

## Tournament Trees

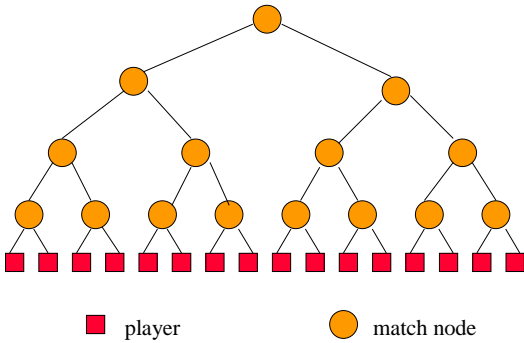


Winner trees.  
Loser Trees.

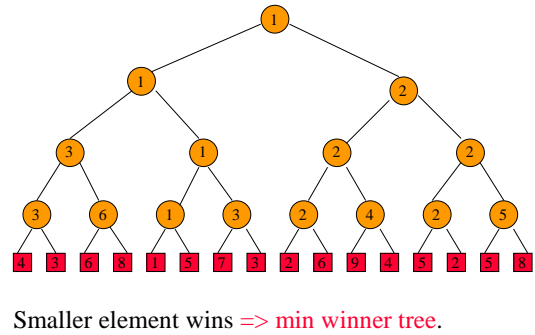
## Winner Trees

Complete binary tree with  $n$  external nodes and  $n - 1$  internal nodes.  
External nodes represent tournament players.  
Each internal node represents a match played between its two children; the winner of the match is stored at the internal node.  
Root has overall winner.

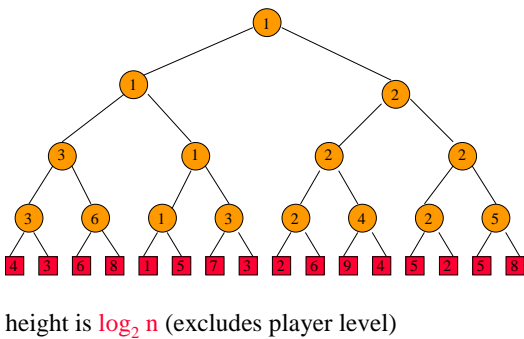
### Winner Tree For 16 Players



### Winner Tree For 16 Players



### Winner Tree For 16 Players



## Complexity Of Initialize

- $O(1)$  time to play match at each match node.
- $n - 1$  match nodes.
- $O(n)$  time to initialize  $n$  player winner tree.

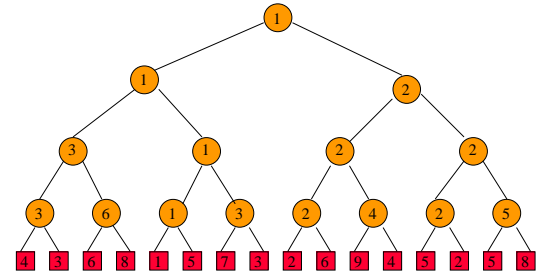
## Applications

Sorting.

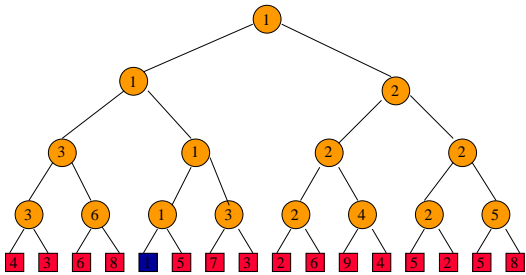
Put elements to be sorted into a winner tree.

Repeatedly extract the winner and replace by a large value.

