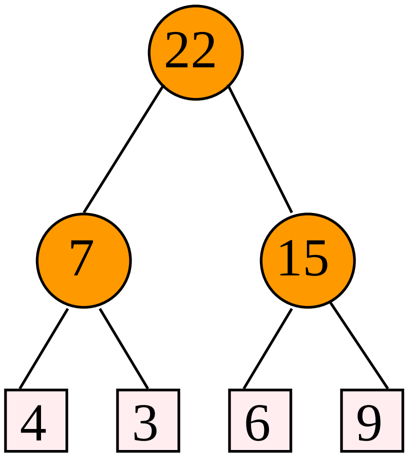
Cost = 42

Best merge order?

Weighted External Path Length

WEPL(T) = Σ(weight of external node i)

* (distance of node i from root of T)



WEPL(T) = 4 * 2 + 3*2 + 6*2 + 9*2

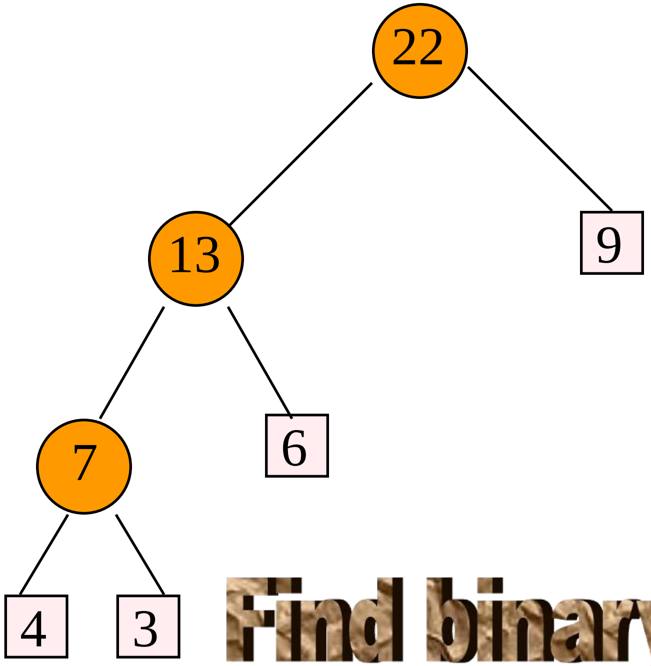
= 44

= Merge Cost

Weighted External Path Length

WEPL(T) = Σ(weight of external node i)

* (distance of node i from root of T)



WEPL(T) = 4 * 3 + 3*3 + 6*2 + 9*1

= 42

= Merge Cost

Find binary tree with minimum WEPL.

Other Applications

- Message coding and decoding.
- Lossless data compression.

Message Coding & Decoding

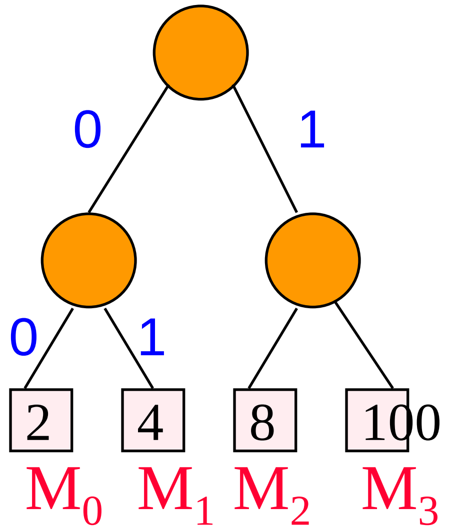
- Messages M_0, M_1, M_2, …, M_{n-1} are to be transmitted.
- The messages do not change.
- Both sender and receiver know the messages.
- So, it is adequate to transmit a code that identifies the message (e.g., message index).
- M_i is sent with frequency f_i.
- Select message codes so as to minimize transmission and decoding times.

Example

- $n = 4$ messages.
- The frequencies are [2, 4, 8, 100].
- Use 2-bit codes [00, 01, 10, 11].
- Transmission cost $= 2*2 + 4*2 + 8*2 + 100*2$

  $= 228$.
- Decoding is done using a binary tree.

