Kuncheng Feng
CSC 416
Solution document for :-
Programming Challenge: Missionaries and Cannibals SSPS

Learning abstract:

This challenge helps to deepen the understanding of object-oriented features of Lisp, by solving the classic “missionaries and cannibals” problem with a state space approach. This challenge also provides experience in simple breadth first search of nodes, as well as stepwise refinement of similar and redacted codes.

Source Code

For better organization, the codes have been organized into 5 files:

- mc_bank.l
- mc_node.l
- mc_operator.l
- mc_ssps.l
- mc_state.l

Note: Due to my learning process, some small alterations to variable names have been made, but all the structures and functions remain the same.

mc_bank.l

```
; File: mc_bank.l
;---------------------------------------------------------------------------
;
; MODELING A BANK

(defclass bank()
        (
                (missionaries :accessor bank-missionaries :initarg :missionaries)
                (cannibals :accessor bank-cannibals :initarg :cannibals)
                (boat :accessor bank-boat :initarg :boat)
        )
)

(defmethod display((b bank))
        (format t "missionaries: ~A    " (bank-missionaries b))
        (format t "cannibals: ~A    " (bank-cannibals b))
        (format t "boat: ~A~%" (bank-boat b))
        nil
)

(defmethod empty-bank((b bank))
        (and
                (equal (bank-missionaries b) nil)
                (equal (bank-cannibals b) nil)
```

```
                (equal (bank-boat b) nil)
        )
)

(defmethod full-bank((b bank))
        (and
                (equal (bank-missionaries b) '(m m m))
                (equal (bank-cannibals b) '(c c c))
                (equal (bank-boat b) 'b)
        )
)

(defmethod feast-bank((b bank) &aux mNum cNum)
        (setf mNum (length (bank-missionaries b)))
        (setf cNum (length (bank-cannibals b)))
        (and
                (> mNum 0)
                (> cNum mNum)
        )
)

(defmethod equal-bank((this bank) (other bank))
        (and
                (equal (length (bank-missionaries this)) (length (bank-missionaries other)))
                (equal (length (bank-cannibals this)) (length (bank-cannibals other)))
                (equal (bank-boat this) (bank-boat other))
        )
)

(defmethod copy-bank((b bank))
        (make-instance 'bank
                :missionaries (bank-missionaries b)
                :cannibals (bank-cannibals b)
                :boat (bank-boat b)
        )
)

(defmethod has-boat((b bank))
        (equal (bank-boat b) 'b)
)

(defmethod add-missionaries((b bank) missionaries)
        (setf (bank-missionaries b) (append (bank-missionaries b) missionaries))
)

(defmethod add-cannibals((b bank) cannibals)
        (setf (bank-cannibals b) (append (bank-cannibals b) cannibals))
)

(defmethod add-boat((b bank))
        (setf (bank-boat b) 'b)
)

(defmethod remove-missionaries((b bank) missionaries)
        (setf (bank-missionaries b) (remove 'm (bank-missionaries b) :count (length
missionaries)))
)

(defmethod remove-cannibals((b bank) cannibals)
        (setf (bank-cannibals b) (remove 'c (bank-cannibals b) :count (length cannibals)))
)

(defmethod remove-boat((b bank))
        (setf (bank-boat b) nil)
)
```

mc_state.l

```
; File: mc_state.l
;---------------------------------------------------------------------
;
; MODELING A STATE

(defclass state()
	(
		(left-bank :accessor state-left-bank :initarg :left-bank)
		(right-bank :accessor state-right-bank :initarg :right-bank)
	)
)

(defmethod display ((s state))
	(format t "Current bank: ~A~%" (state-current-bank s))

	(format t "Left  bank: ")
	(display (state-left-bank s))

	(format t "Right bank: ")
	(display (state-right-bank s))

	nil
)

(defmethod equal-state ((this state) (other state))
	(and
		(equal-bank (state-left-bank this) (state-left-bank other))
		(equal-bank (state-right-bank this) (state-right-bank other))
	)
)

(defmethod goalp((s state))
	(and
		(empty-bank (state-left-bank s))
		(full-bank (state-right-bank s))
	)
)

(defmethod feast-state-p((s state))
	(or
		(feast-bank (state-left-bank s))
		(feast-bank (state-right-bank s))
	)
)

(defmethod exploredp((s state) explored-list)
;	### best to use member with two keyword args -- :key and :test
	(member s explored-list :test #'equal-state :key #'node-state)
)

; Returns either 'left or 'right
(defmethod state-current-bank((s state))
	(if (has-boat (state-left-bank s))
		'left
		'right
	)
)

; Return a new state instance with the same values
(defmethod copy-state((s state))
	(make-instance 'state
		:left-bank (copy-bank (state-left-bank s))
		:right-bank (copy-bank (state-right-bank s))
	)
)

; Check if the current state supports the upcoming operation
```

```
(defmethod applicable-state((s state) crew &aux this-bank)
	(if (equal (state-current-bank s) 'left)
		(setf this-bank (state-left-bank s))
		(setf this-bank (state-right-bank s))
	)

	(and
		(<= (count 'm crew) (length (bank-missionaries this-bank)))
		(<= (count 'c crew) (length (bank-cannibals this-bank)))
	)
)

; Move the crews into the other side of the bank, the boat as well
(defmethod move-state((s state) crew &aux this-bank other-bank m_List c_List)
	; Define variables for easier access
	(cond
		((equal (state-current-bank s) 'left)
			(setf this-bank (state-left-bank s))
			(setf other-bank (state-right-bank s))
		)
		(t
			(setf this-bank (state-right-bank s))
			(setf other-bank (state-left-bank s))
		)
	)
	(setf m_List (remove 'c crew))
	(setf c_List (remove 'm crew))

	(remove-missionaries this-bank m_List)
	(add-missionaries other-bank m_List)

	(remove-cannibals this-bank c_List)
	(add-cannibals other-bank c_List)

	(remove-boat this-bank)
	(add-boat other-bank)
)
```

mc_operator.l

```
; File: mc_operator.l
;---------------------------------------------------------------------------
; MODELING A STATE SPACE OPERATOR
(defclass operator ()
	(
		(name :accessor operator-name :initarg :name)
		(precondition :accessor operator-precondition :initarg :precondition)
		(description :accessor operator-description :initarg :description)
	)
)

(defmethod display((op operator))
	(format t "Operator name: ~A~% " (operator-name op))
	(format t "Precondition: ~A~% " (operator-precondition op))
	(format t "Description: ~A~%" (operator-description op))
)

; Total of 5 operators:
; *move-c* *move-c-c* *move-m* *move-m-m* *move-c-m*
; They will be concatenated into *operator-list*
(defmethod establish-operators()
	(setf *move-c*
		(make-instance 'operator
			:name 'move-c
			:precondition "Current bank has at least 1 cannibal and a boat."
```

```
                        :description "Move (b c) to the other bank."
                )
        )
        (setf *move-c-c*
                (make-instance 'operator
                        :name 'move-c-c
                        :precondition "Current bank has at least 2 cannibals and a boat."
                        :description "Move (b c c) to the other bank."
                )
        )
        (setf *move-m*
                (make-instance 'operator
                        :name 'move-m
                        :precondition "Current bank has at least 1 missionary and a boat."
                        :description "Move (b m) to the other bank."
                )
        )
        (setf *move-m-m*
                (make-instance 'operator
                        :name 'move-m-m
                        :precondition "Current bank has at least 2 missionaries and a boat."
                        :description "Move (b m m) to the other bank."
                )
        )
        (setf *move-c-m*
                (make-instance 'operator
                        :name 'move-c-m
                        :precondition "Current bank has at least 1 cannibal, at least 1
missionary, and a boat."
                        :description "Move (b c m) to the other bank."
                )
        )
        (setf *operator-list*
                (list *move-c* *move-c-c* *move-m* *move-m-m* *move-c-m*)
        )
        nil
)


;
-------------------------------------------------------------------------------------------------
; Check for applicability

; Check if the state satisfies the precondition of the operator
(defmethod applicable-operator((o operator) (s state) &aux name)
        (setf name (operator-name o))

        (cond
                ((equal name 'move-c)
                        (applicable-state s '(c))
                )
                ((equal name 'move-c-c)
                        (applicable-state s '(c c))
                )
                ((equal name 'move-m)
                        (applicable-state s '(m))
                )
                ((equal name 'move-m-m)
                        (applicable-state s '(m m))
                )
                ((equal name 'move-c-m)
                        (applicable-state s '(c m))
                )
                (t
                        (format t "ERROR! Invalid operator! ~%")
                )
        )
)
```

```
;
-------------------------------------------------------------------------------------------
; Applying the operators

; Return a new state that has been operated on.
(defmethod apply-operator((o operator) (s state) &aux name this-bank other-bank)
	; Define variables for easier access
	(setf name (operator-name o))

	; Match the operator name and apply
	(cond
		((equal name 'move-c)
			(move-state s '(c))
		)
		((equal name 'move-c-c)
			(move-state s '(c c))
		)
		((equal name 'move-m)
			(move-state s '(m))
		)
		((equal name 'move-m-m)
			(move-state s '(m m))
		)
		((equal name 'move-c-m)
			(move-state s '(c m))
		)
		(t
			(format t "ERROR! Invalid operator! ~%")
		)
	)
)
```

mc_node.l

```
; File: mc_node.l
;---------------------------------------------------------------------
;
; MODELING A NODE

(defclass node()
	(
		(name :accessor node-name :initarg :name)
		(state :accessor node-state :initarg :state)
		(parent :accessor node-parent :initarg :parent)
		(operator :accessor node-operator :initarg :operator)
	)
)

; Altered slightly
(defmethod display ((n node))
	(format t "~A " (node-name n))
	(if (not (rootp n))
		(let ()
			(format t "~A " (node-name (node-parent n)))
			(format t "~A " (operator-name (node-operator n)))
		)
	)
	(terpri)
	(display (node-state n))
	nil
)

(defmethod rootp((n node))
	(equal (node-name n) 'root)
)
```

