# Parallel Algorithms

Umut Acar
Carnegie Mellon University

July 12, 2018

## 1 Recap

Application of sequences

- Maximum Contiguous Subsequence Sum (MCSS)
  - Scans
  - Divide and Conquer
  - Both: O(n) work, O(lg n) span - Esentially Optimal
- BFS on Graphs
  - O(m lg n) work, O(m) work, O(diameter) span (lg n) - "Optimal"

## 2 Binary Search Trees (Sets and Dictionaries)

```
datatype α tree = NODE (α tree, α, α tree) | LEAF
find:  α × α tree → bool
insert: α × α tree →α tree
delete:  α × α tree →α tree
```

Parallel goal: Do multiple finds, inserts, and deletes simultaneously.

```
intersection: α tree × α tree → α tree
union: α tree × α tree → α tree
difference: α tree × α tree → α tree
```

intersection: $k(t_1) \cap k(t_2)$
union: $k(t_1) \cup k(t_2)$
difference: $k(t_1) \backslash k(t_2)$

# 3 Sequential BST's: Balancing

- Splay Trees
- Red-Black Trees
- AVL Trees
- Weight-Balanced Trees
- Treaps - Probabilistically balanced trees

```
split: α tree × α → bool × α tree × α tree
join: α tree × α tree → α tree
singleton: α → α tree
```

## 3.1 use split, join: minimalist implementation

```
find t k =
    let
        val (found, l, r) = split t k
    in
        found
    end

insert k t =
    let
        val (found, l, r) = split t k
    in
        if found then t
        else
            join (l, join (singleton(k), r))
        end
    end

delete k t =
    let
        val (_, l, r) = split t k
    in
        join(l, r)
    end
```

```
intersection t u =
    case (t, u) of
        (LEAF, u) => LEAF
        (t, LEAF) => LEAF
        (NODE(l1, k, r1), u) =>
            let
                val (flag, l2, r2) = split k u
                val (l, r) = intersection(l1, l2)
                             || intersection(r1, r2)
            in
                if flag then
                    join(l, join(singleton(k), r))
                else
                    join(l,r)
                end
            end
    end

union t u =
    case (t, u) of
        (LEAF, u) => u
        (t, LEAF) => t
        (NODE(l1, k, r1), u) =>
            let
                val (_, l2, r2) = split k u
                val (l, r) = union(l1, l2)
                             || union(r1, r2)
            in
                join(l, join(singleton(k), r))
            end
    end
```

## 3.2 Efficiency

### 3.2.1 Work

Assumption: $m \leq n$

$$\begin{aligned} W(m,n) &= 2W\left(\frac{m}{2},\frac{n}{2}\right) + O(lg(m+n)) \\ &= m * lg(m+n) \\ &= O\left(m * lg\left(\frac{n+m}{m}\right)\right) \\ &= O\left(m * lg\left(\frac{n}{m}\right)\right) \end{aligned}$$

So, if $m \sim n$ this become O(n)
If, $m << n$ then this becomes O(m*lg(n)) which is OPTIMAL

## 3.3 Span

$$O\left(lg^2(m+n)\right)$$