algQ

Algorithm Queue

◇ *Local*

- **Node* leaf: pointer the the process's leaf in the tree

◇ *Shared*

- *Tree* to complete

◇ *Structures*

► *Node*

- **Node* left, right, parent
- *Block[]* blocks: index 0 contains an empty block with all fields equal to 0 and en pointers to the first block of the coresponding children. blocks[i] returns the ith block stored.
- *int* head= 1: index of the first empty cell of blocks
- *int* counter= 0
- *int[]* super: super[i] stores the index of a superblock in parent that contains some block of this node whose time is field i

► *Block*

- *int* $\text{num}_{\text{enq-left}}$, $\text{sum}_{\text{enq-left}}$: #enqueues from subblocks in left child, prefix sum of $\text{num}_{\text{enq-left}}$
- *int* $\text{num}_{\text{deq-left}}$, $\text{sum}_{\text{deq-left}}$: #dequeues from subblocks in left child, prefix sum of $\text{num}_{\text{deq-left}}$
- *int* $\text{num}_{\text{enq-right}}$, $\text{sum}_{\text{enq-right}}$: #enqueues from subblocks in right child, prefix sum of $\text{num}_{\text{enq-right}}$
- *int* $\text{num}_{\text{deq-right}}$, $\text{sum}_{\text{deq-right}}$: #dequeues from subblocks in right child, prefix sum of $\text{num}_{\text{deq-right}}$
- *int* num_{enq}, num_{deq} : # enqueue, dequeue operations in the block
- *int* sum_{enq}, sum_{deq} : sum of # enqueue, dequeue operations in blocks up to this one
- *int* num, sum : total # operations in block, prefix sum of num
- *int* end_{left}, $\text{end}_{\text{right}}$: index of the last subblock in the left and right child
- *int* group : id of the group of blocks including this propagated together, more precisely the value of the node's counter when propagating this block.

► *Leaf Block* extends *Block*

- *Object* element Each block in a leaf also represents an operation. The element shows the operations argument if it is an enqueue, and if it is a dequeue the value is null.

► *Root Block* extends *Block*

- *int* size size of queue after this block's operations finish
- *int* $\text{sum}_{\text{non-null deq}}$ count of non-null dequeus up to this block

```
1:  void ENQUEUE(Object e)
2:      block b= NEW(block)
3:      b.element= e
4:      b.sum_enq=1
5:      APPEND(b)
6:  end ENQUEUE

7:  Object DEQUEUE()
8:      block b= NEW(block)
9:      b.element= null
10:     b.sum_deq=1
11:     APPEND(b)
12:     <i, b>= INDEX(l_pid, op.loc, 1)
13:     res= COMPUTEHEAD(i, b)        ▷ Index of the enqueue whose argument
        should be returned
14:     return GET(res)
15: end DEQUEUE

16: int COMPUTEHEAD(int i, int b)     ▷ Computes head of the queue when
    ith dequeue in bth block occurs. The dequeue should return the argument
    of the head enqueue.
17:     if root.blocks[b-1].size + root.blocks[b].num_enq - i < 0 then
18:         return -1
19:     else return root.blocks[b-1].sum_non-null deq + i
20:     end if
21: end COMPUTEHEAD

22: void APPEND(block b)
23:     b.group= this.leaf.head
24:     l_pid.blocks[this.leaf.head]= b
25:     this.leaf.head+=1
26:     PROPAGATE(this.leaf.parent)
27: end APPEND
```