mc_ssps.l

```
; File: mc_ssps.l
;
;---------------------------------------------------------------------
; Description:
;
; This program is a state space problem solver for a classic missionaries and cannibls
problem.
; A state space tree is grown in concert with breadth first search for a solution
;
; ---------------------------------------------------------------------
; REPRESENTATIONAL NOTES
;
; Banks are represented as a 3-slot class consisting of
; missionaries, cannibals, and a boat.
;
; States are represented as a 2-slot class consisting of
; left-bank (object), right-bank (object).
;
; Operators are represented as a 3-slot class consisting of
; a name, a precondition, and a description.
;
; Nodes are represented as a 4-slot class consisting of
; a name, a state, a parent node, and a move (state space operator)


(load "mc_bank.l")
(load "mc_state.l")
(load "mc_node.l")
(load "mc_operator.l")

;-------------------------------------------------------------------------------
; THE MAIN PROGRAM - argument values of e u x eu ex ux eux will cause tracing
(defmethod mc ((trace-instruction symbol))
        (setf *trace-instruction* trace-instruction)
        (establish-operators)
        (setup)
        (solve)
)


;-------------------------------------------------------------------------------
; SOLVE PERFORMS BREADTH FIRST SEARCH
; Exploredp is modified slightly
(defmethod solve (&aux kids e-node)
        (if (member *trace-instruction* '(u eu ux eux)) (display-the-unexplored-list))
        (if (member *trace-instruction* '(x ex ux eux)) (display-the-explored-list))
        (cond
                ((null *unexplored*)
                        (format t ">>> THERE IS NO SOLUTION.~%")
                        (return-from solve NIL)
                )
        )

        (setf e-node (pop *unexplored*))

        (if (member *trace-instruction* '(e ex eu eux)) (display-the-e-node e-node))

        (cond
                ((goalp (node-state e-node))
                        (format t "~%>>> GOT A SOLUTION!")
                        (display-solution e-node)
                )
                ((feast-state-p (node-state e-node))
                        (solve)
                )
                ((exploredp (node-state e-node) *explored*)
                        (solve)
```

```
                )
                (t
                        (push e-node *explored*)
                        (setf kids (children-of e-node))
                        (setf *unexplored* (append *unexplored* kids))
                        (solve)
                )
        )
        nil
)

(defmethod display-the-unexplored-list()
        (format t "~%>>> Unexplored list~%")
        (mapcar #'display *unexplored*)
        nil
)

(defmethod display-the-explored-list()
        (format t "~%>>> Explored list~%")
        (mapcar #'display *explored*)
        nil
)

(defmethod display-the-e-node((n node))
        (format t "~%>>> E-node~%")
        (display n)
)

(defmethod display-solution((n node))
        (cond
                ((rootp n)
                        (terpri)
                )
                (t
                        (display-solution (node-parent n))
                        (format t "~A~%" (operator-description (node-operator n)))
                )
        )
        nil
)

;---------------------------------------------------------------------
; THE SETUP
(defmethod setup (&aux root lb rb istate)
        ;; establish root node
        (setf lb (make-instance 'bank :missionaries '(m m m) :cannibals '(c c c) :boat 'b))
        (setf rb (make-instance 'bank :missionaries '() :cannibals '() :boat nil))
        (setf istate (make-instance 'state :left-bank lb :right-bank rb))
        (setf root (make-instance 'node :state istate :name 'root))

        ;; initialize list of unexplored nodes
        (setf *unexplored* (list root))

        ;; initialize list of explored nodes
        (setf *explored* ())

        ; get ready to create good names
        (setf *ng* (make-instance 'name-generator :prefix "N"))
)


;---------------------------------------------------------------------
; GENERATING CHILDREN
(defmethod children-of ((n node) &aux kids e-state)
        (setf kids (list))
        (setf e-state (node-state n))

        (if (applicable-operator *move-c* e-state)
                (push (child-of n *move-c*) kids)
        )
```

```
        (if (applicable-operator *move-c-c* e-state)
                (push (child-of n *move-c-c*) kids)
        )
        (if (applicable-operator *move-m* e-state)
                (push (child-of n *move-m*) kids)
        )
        (if (applicable-operator *move-m-m* e-state)
                (push (child-of n *move-m-m*) kids)
        )
        (if (applicable-operator *move-c-m* e-state)
                (push (child-of n *move-c-m*) kids)
        )

        kids
)

; Since the new state is an instance, it's values are modified inside "apply-operator"
(defmethod child-of ((n node) (o operator) &aux new-state)
        (setf new-node (make-instance 'node))
        (setf (node-name new-node) (next *ng*))
        (setf (node-parent new-node) n)
        (setf (node-operator new-node) o)
        (setf new-state (copy-state (node-state n)))
        (apply-operator o new-state)
        (setf (node-state new-node) new-state)
        new-node
)


;-----------------------------------------------------------------------
; MODELLING A NAME-GENERATOR
(defclass name-generator ()
        (
                (prefix :accessor name-generator-prefix :initarg :prefix :initform "name")
                (nr :accessor name-generator-nr :initform 0)
        )
)

(defmethod next ((ng name-generator))
        (setf (name-generator-nr ng) (+ 1 (name-generator-nr ng)))
        (concatenate 'string
                (name-generator-prefix ng)
                (write-to-string (name-generator-nr ng))
        )
)
```

“Quiet” Demo

```
[]> (mc nil)

>>> GOT A SOLUTION!
Move (b c m) to the other bank.
Move (b m) to the other bank.
Move (b c c) to the other bank.
Move (b c) to the other bank.
Move (b m m) to the other bank.
Move (b c m) to the other bank.
Move (b m m) to the other bank.
Move (b c) to the other bank.
Move (b c c) to the other bank.
Move (b m) to the other bank.
Move (b c m) to the other bank.
NIL
```

“Expand / Explore” Node

Demo

```
[]> (mc 'e)

>>> E-node
ROOT
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> E-node
N5 ROOT MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C)    boat: B

>>> E-node
N4 ROOT MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C C C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: NIL    boat: B

>>> E-node
N3 ROOT MOVE-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: NIL    boat: B

>>> E-node
```

```
N2 ROOT MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C)    boat: B

>>> E-node
N1 ROOT MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C)    boat: B

>>> E-node
N8 N5 MOVE-C-M
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> E-node
N7 N5 MOVE-M
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: B
Right bank: missionaries: NIL    cannibals: (C)    boat: NIL

>>> E-node
N6 N5 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: (M)    cannibals: NIL    boat: NIL

>>> E-node
N10 N2 MOVE-C-C
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> E-node
N9 N2 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: B
Right bank: missionaries: NIL    cannibals: (C)    boat: NIL

>>> E-node
N11 N1 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> E-node
N16 N7 MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C C)    boat: B

>>> E-node
N15 N7 MOVE-M-M
Current bank: RIGHT
```

```
Left  bank: missionaries: (M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: (C)    boat: B

>>> E-node
N14 N7 MOVE-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C)    boat: B

>>> E-node
N13 N7 MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: NIL    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C C)    boat: B

>>> E-node
N12 N7 MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C)    boat: B

>>> E-node
N18 N13 MOVE-C-C
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: B
Right bank: missionaries: NIL    cannibals: (C)    boat: NIL

>>> E-node
N17 N13 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: B
Right bank: missionaries: NIL    cannibals: (C C)    boat: NIL

>>> E-node
N22 N17 MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: NIL    boat: NIL
Right bank: missionaries: (M)    cannibals: (C C C)    boat: B

>>> E-node
N21 N17 MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: (C C)    boat: B

>>> E-node
N20 N17 MOVE-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C C)    boat: B

>>> E-node
N19 N17 MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: NIL    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C C)    boat: B
```

```
>>> E-node
N27 N21 MOVE-C-M
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: B
Right bank: missionaries: (M)    cannibals: (C)    boat: NIL

>>> E-node
N26 N21 MOVE-M-M
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: B
Right bank: missionaries: NIL    cannibals: (C C)    boat: NIL

>>> E-node
N25 N21 MOVE-M
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C)    boat: B
Right bank: missionaries: (M)    cannibals: (C C)    boat: NIL

>>> E-node
N24 N21 MOVE-C-C
Current bank: LEFT
Left  bank: missionaries: (M)    cannibals: (C C C)    boat: B
Right bank: missionaries: (M M)    cannibals: NIL    boat: NIL

>>> E-node
N23 N21 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M)    cannibals: (C C)    boat: B
Right bank: missionaries: (M M)    cannibals: (C)    boat: NIL

>>> E-node
N32 N27 MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: (C C)    boat: B

>>> E-node
N31 N27 MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C)    boat: B

>>> E-node
N30 N27 MOVE-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: (C)    boat: B

>>> E-node
N29 N27 MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: NIL    boat: NIL
Right bank: missionaries: (M)    cannibals: (C C C)    boat: B

>>> E-node
```

```
N28 N27 MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C C)    boat: B

>>> E-node
N36 N31 MOVE-C-M
Current bank: LEFT
Left  bank: missionaries: (M)    cannibals: (C C C)    boat: B
Right bank: missionaries: (M M)    cannibals: NIL    boat: NIL

>>> E-node
N35 N31 MOVE-M-M
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: B
Right bank: missionaries: (M)    cannibals: (C)    boat: NIL

>>> E-node
N34 N31 MOVE-M
Current bank: LEFT
Left  bank: missionaries: (M)    cannibals: (C C)    boat: B
Right bank: missionaries: (M M)    cannibals: (C)    boat: NIL

>>> E-node
N33 N31 MOVE-C
Current bank: LEFT
Left  bank: missionaries: NIL    cannibals: (C C C)    boat: B
Right bank: missionaries: (M M M)    cannibals: NIL    boat: NIL

>>> E-node
N38 N33 MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C)    boat: B

>>> E-node
N37 N33 MOVE-C
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C)    boat: B

>>> E-node
N43 N38 MOVE-C-M
Current bank: LEFT
Left  bank: missionaries: (M)    cannibals: (C C)    boat: B
Right bank: missionaries: (M M)    cannibals: (C)    boat: NIL

>>> E-node
N42 N38 MOVE-M-M
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C)    boat: B
Right bank: missionaries: (M)    cannibals: (C C)    boat: NIL

>>> E-node
N41 N38 MOVE-M
Current bank: LEFT
```

```
Left  bank: missionaries: (M)    cannibals: (C)    boat: B
Right bank: missionaries: (M M)    cannibals: (C C)    boat: NIL

>>> E-node
N40 N38 MOVE-C-C
Current bank: LEFT
Left  bank: missionaries: NIL    cannibals: (C C C)    boat: B
Right bank: missionaries: (M M M)    cannibals: NIL    boat: NIL

>>> E-node
N39 N38 MOVE-C
Current bank: LEFT
Left  bank: missionaries: NIL    cannibals: (C C)    boat: B
Right bank: missionaries: (M M M)    cannibals: (C)    boat: NIL

>>> E-node
N46 N41 MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: NIL    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C C)    boat: B

>>> GOT A SOLUTION!
Move (b c m) to the other bank.
Move (b m) to the other bank.
Move (b c c) to the other bank.
Move (b c) to the other bank.
Move (b m m) to the other bank.
Move (b c m) to the other bank.
Move (b m m) to the other bank.
Move (b c) to the other bank.
Move (b c c) to the other bank.
Move (b m) to the other bank.
Move (b c m) to the other bank.
NIL
```