Example



- Decoding cost = 2*2 + 4*2 + 8*2 + 100*2

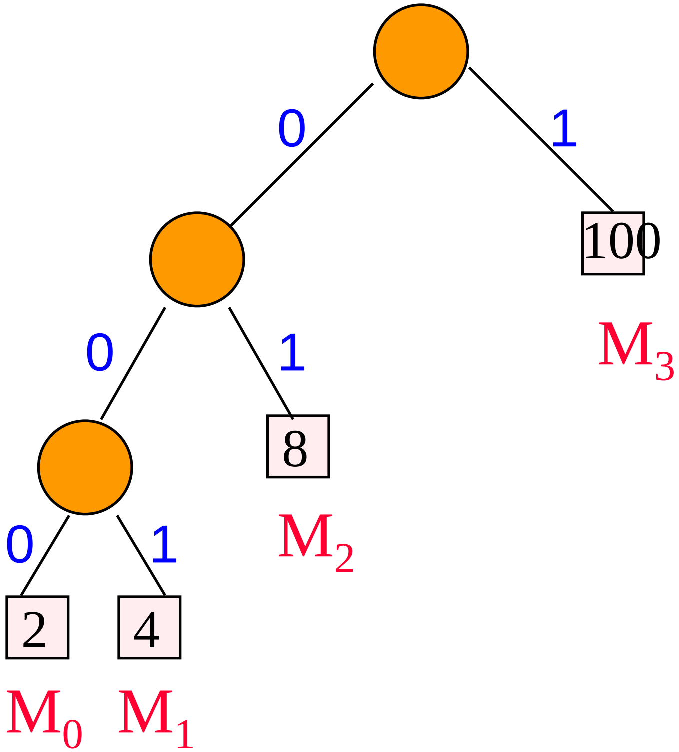
  = 228

  = transmission cost

  = WEPL

Example

- Every binary tree with n external nodes defines a code set for n messages.



- Decoding cost

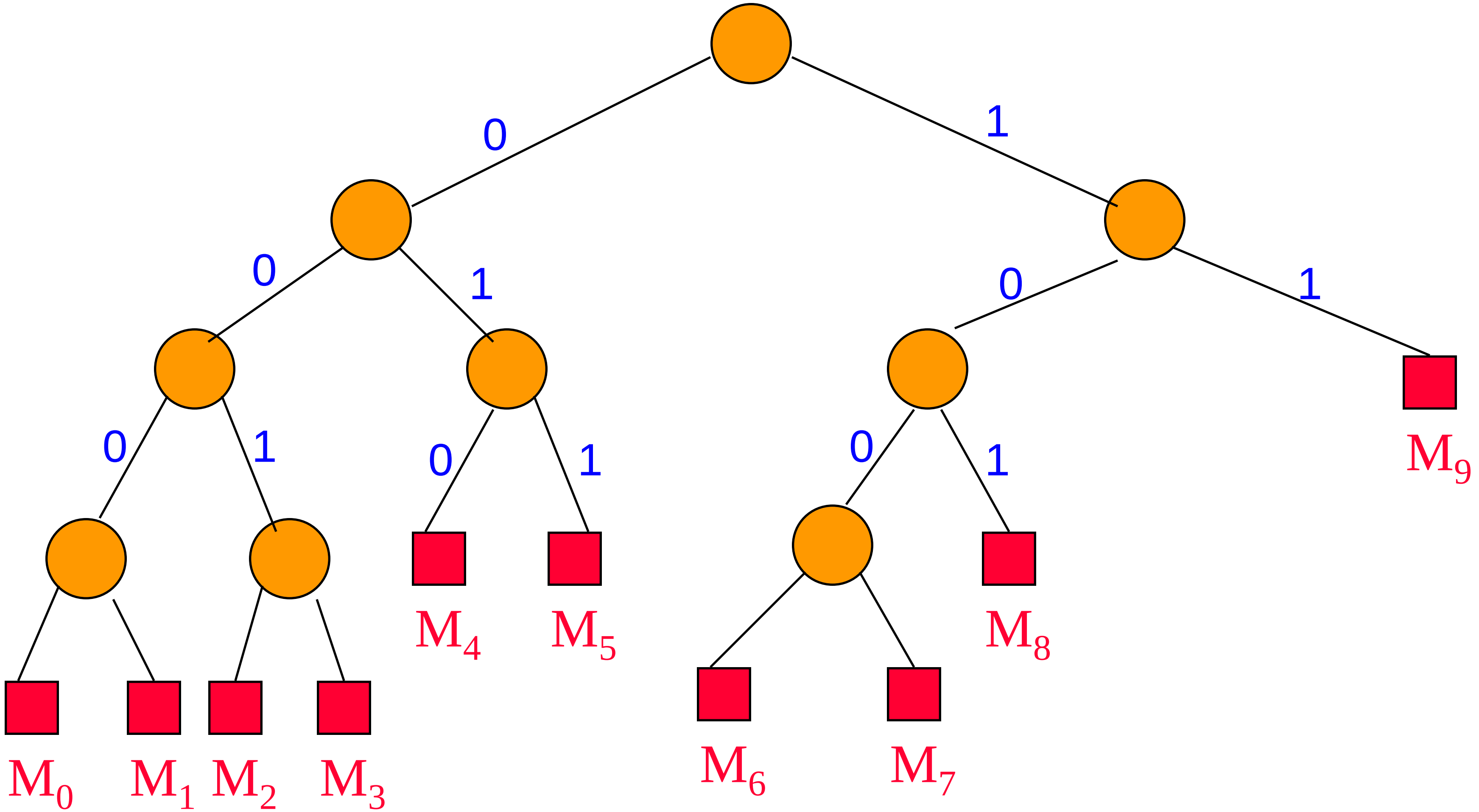
$= 2*3 + 4*3 + 8*2 + 100*1$

$= 144$

= transmission cost

= WEPL

Another Example



No code is a prefix of another!