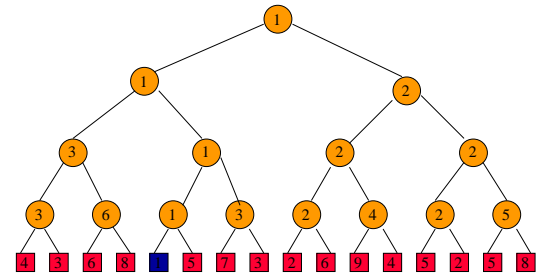
## Sort 16 Numbers



## Sort 16 Numbers

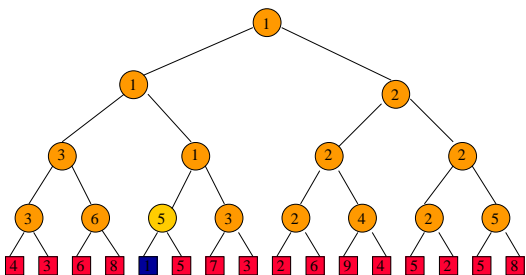


## Sort 16 Numbers



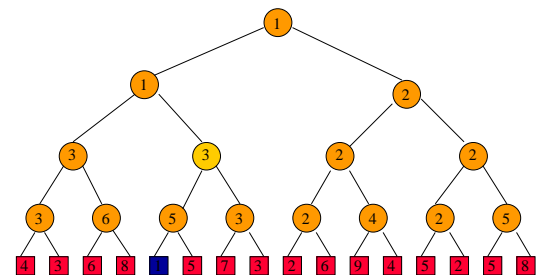
Sorted array.

## Sort 16 Numbers

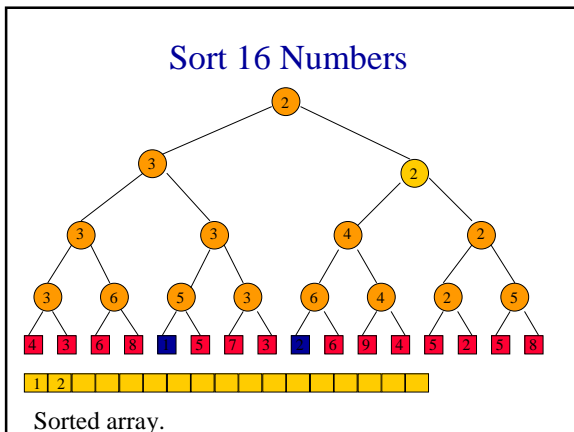