State Space Tree Representation

Node: root
Parent: nil
Operator: nil
Left bank: (m m m c c c b)
Right bank: ()

Root

Node: n5
Parent: root
Operator: move-c-m
Left bank: (m m c c)
Right bank: (m c b)

Root

↓

n5

Node: n4
Parent: root
Operator: move-m-m
Left bank: (m c c c)
Right bank: (m m b)

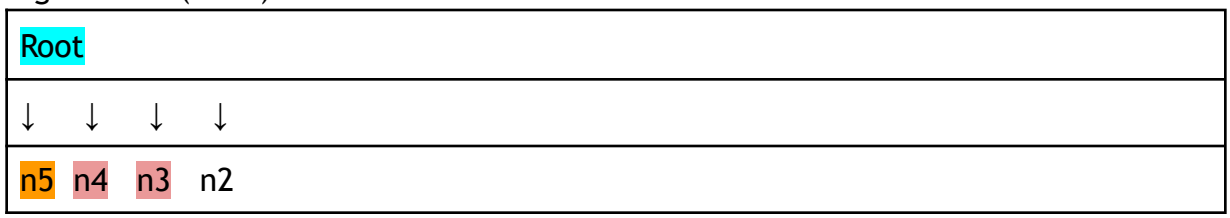
Root

↓ ↓

n5 n4

Node: n3
Parent: root
Operator: move-m
Left bank: (m m c c c)
Right bank: (m b)

Node: n2
Parent: root
Operator: move-c-c
Left bank: (m m m c)
Right bank: (c c b)



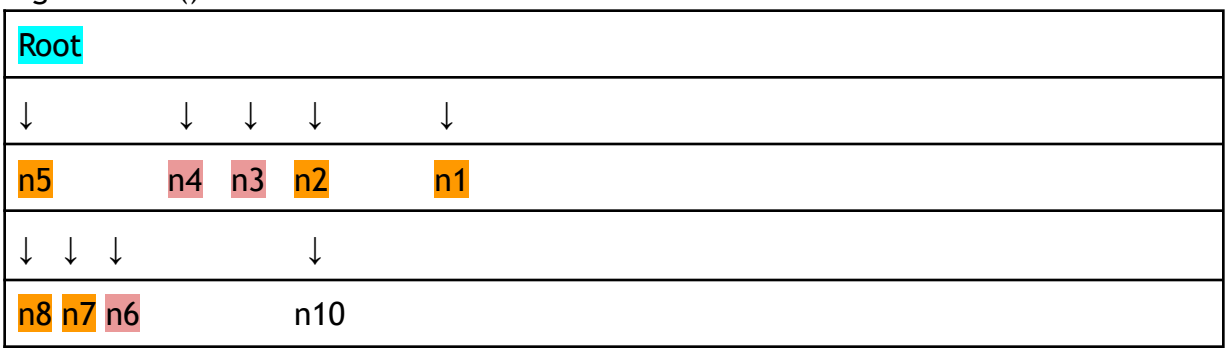
Node: n1
Parent: root
Operator: move-c
Left bank: (m m m c c)
Right bank: (c b)

Node: n8
Parent: n5
Operator: move-c-m
Left bank: (m m m c c c b)
Right bank: ()

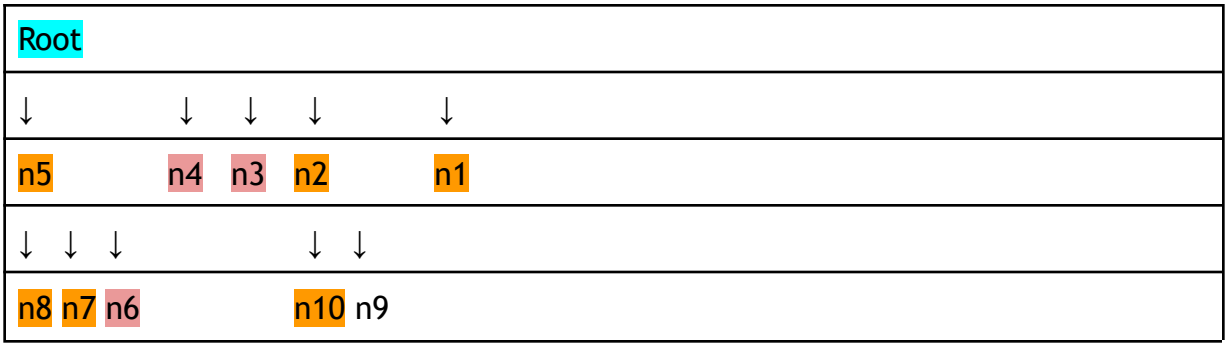
Node: n7
Parent: n5
Operator: move-m
Left bank: (m m m c c b)
Right bank: (c)

Node: n6
Parent: n5
Operator: move-c
Left bank: (m m c c c b)
Right bank: (m)

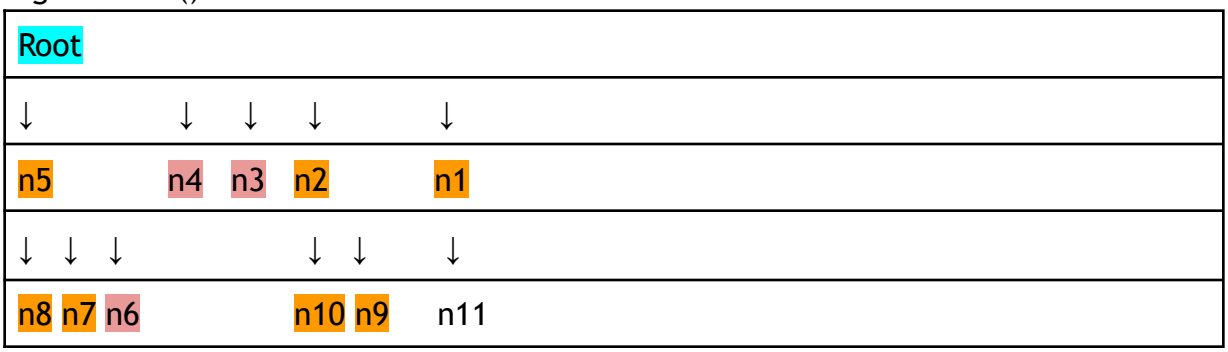
Node: n10
Parent: n2
Operator: move-c-c
Left bank: (m m m c c c b)
Right bank: ()



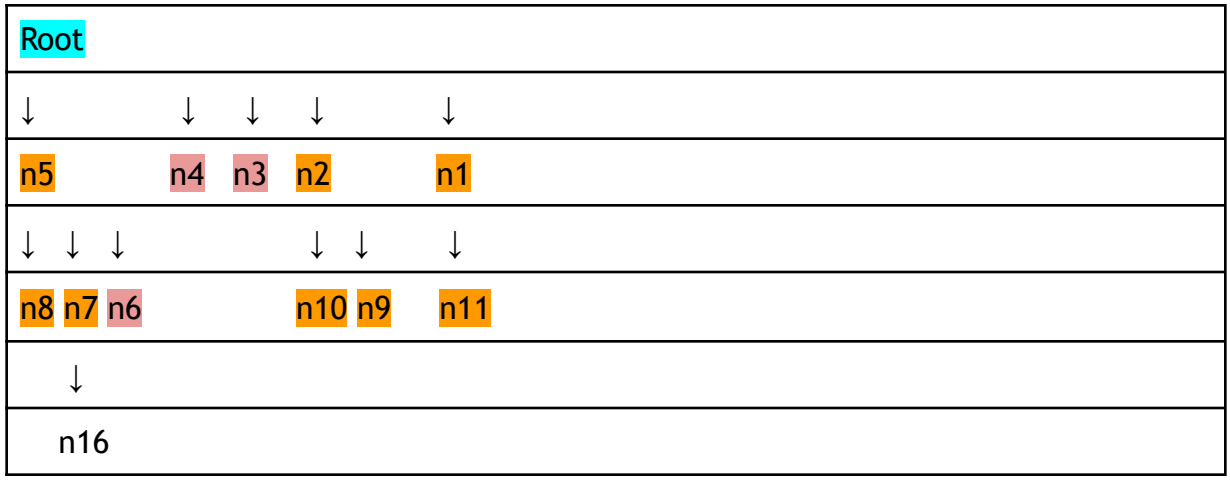
Node: n9
Parent: n2
Operator: move-c
Left bank: (m m m c c b)
Right bank: (c)



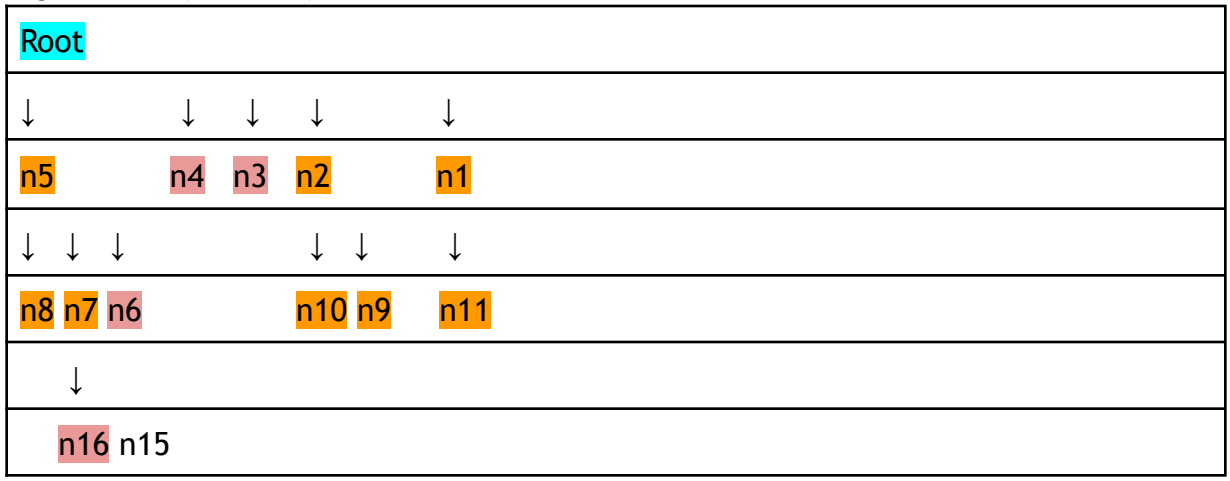
Node: n11
Parent: n1
Operator: move-c
Left bank: (m m m c c c b)
Right bank: ()



