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.

Abstract Data Type WinnerTree

```cpp
AbstractData Type WinnerTree
{
  instances
  complete binary trees with each node pointing to the winner of the match played there; the external nodes represent the players
  operations
  initialize(a): initialize a winner tree for the players in the array a
  winner(): return the tournament winner
  rePlay(i): replay matches following a change in player i
}
```

Winner Tree For 16 Players

Smaller element wins ⇒ min winner tree.

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.

height is $\log_2 n$ (excludes player level)
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 with 6.

Replace matches on path to root.

Replace matches on path to root.
Opponent is player who lost last match played at this node.

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

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 $\Theta(n)$.

Replace winner with 9 and replay matches.
Complexity of Replay

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

Applications

- Sorting.
- Insert 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.

Sorted array.
Sort 16 Numbers

Sorted array.

Sort 16 Numbers

Sorted array.

Sort 16 Numbers

Sorted array.

Sort 16 Numbers

Sorted array.
Sort 16 Numbers

Sorted array.

Sort 16 Numbers

Sorted array.

Sort 16 Numbers

Sorted array.

Sort 16 Numbers

Sorted array.
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 $O(n \log n)$.
- More precisely $\Theta(n \log n)$.

More Tournament Tree Applications

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

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

---

**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.

---

Pack blue item next.

Doesn’t fit, so start a new bin.

---

Start a new bin.

Pack blue item into the 2nd bin.
First Fit

\( n = 4 \)
weights = [4, 7, 3, 6]
capacity = 10

Pack orange item into first bin.

Pack green item.
Need a new bin.

Bin Packing Heuristics

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

weights = [7, 6, 4, 3]
capacity = 10
Not optimal.
2 bins suffice.
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.

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.