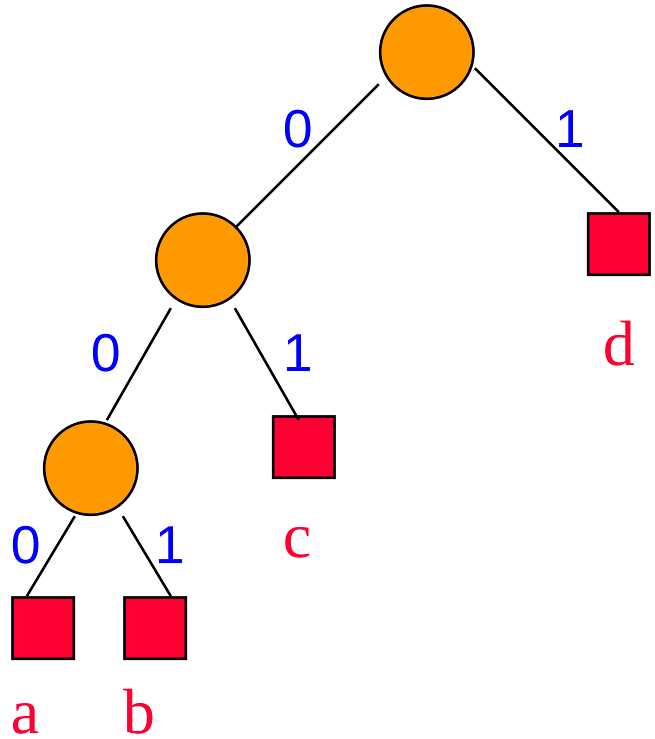
Lossless Data Compression

- Alphabet = {a, b, c, d}.
- String with 10 as, 5 bs, 100 cs, and 900 ds.
- Use a 2-bit code.
  - a = 00, b = 01, c = 10, d = 11.
  - Size of string = 10*2 + 5*2 + 100*2 + 900*2
    = 2030 bits.
  - Plus size of code table.

Lossless Data Compression

- Use a variable length code that satisfies prefix property (no code is a prefix of another).
  - a = 000, b = 001, c = 01, d = 1.
  - Size of string = 10*3 + 5*3 + 100*2 + 900*1
    = 1145 bits.
  - Plus size of code table.
  - Compression ratio is approx. 2030/1145 = 1.8.

Lossless Data Compression



- Decode 0001100101…
- addbc…
- Compression ratio is maximized when the decode tree has minimum WEPL.

Huffman Trees

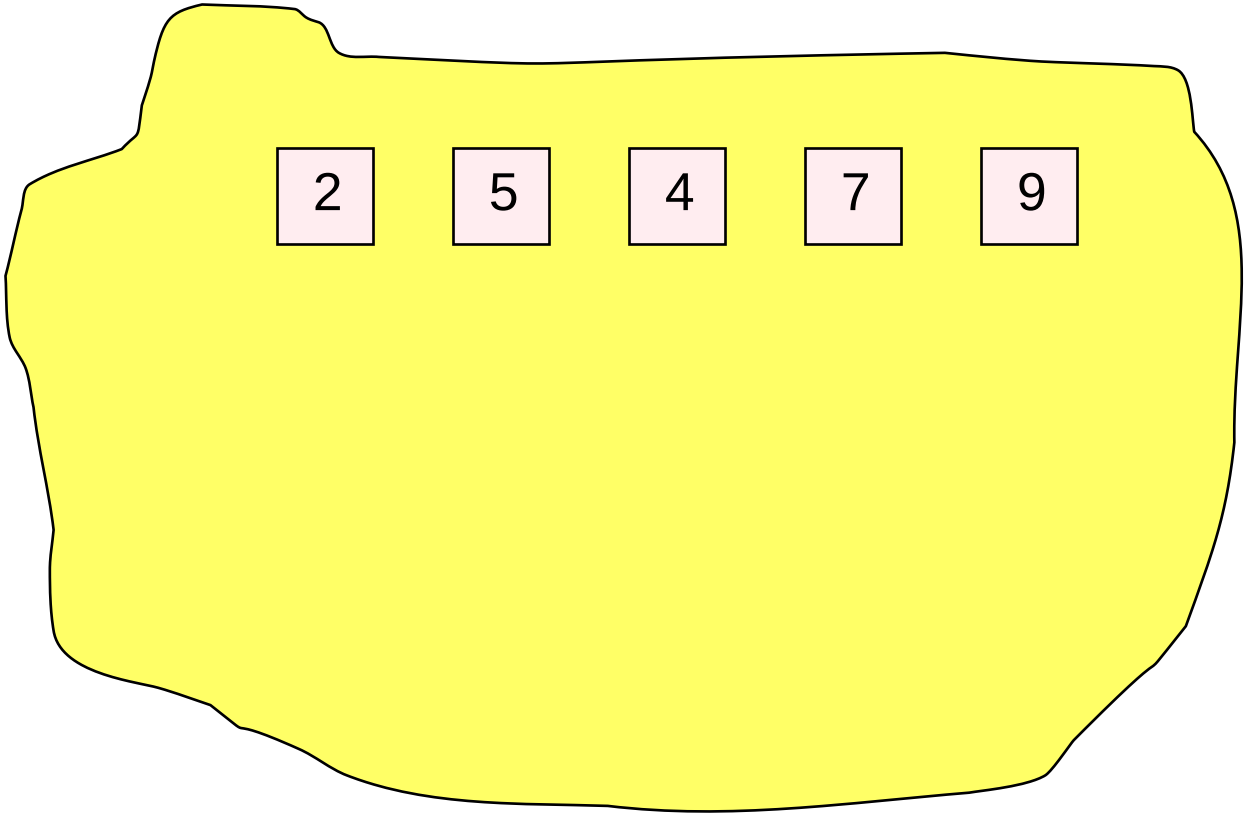
- Trees that have minimum WEPL.
- Binary trees with minimum WEPL may be constructed using a greedy algorithm.
- For higher order trees with minimum WEPL, a preprocessing step followed by the greedy algorithm may be used.
- Huffman codes: codes defined by minimum WEPL trees.