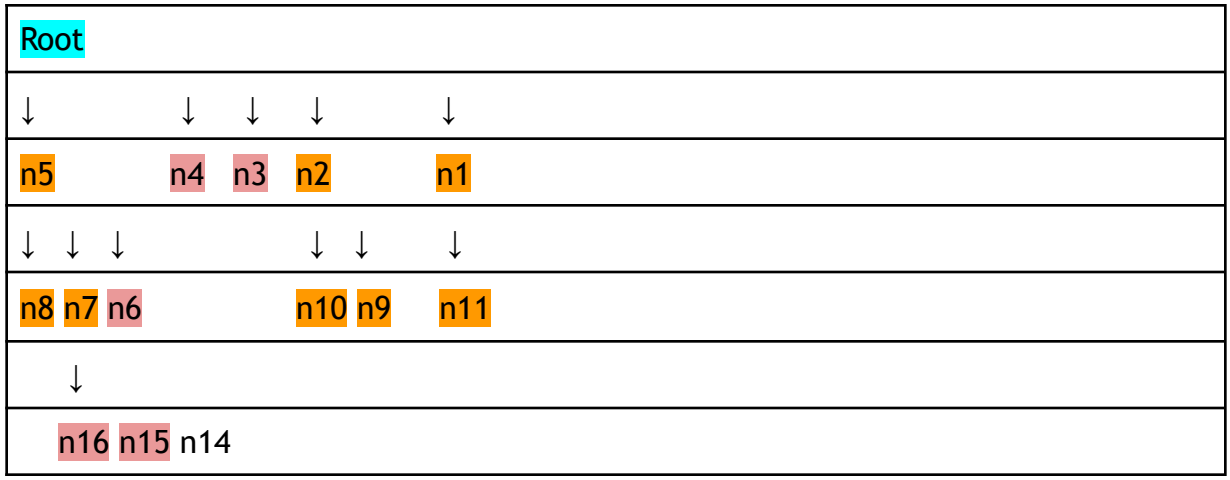
Node: n16
Parent: n7
Operator: move-c
Left bank: (m m c)
Right bank: (m c c b)



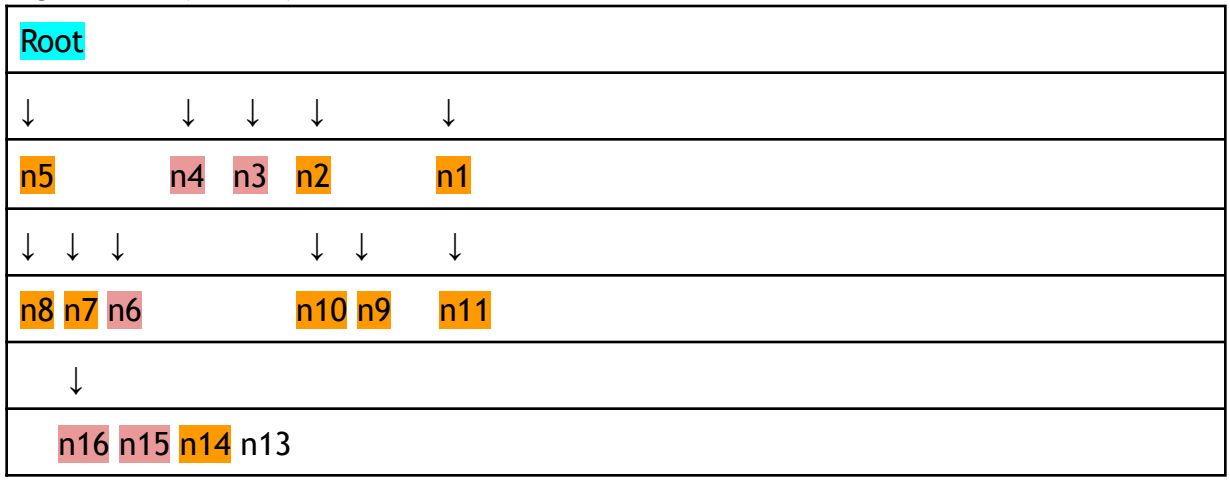
Node: n15
Parent: n7
Operator: move-m-m
Left bank: (m c c)
Right bank: (m m c b)



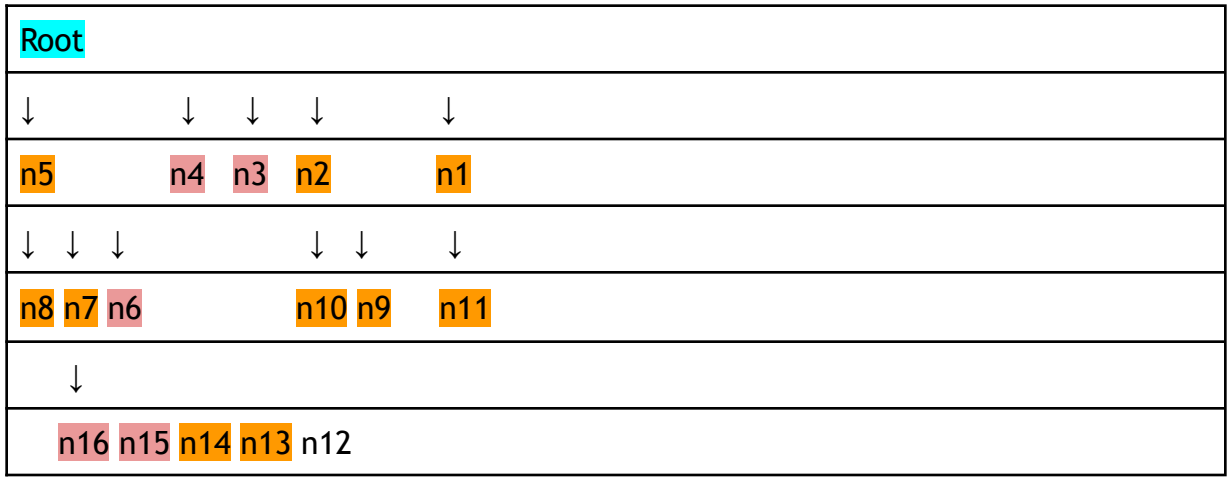
Node: n14
Parent: n7
Operator: move-m
Left bank: (m m c c)
Right bank: (m c b)



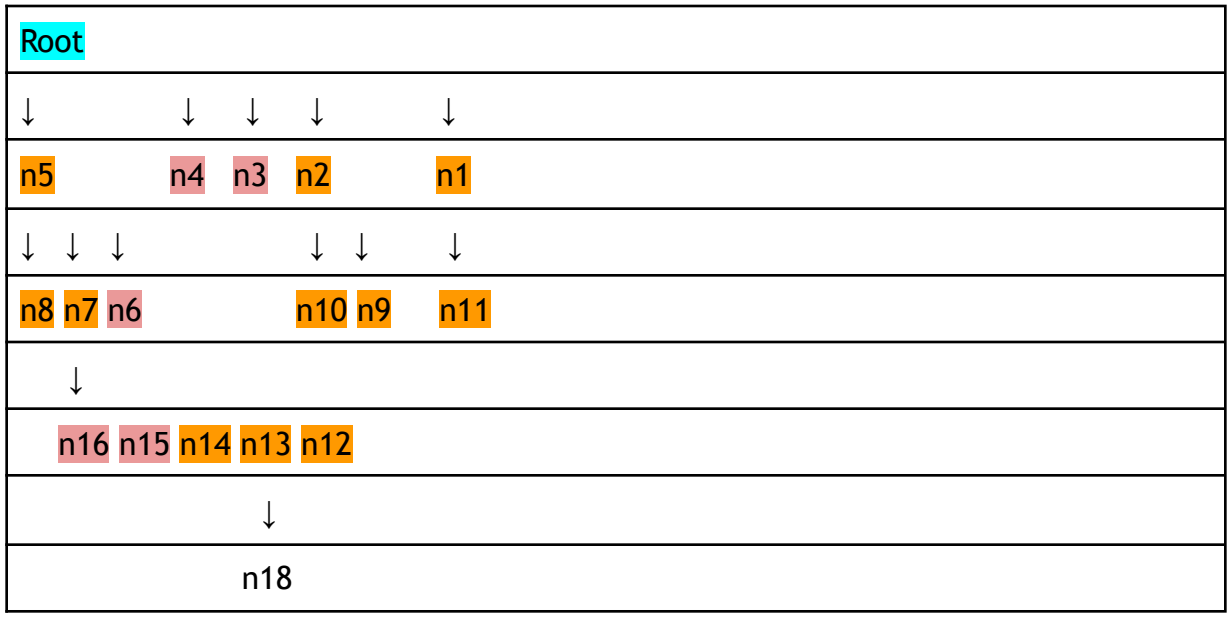
Node: n13
Parent: n7
Operator: move-c-c
Left bank: (m m m)
Right bank: (c c c b)