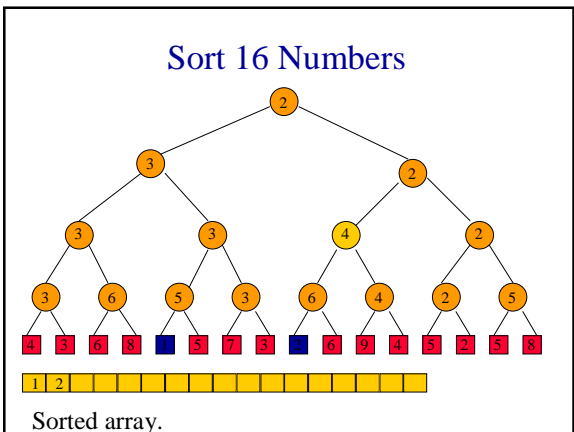
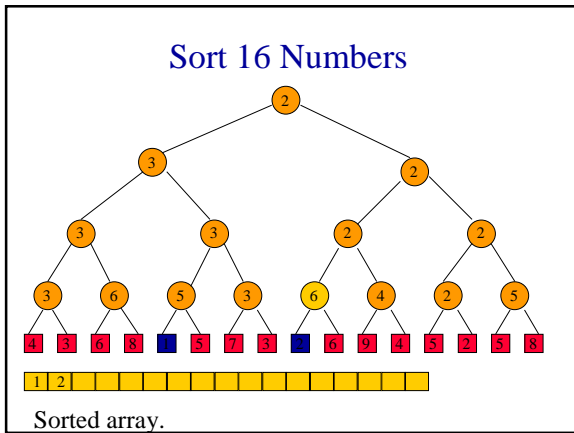
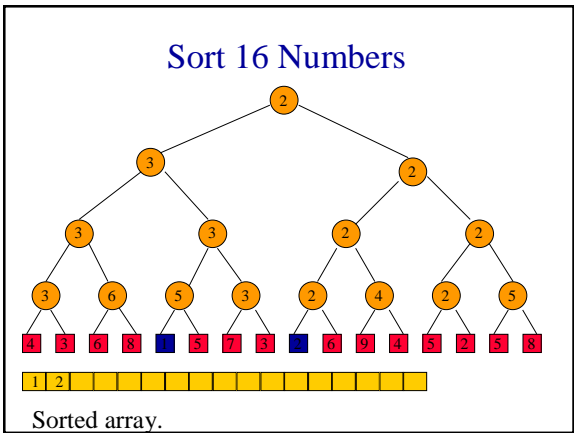
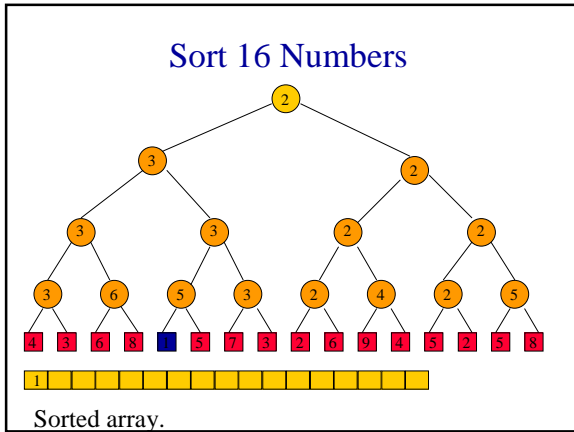
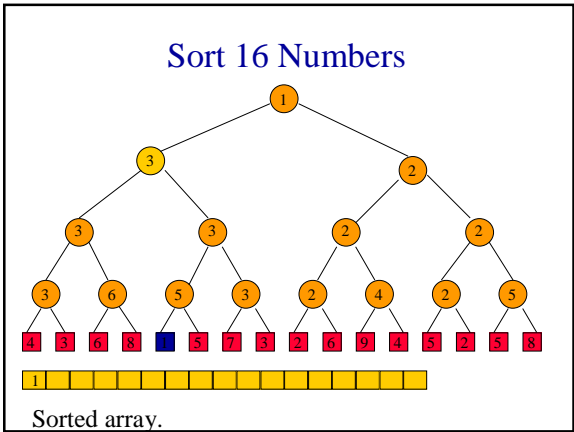