Greedy Algorithm For Binary Trees

- Start with a collection of external nodes, each with one of the given weights. Each external node defines a different tree.
- Reduce number of trees by 1.
  - Select 2 trees with minimum weight.
  - Combine them by making them children of a new root node.
  - The weight of the new tree is the sum of the weights of the individual trees.
  - Add new tree to tree collection.
- Repeat reduce step until only 1 tree remains.

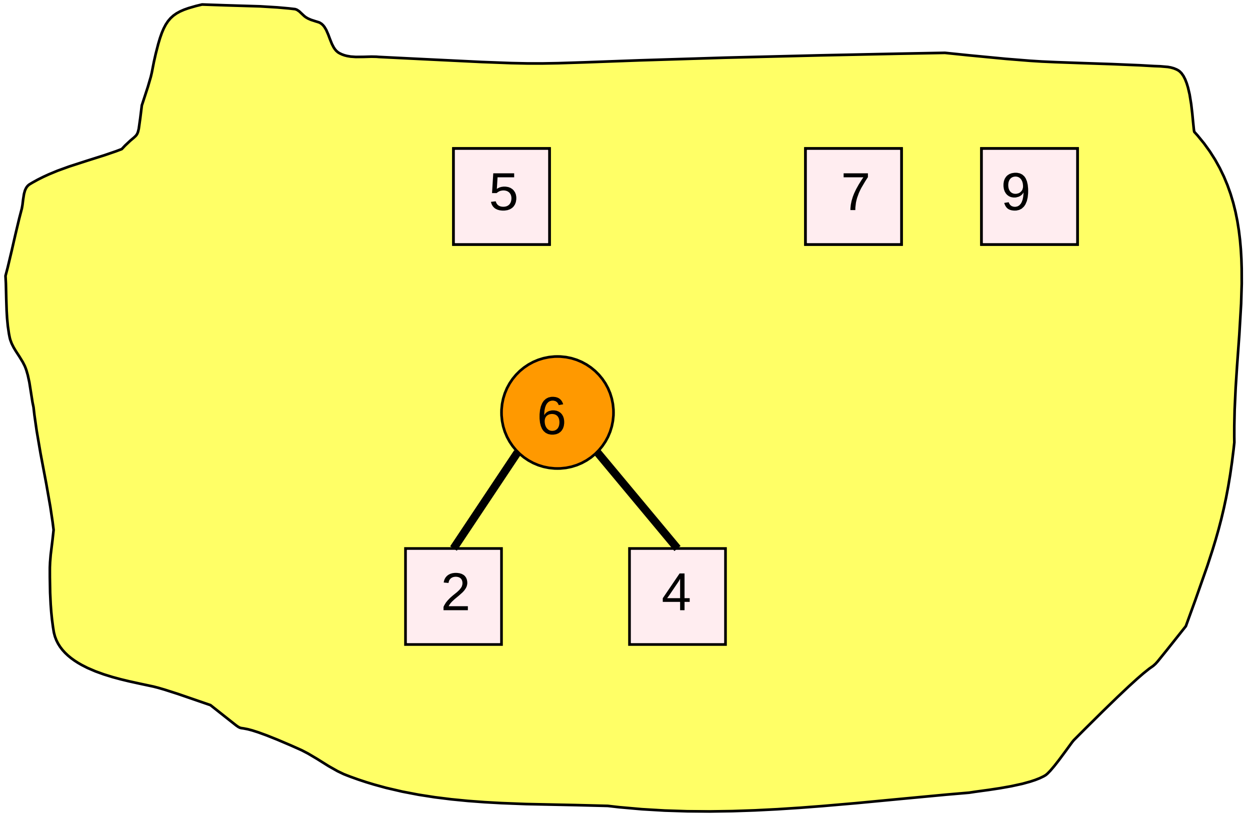
Example

- n = 5, w[0:4] = [2, 5, 4, 7, 9].