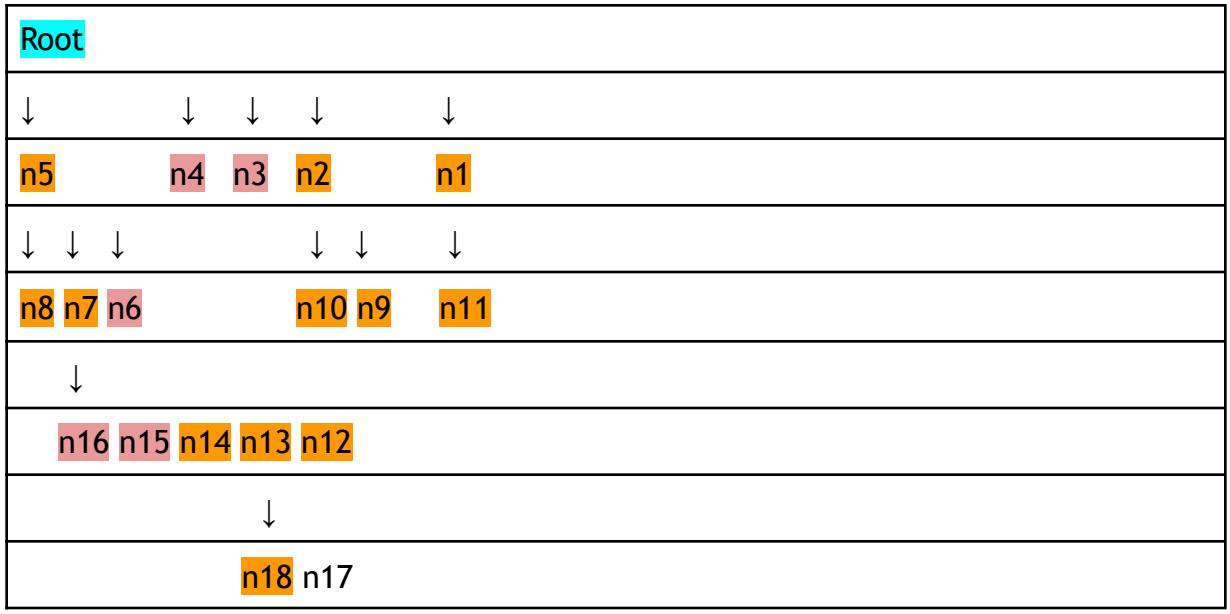
Node: n12
Parent: n7
Operator: move-c
Left bank: (m m m c)
Right bank: (c c b)



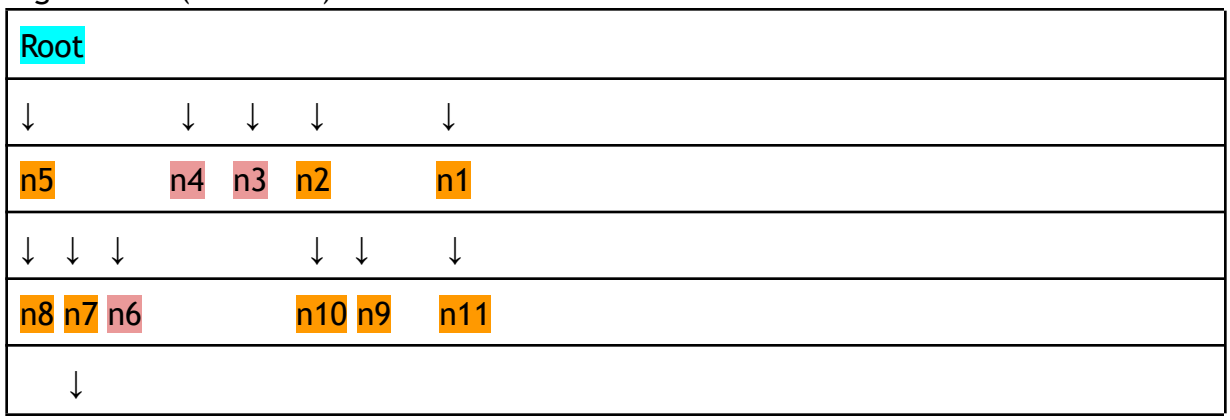
Node: n18
Parent: n13
Operator: move-c
Left bank: (m m m c c b)
Right bank: (c)



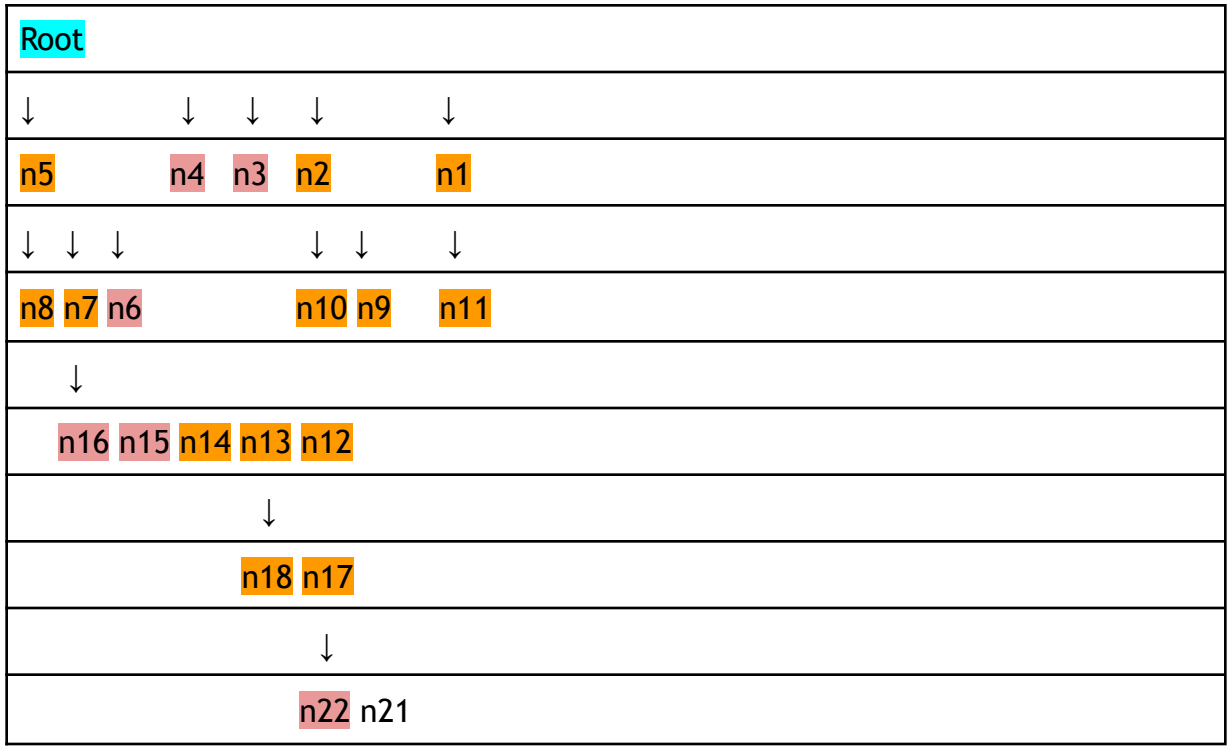
Node: n17
Parent: n13
Operator: move-c
Left bank: (m m m c b)
Right bank: (c c)



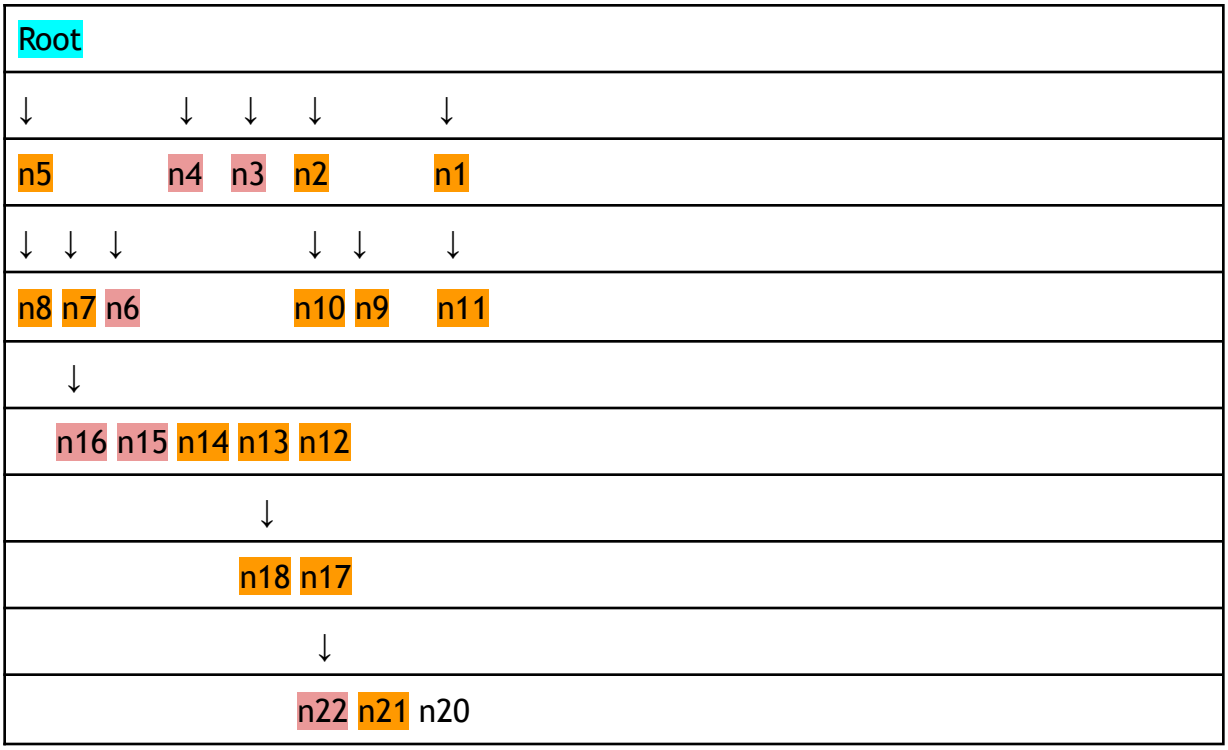
Node: n22
Parent: n17
Operator: move-c-m
Left bank: (m m)
Right bank: (m c c c b)



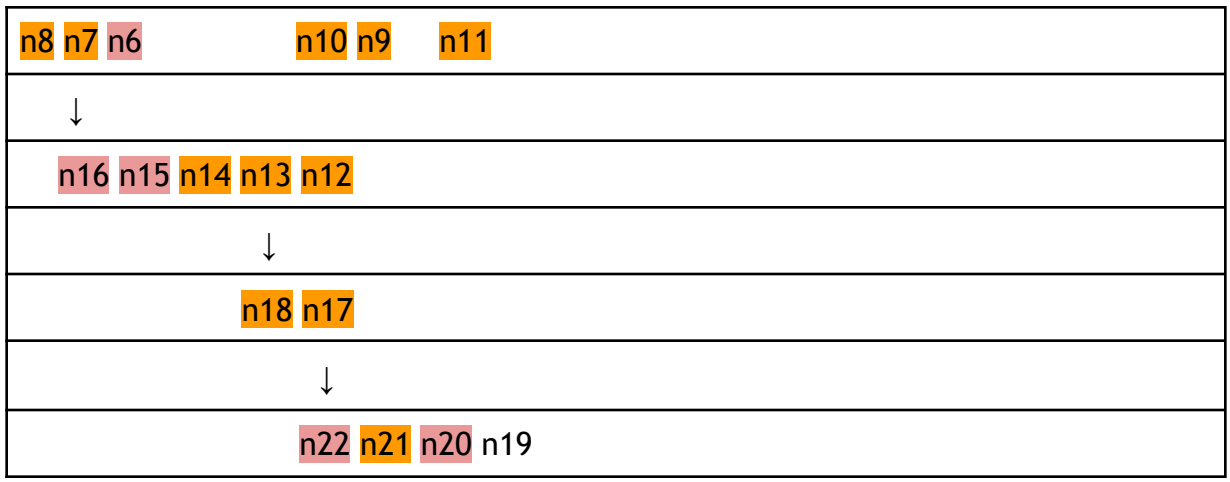
Node: n21
Parent: n17
Operator: move-c-m
Left bank: (m c)
Right bank: (m m c c b)