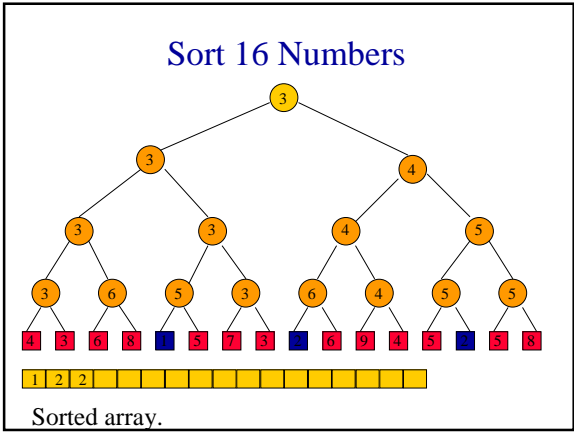
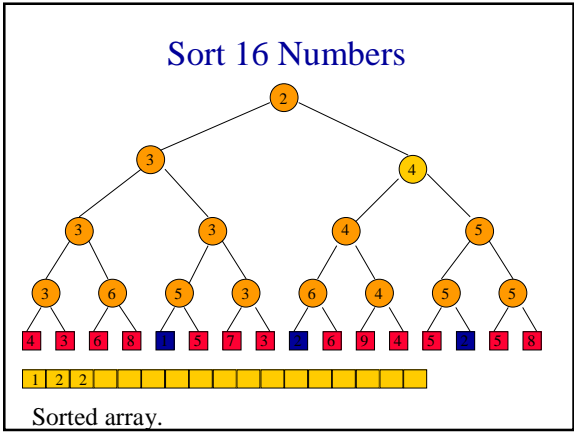
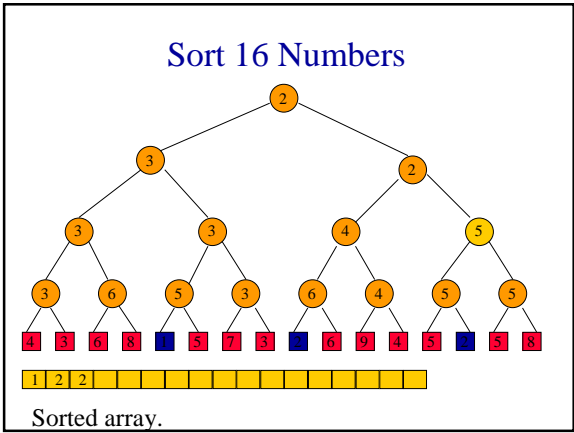
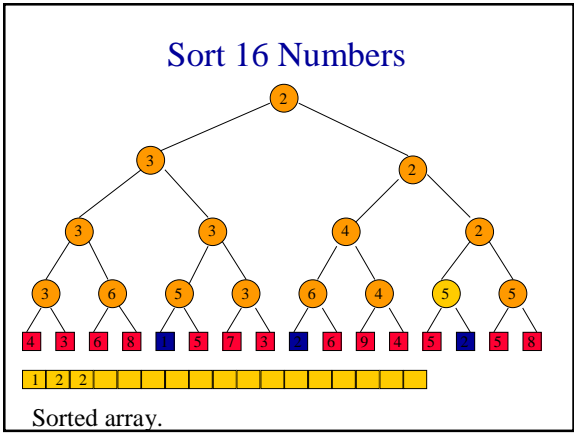
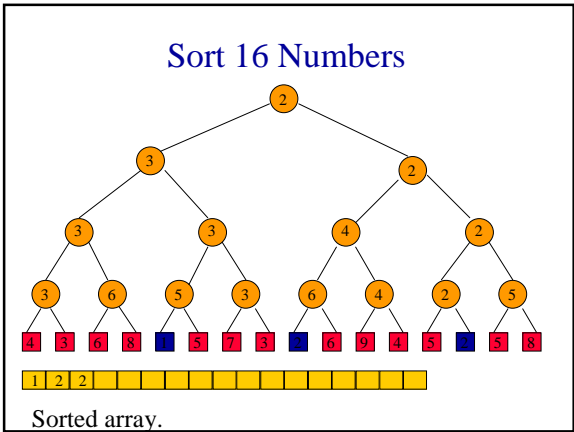
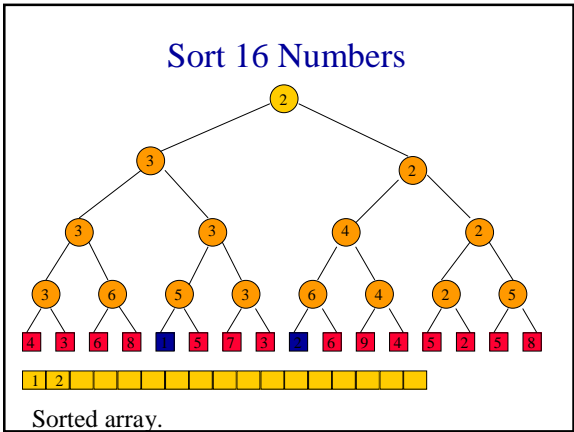
Sorted array.