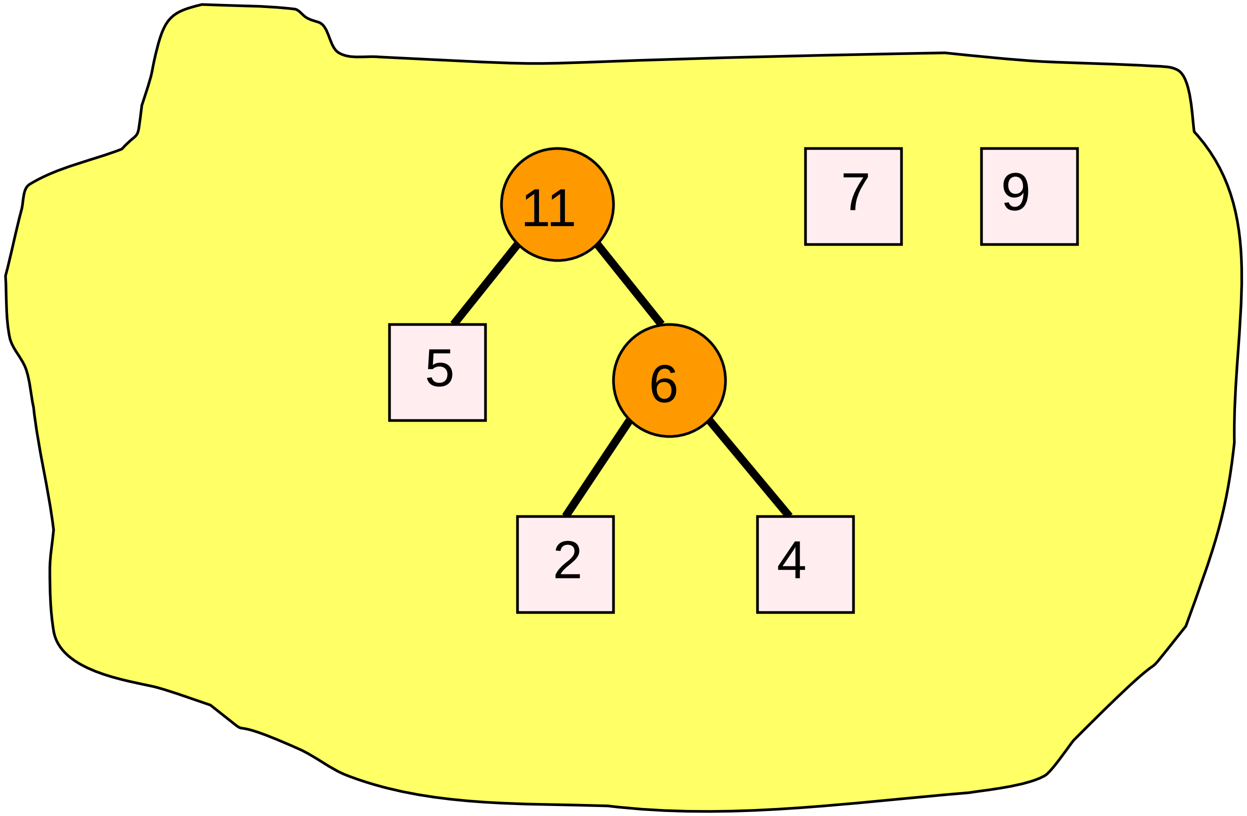
Example

- n = 5, w[0:4] = [2, 5, 4, 7, 9].



