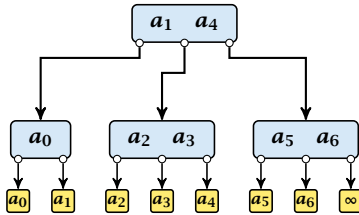


## 4.5 Inserting into a (2, 3)-tree

Given a (2, 3)-tree with  $n$  elements, and a sequence  $x_0 < x_1 < x_2 < \dots < x_k$  of elements. We want to insert elements  $x_1, \dots, x_k$  into the tree ( $k \ll n$ ).

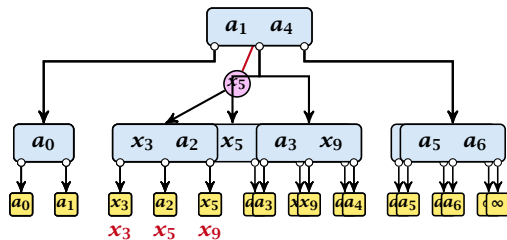
**time:**  $\mathcal{O}(\log n)$ ; **work:**  $\mathcal{O}(k \log n)$



## 4.5 Inserting into a (2, 3)-tree

- determine for every  $x_i$  the leaf element before which it has to be inserted  
time:  $\mathcal{O}(\log n)$ ; work:  $\mathcal{O}(k \log n)$ ; **CREW PRAM**  
all  $x_i$ 's that have to be inserted before the same element form a **chain**
- determine the largest/smallest/middle element of every chain  
time:  $\mathcal{O}(\log k)$ ; work:  $\mathcal{O}(k)$ ;
- insert the middle element of every chain  
compute new chains  
time:  $\mathcal{O}(\log n)$ ; work:  $\mathcal{O}(k_i \log n + k)$ ;  $k_i = \#$ inserted elements  
(computing new chains is constant time)
- repeat Step 3 for logarithmically many rounds  
time:  $\mathcal{O}(\log n \log k)$ ; work:  $\mathcal{O}(k \log n)$ ;

## Step 3



- ▶ each internal node is split into at most two parts
- ▶ each split operation promotes at most one element
- ▶ hence, on every level we want to insert at most one element per successor pointer
- ▶ we can use the same routine for every level

## 4.5 Inserting into a (2, 3)-tree

- ▶ Step 3, works in phases; one phase for every level of the tree
- ▶ Step 4, works in rounds; in each round a different set of elements is inserted

### Observation

We can start with phase  $i$  of round  $r$  as long as phase  $i$  of round  $r - 1$  and (of course), phase  $i - 1$  of round  $r$  has finished.

This is called **Pipelining**. Using this technique we can perform all rounds in Step 4 in just  $\mathcal{O}(\log k + \log n)$  many parallel steps.