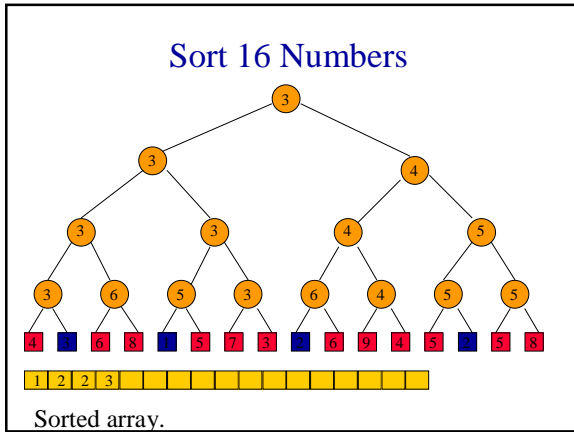
## Sort 16 Numbers



Sorted array.





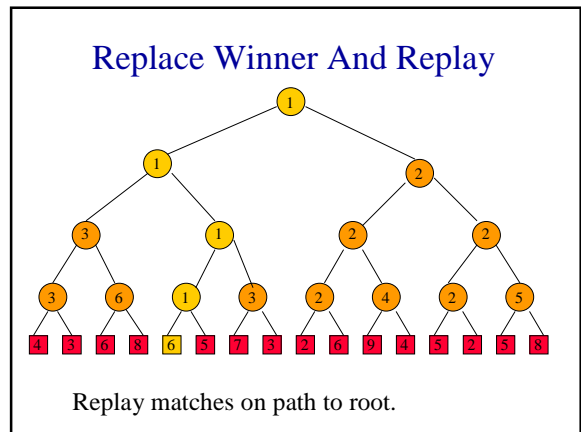
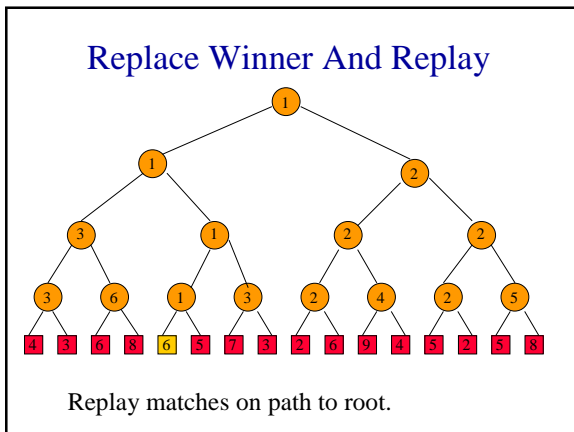
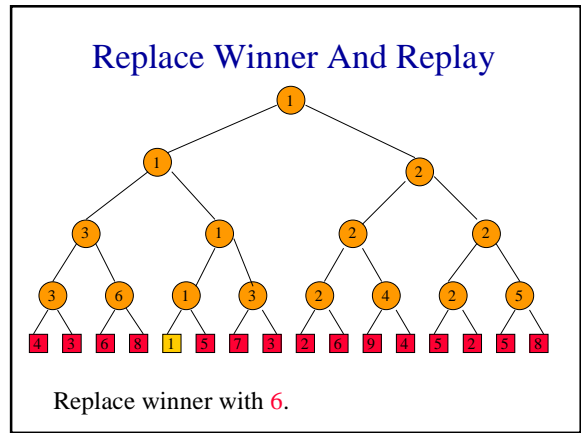


### Time To Sort

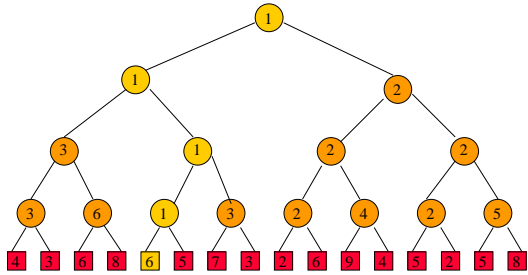
- Initialize winner tree.
  - $O(n)$  time
- Remove winner and replay.
  - $O(\log n)$  time
- Remove winner and replay  $n$  times.
  - $O(n \log n)$  time
- Total sort time is  $O(n \log n)$ .
- Actually  $\Theta(n \log n)$ .

### Winner Tree Operations

- Initialize
  - $O(n)$  time
- Get winner
  - $O(1)$  time
- Remove/replace winner and replay
  - $O(\log n)$  time
  - more precisely  $\Theta(\log n)$



### Replace Winner And Replay

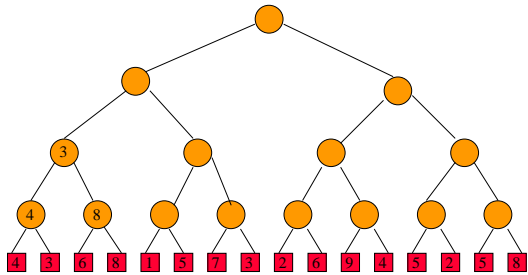


Opponent is player who lost last match played at this node.

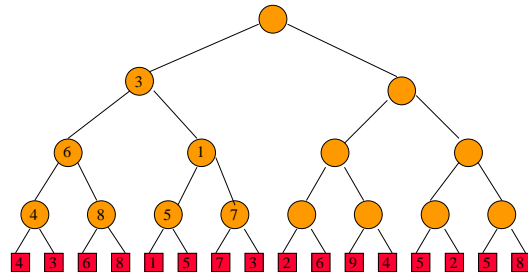
### Loser Tree

Each match node stores the match loser rather than the match winner.