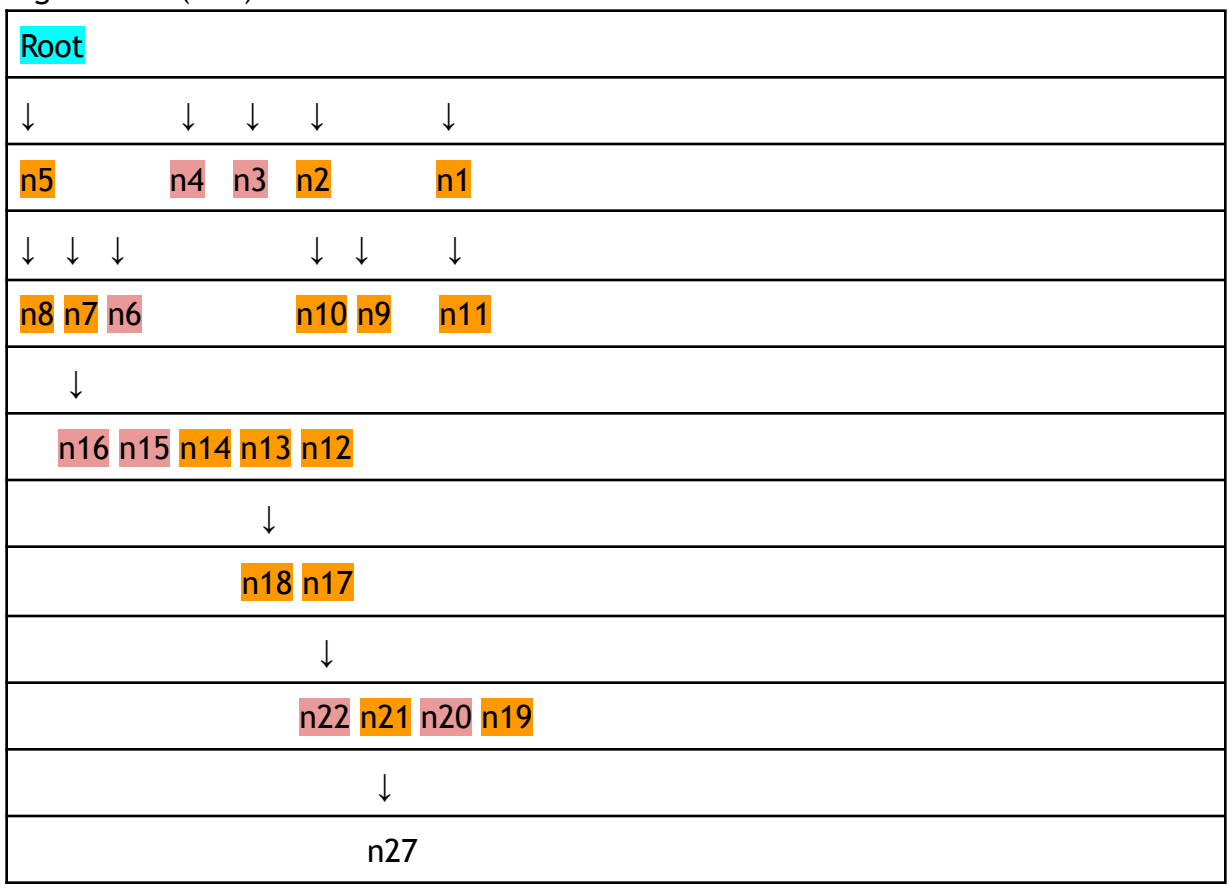
Node: n20
Parent: n17
Operator: move-m
Left bank: (m m c)
Right bank: (m c c b)



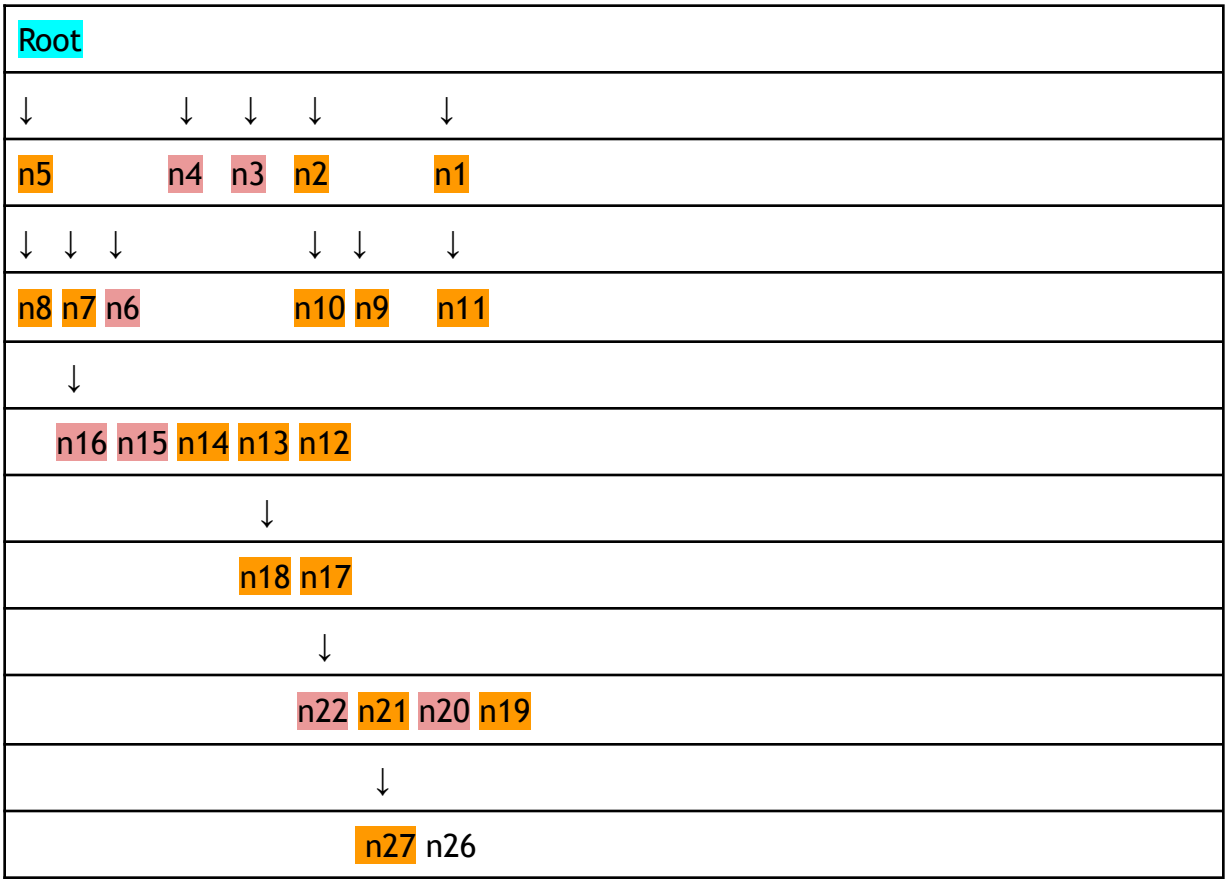
Node: n19
Parent: n17
Operator: move-m
Left bank: (m m m)
Right bank: (c c c b)



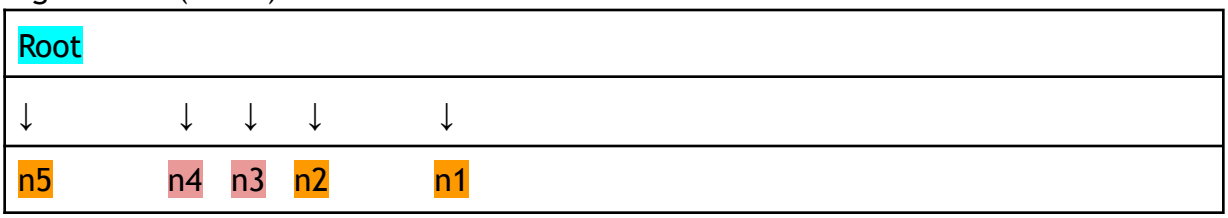
Node: n27
Parent: n21
Operator: move-c-m
Left bank: (m m c c b)
Right bank: (m c)