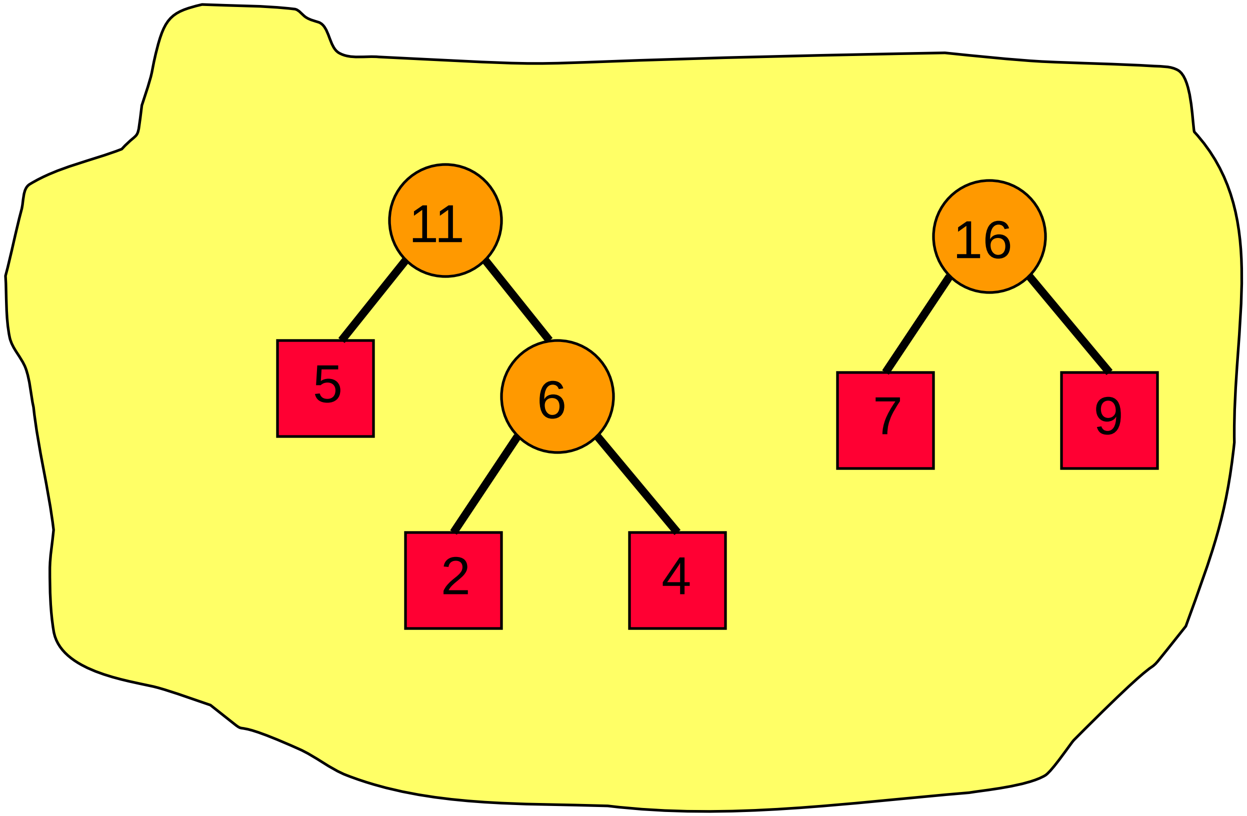
Example

- n = 5, w[0:4] = [2, 5, 4, 7, 9].