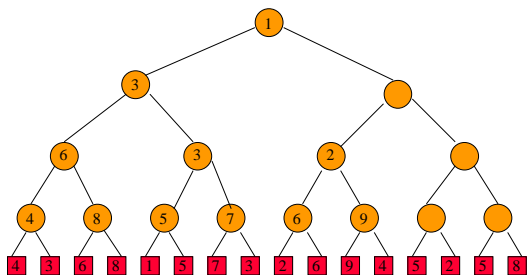
### Min Loser Tree For 16 Players



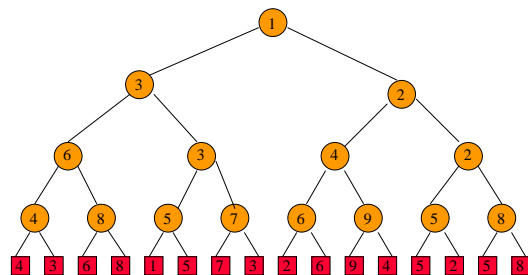
### Min Loser Tree For 16 Players

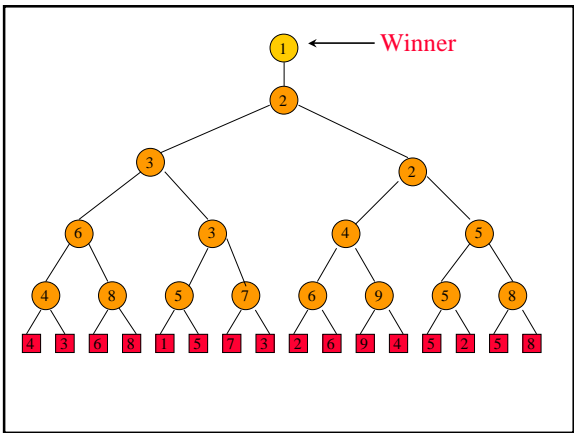
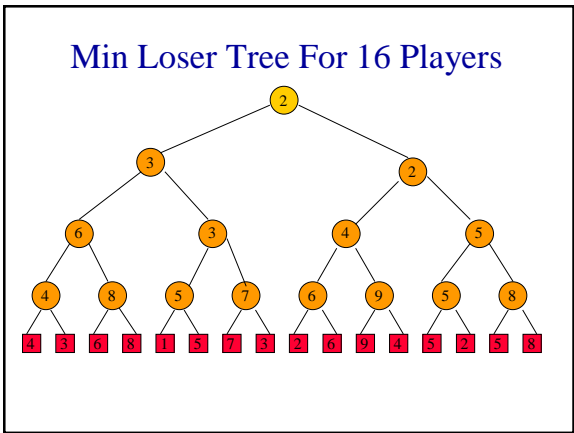
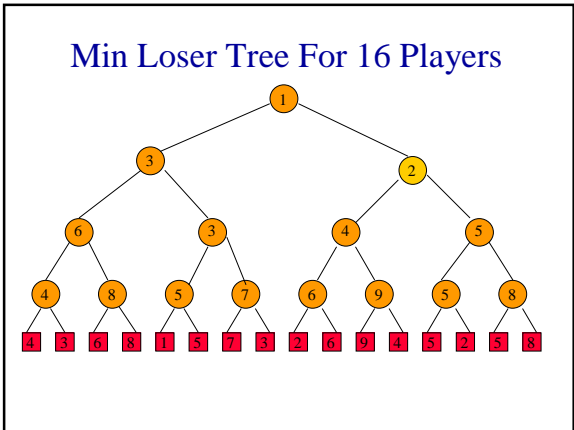
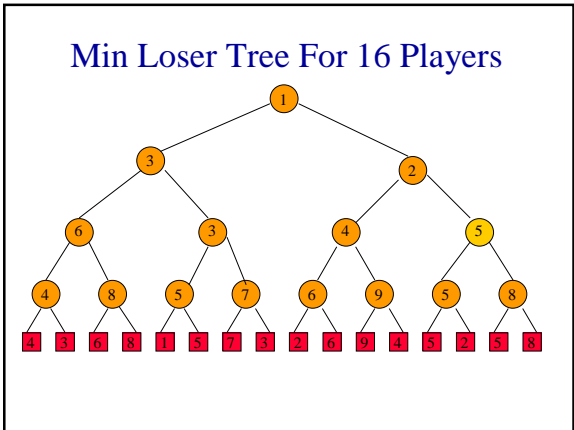


### Min Loser Tree For 16 Players




### Min Loser Tree For 16 Players

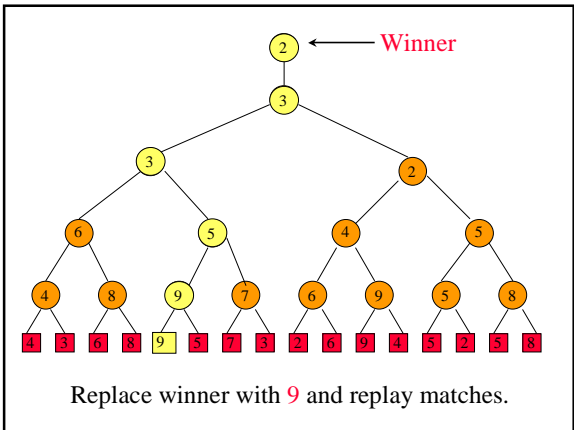




### Complexity Of Loser Tree Initialize



- One match at each match node.
- One store of a left child winner.
- Total time is  $O(n)$ .
- More precisely  $\Theta(n)$ .