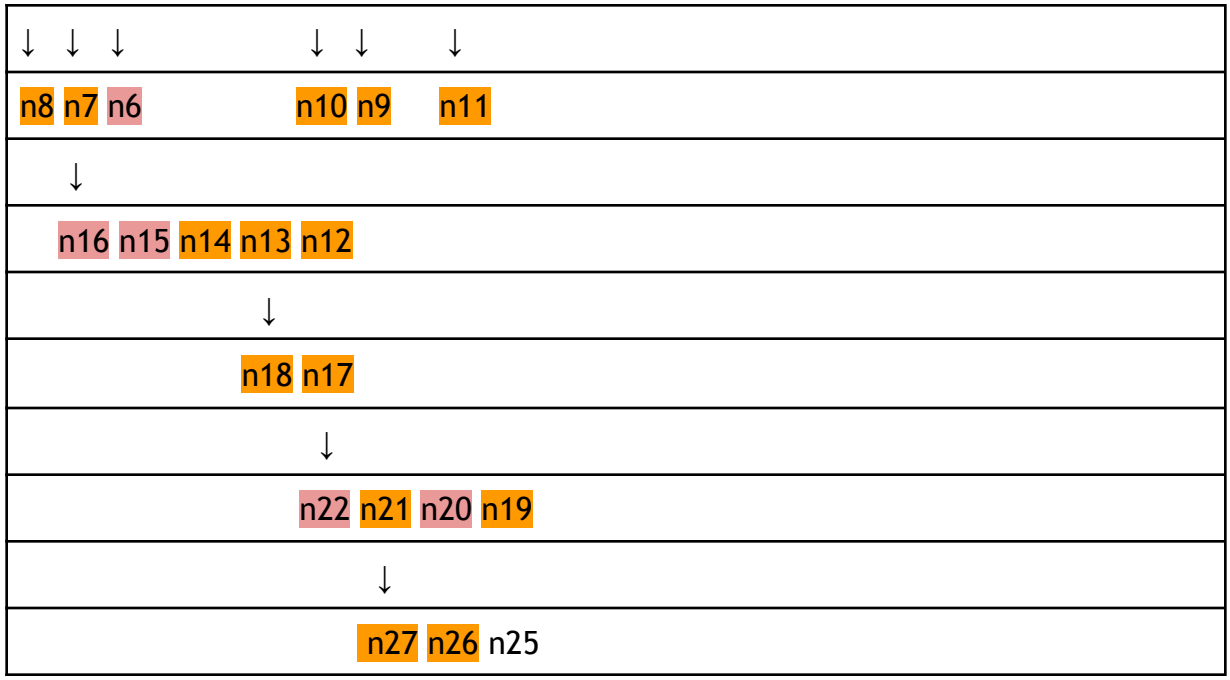


Node: n26
Parent: n21
Operator: move-m-m
Left bank: (m m m c b)
Right bank: (c c)

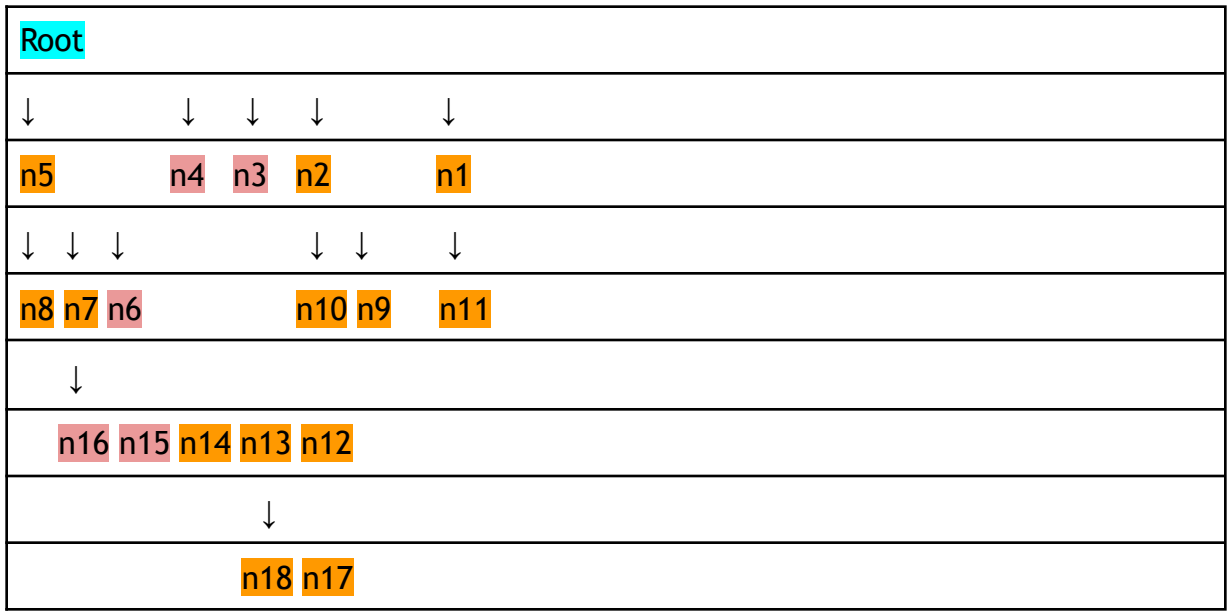


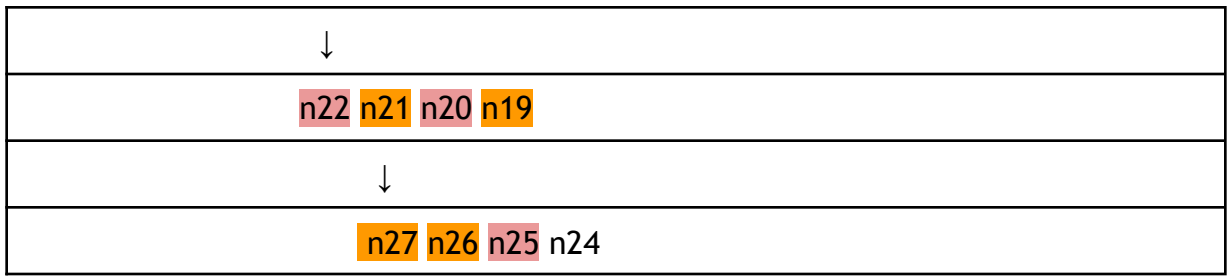
Node: n25
Parent: n21
Operator: move-m
Left bank: (m m c b)
Right bank: (m c c)



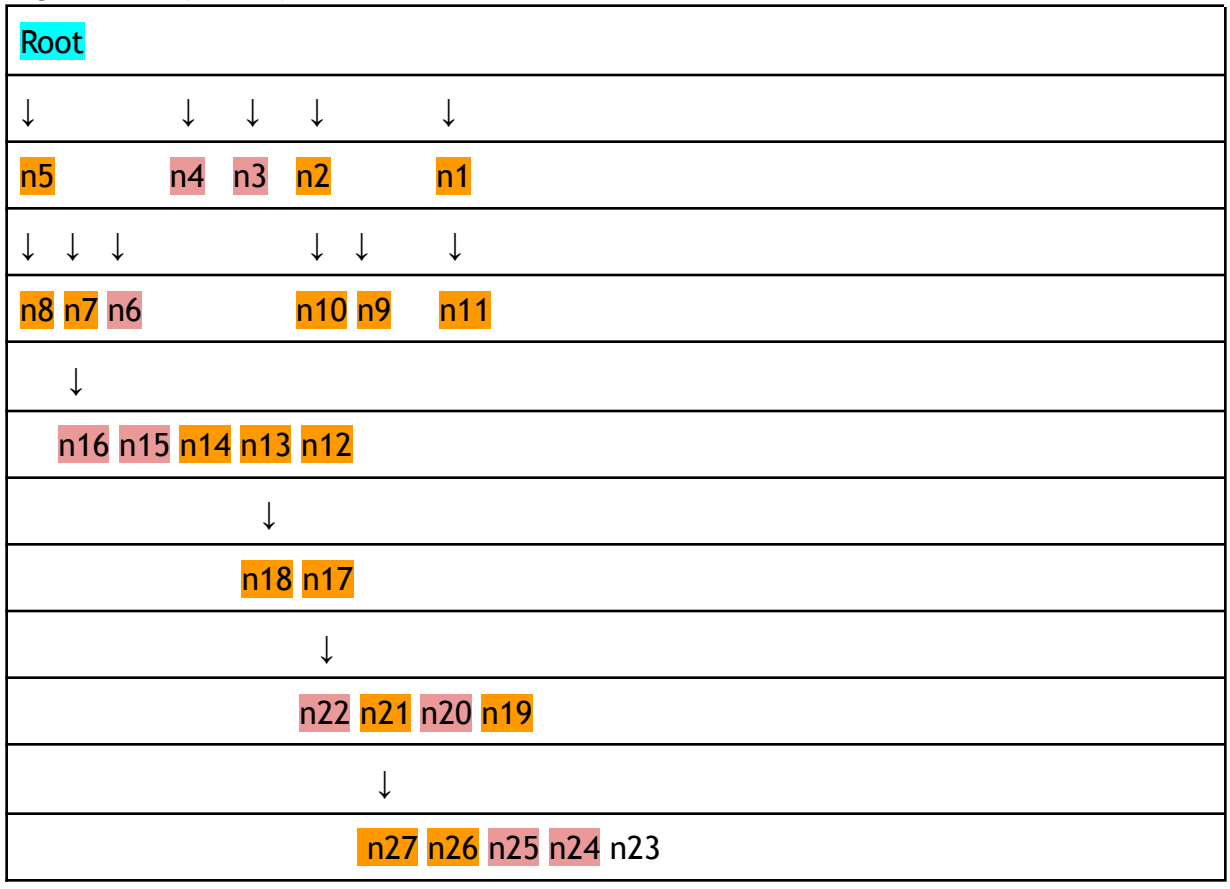


Node: n24
Parent: n21
Operator: move-c-c
Left bank: (m c c c b)
Right bank: (m m)