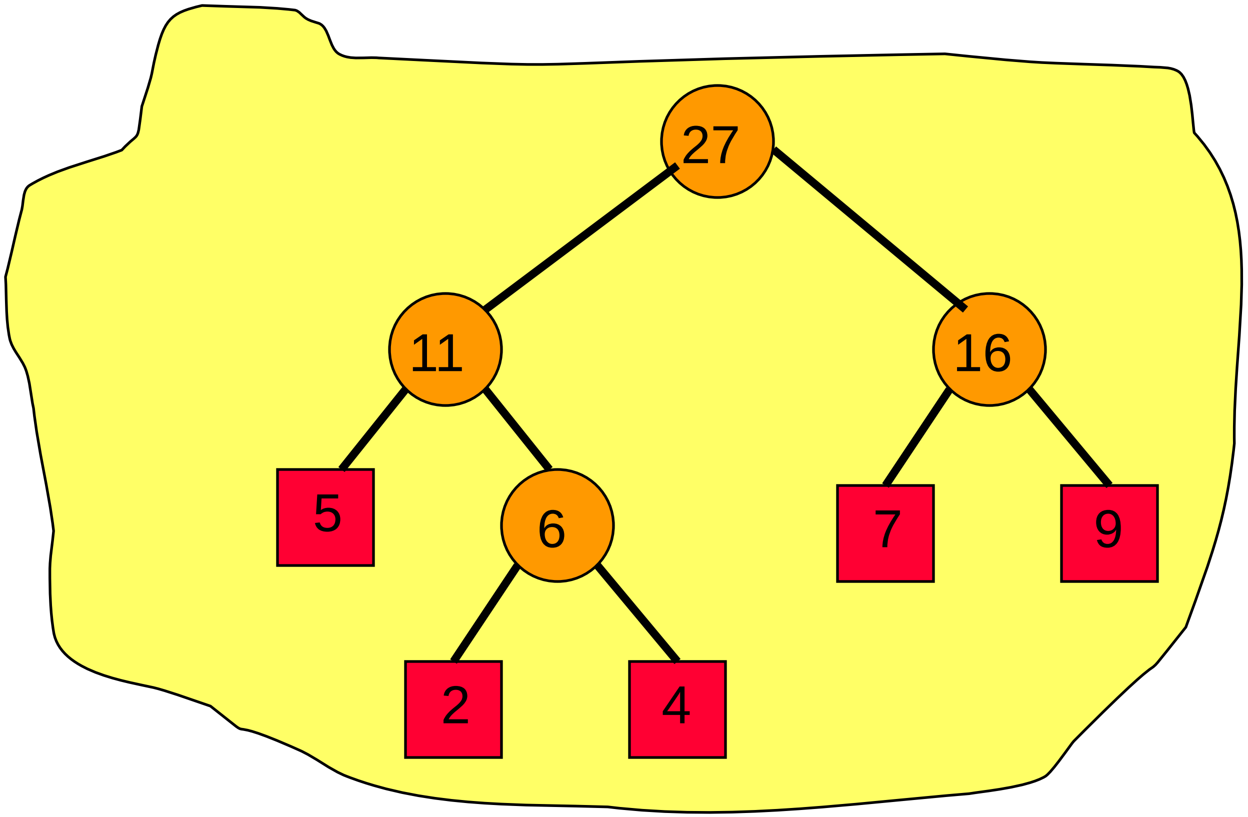
Example

- n = 5, w[0:4] = [2, 5, 4, 7, 9].



Example

- n = 5, w[0:4] = [2, 5, 4, 7, 9].

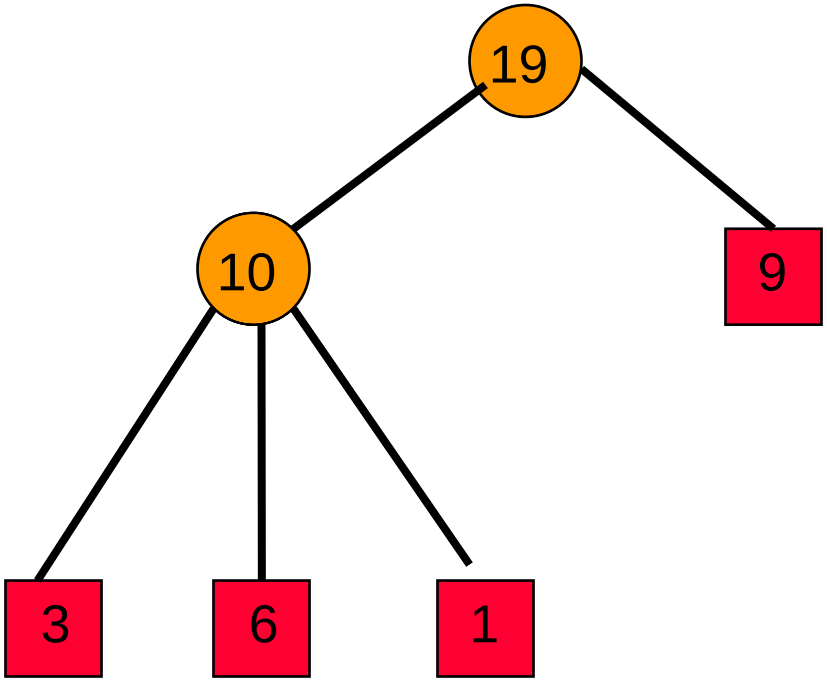


Data Structure For Tree Collection

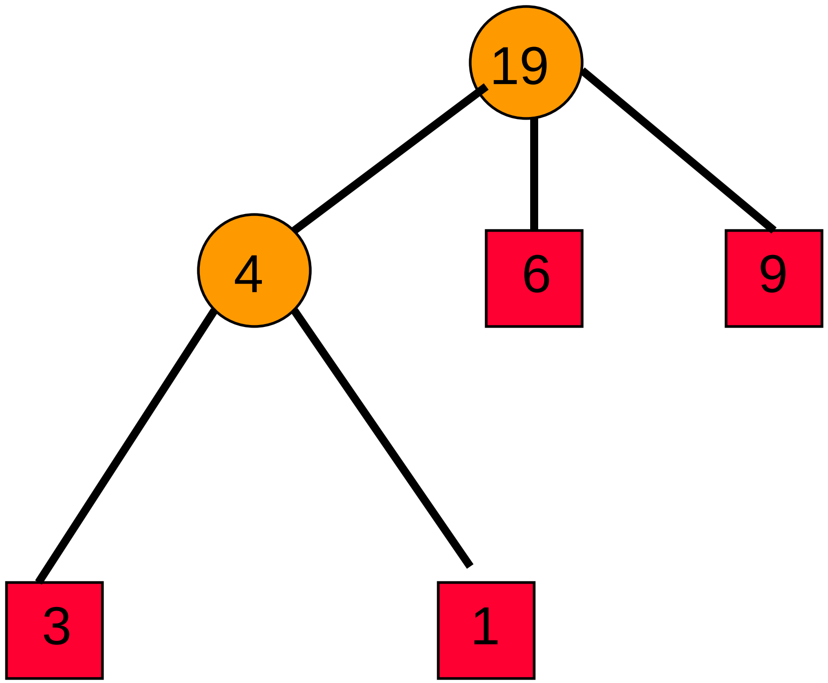
- Operations are:
  - Initialize with n trees.
  - Remove 2 trees with least weight.
  - Insert new tree.
- Use a min heap.
- Initialize … $O(n)$.
- $2(n - 1)$ remove min operations … $O(n \log n)$.
- $n - 1$ insert operations … $O(n \log n)$.
- Total time is $O(n \log n)$.
- Or, $(n - 1)$ remove mins and $(n - 1)$ change mins.