## Complexity Of Replay



- One match at each level that has a match node.
- $O(\log n)$
- More precisely  $\Theta(\log n)$ .

## More Tournament Tree Applications

- $k$ -way merging of runs during an external merge sort
- Truck loading

## Truck Loading



- $n$  packages to be loaded into trucks
- each package has a weight
- each truck has a capacity of  $c$  tons
- minimize number of trucks

## Truck Loading

$n = 5$  packages  
weights [2, 5, 6, 3, 4]  
truck capacity  $c = 10$

Load packages from left to right. If a package doesn't fit into current truck, start loading a new truck.

## Truck Loading

$n = 5$  packages  
weights [2, 5, 6, 3, 4]  
truck capacity  $c = 10$

truck1 = [2, 5]

truck2 = [6, 3]

truck3 = [4]

uses 3 trucks when 2 trucks suffice

## Truck Loading

$n = 5$  packages  
weights [2, 5, 6, 3, 4]  
truck capacity  $c = 10$

truck1 = [2, 5, 3]

truck2 = [6, 4]

## Bin Packing

- $n$  items to be packed into bins
- each item has a size
- each bin has a capacity of  $c$
- minimize number of bins

## Bin Packing

Truck loading is same as bin packing.

Truck is a bin that is to be packed (loaded).

Package is an item/element.

Bin packing to minimize number of bins is NP-hard.

Several fast heuristics have been proposed.

## Bin Packing Heuristics

- First Fit.
  - Bins are arranged in left to right order.
  - Items are packed one at a time in given order.
  - Current item is packed into leftmost bin into which it fits.
  - If there is no bin into which current item fits, start a new bin.

## First Fit

$n = 4$

weights = [4, 7, 3, 6]

capacity = 10



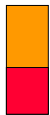
Pack red item into first bin.

## First Fit

$n = 4$

weights = [4, 7, 3, 6]

capacity = 10



Pack blue item next.

Doesn't fit, so start a new bin.

## First Fit

$n = 4$

weights = [4, 7, 3, 6]

capacity = 10

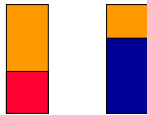


## First Fit

$n = 4$

weights = [4, 7, 3, 6]

capacity = 10



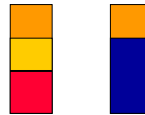
Pack yellow item into first bin.

## First Fit

$n = 4$

weights = [4, 7, 3, 6]

capacity = 10



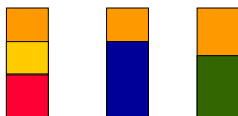
Pack green item.  
Need a new bin.

## First Fit

$n = 4$

weights = [4, 7, 3, 6]

capacity = 10



Not optimal.  
2 bins suffice.

## Bin Packing Heuristics

- First Fit Decreasing.
  - Items are sorted into decreasing order.
  - Then first fit is applied.

## Bin Packing Heuristics

- Best Fit.
  - Items are packed one at a time in given order.
  - To determine the bin for an item, first determine set  $S$  of bins into which the item fits.
  - If  $S$  is empty, then start a new bin and put item into this new bin.
  - Otherwise, pack into bin of  $S$  that has least available capacity.

## Bin Packing Heuristics

- Best Fit Decreasing.
  - Items are sorted into decreasing order.
  - Then best fit is applied.

## Performance



- For first fit and best fit:

$$\text{Heuristic Bins} \leq (17/10)(\text{Minimum Bins}) + 2$$

- For first fit decreasing and best fit decreasing:

$$\text{Heuristic Bins} \leq (11/9)(\text{Minimum Bins}) + 4$$

## Complexity Of First Fit



Use a max tournament tree in which the players are  $n$  bins and the value of a player is the available capacity in the bin.

$O(n \log n)$ , where  $n$  is the number of items.