Higher Order Trees

- Greedy scheme doesn't work!
- 3-way tree with weights [3, 6, 1, 9].

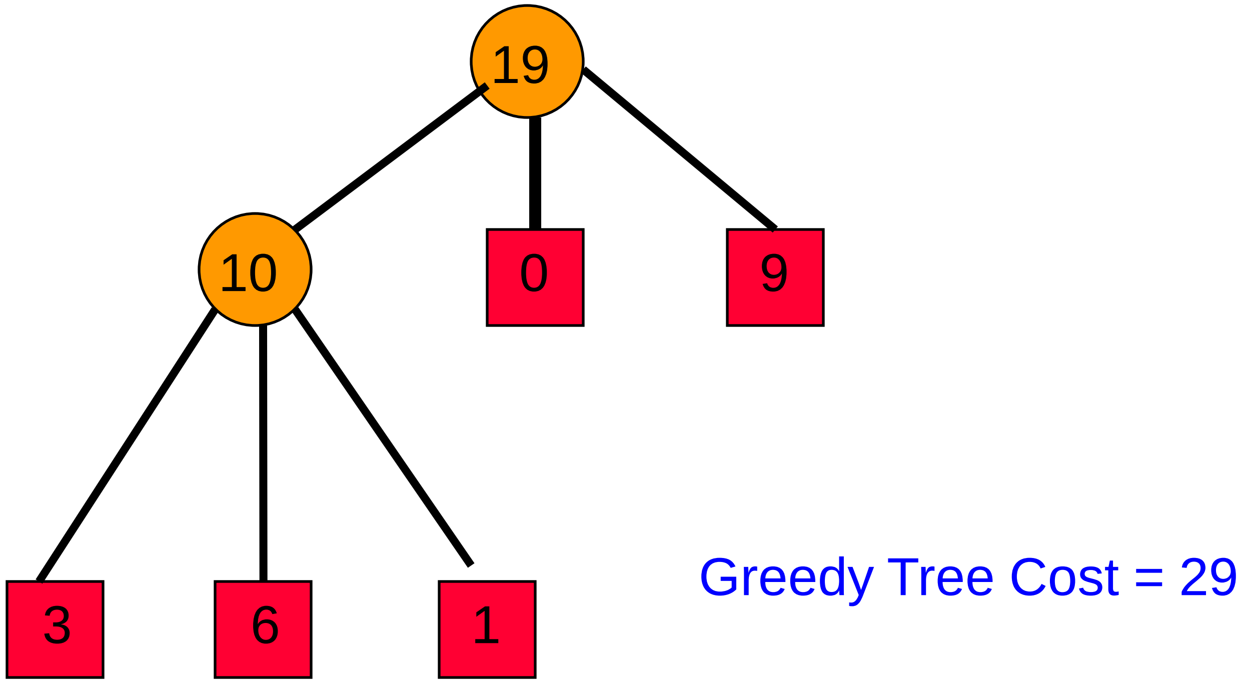


Greedy Tree Cost = 29



Optimal Tree Cost = 23

Cause Of Failure



- One node is not a 3-way node.
- A 2-way node is like a 3-way node, one of whose children has a weight of 0.
- Must start with enough runs/weights of length 0 so that all nodes are 3-way nodes.

How Many Length 0 Runs To Add?

- k-way tree, $k > 1$.
- Initial number of runs is r.
- Add least $q >= 0$ runs of length 0.
- Each k-way merge reduces the number of runs by $k – 1$.
- Number of runs after s k-way merges is

  $r + q – s(k – 1)$
- For some positive integer s, the number of remaining runs must become 1.

How Many Length 0 Runs To Add?

- So, we want

  $r + q - s(k-1) = 1$

  for some positive integer s.
- So, $r + q - 1 = s(k - 1)$.
- Or, $(r + q - 1) \bmod (k - 1) = 0$.
- Or, $r + q - 1$ is divisible by $k - 1$.
  - This implies that $q < k - 1$.
- $(r - 1) \bmod (k - 1) = 0 \Rightarrow q = 0$.
- $(r - 1) \bmod (k - 1) \neq 0 \Rightarrow$

  $$q = k - 1 - (r - 1) \bmod (k - 1).$$
- Or, $q = (1 - r) \bmod (k - 1)$.

Examples

- $k = 2$.
  - $q = (1 - r) \bmod (k - 1) = (1 - r) \bmod 1 = 0$.
  - So, no runs of length 0 are to be added.
- $k = 4$, $r = 6$.
  - $q = (1 - r) \bmod (k - 1) = (1 - 6) \bmod 3$

    $= (-5) \bmod 3$

    $= (6 - 5) \bmod 3$

    $= 1$.
  - So, must start with 7 runs, and then apply greedy method.