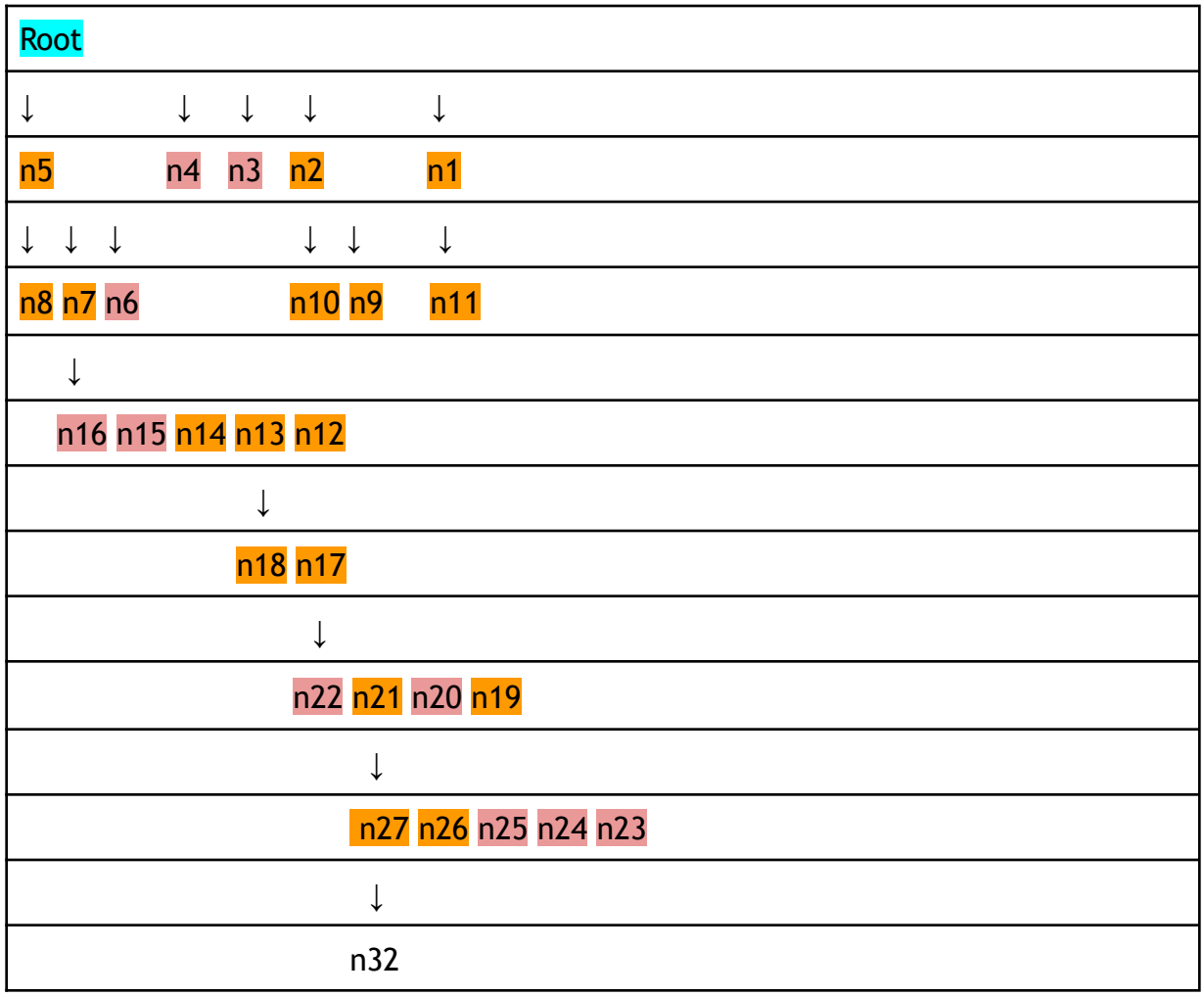


Node: n23
Parent: n21
Operator: move-c
Left bank: (m c c b)
Right bank: (m m c)

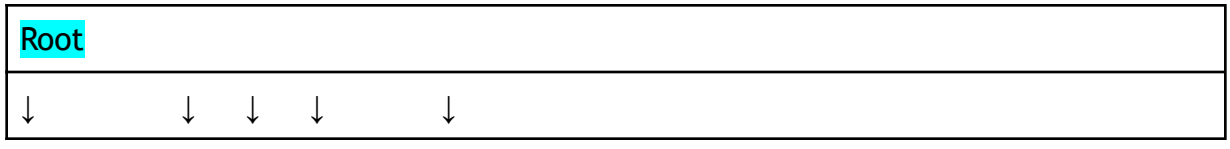


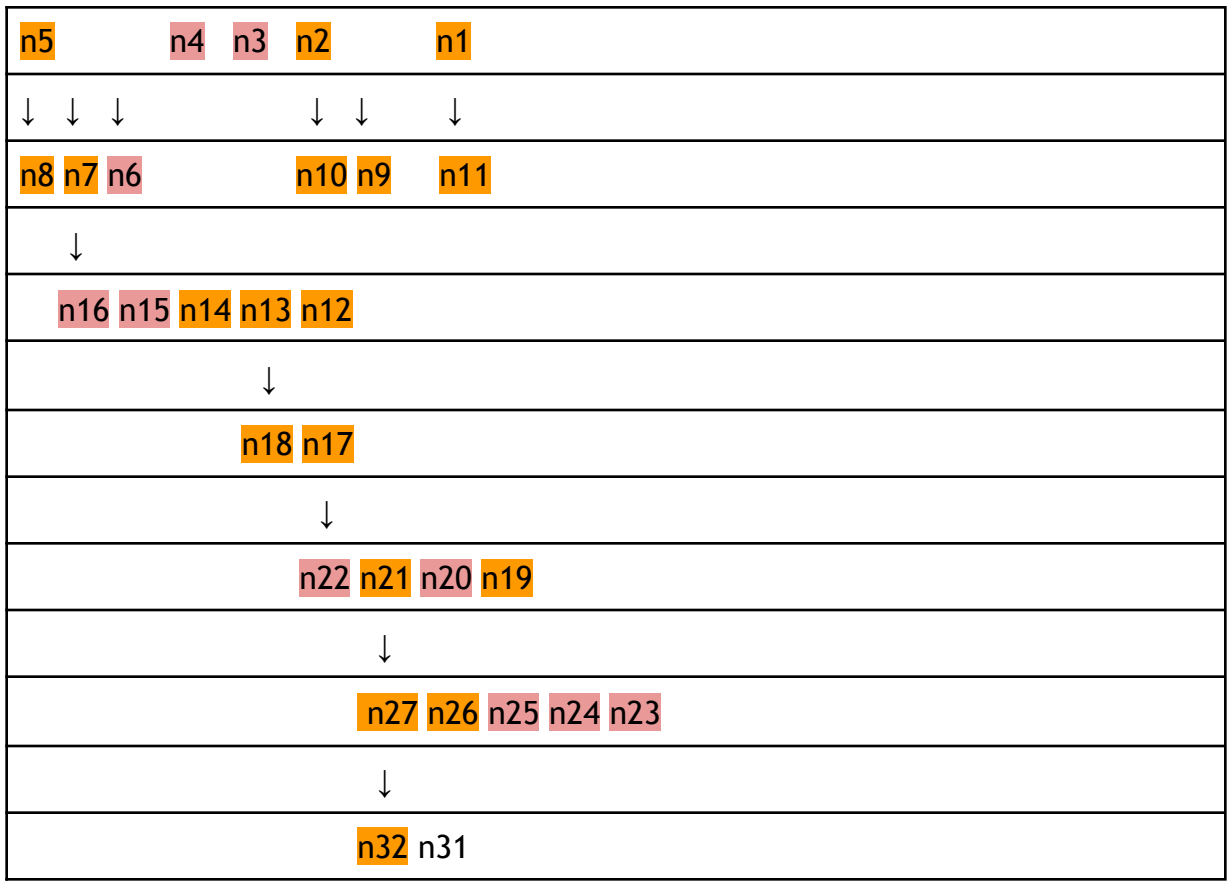
Node: n32

Parent: n27
Operator: move-c-m
Left bank: (m c)
Right bank: (m m c c b)

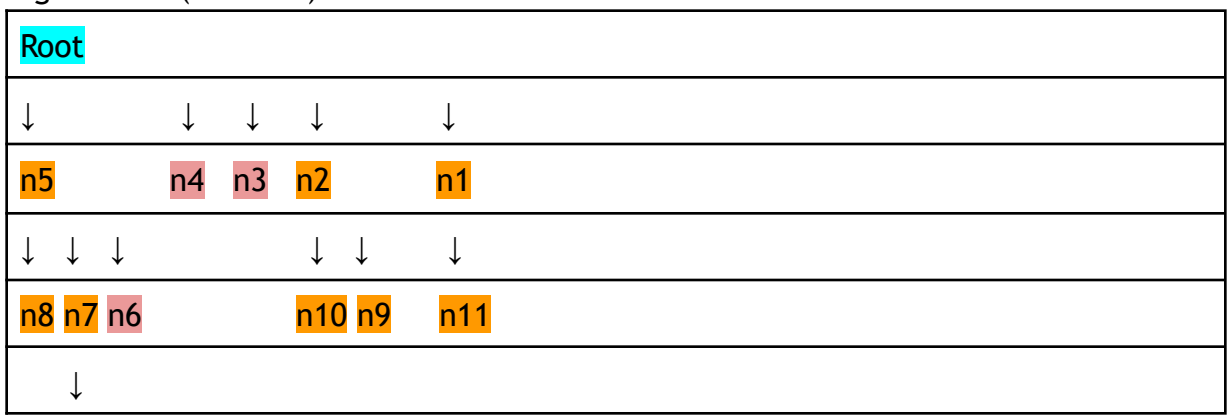


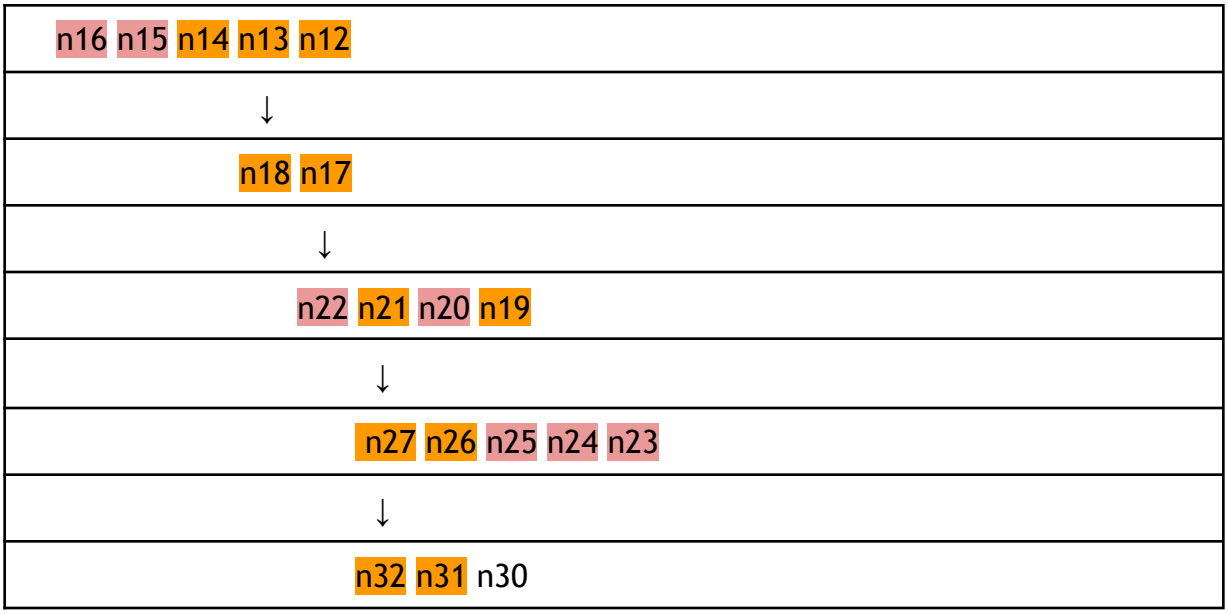
Node: n31
Parent: n27
Operator: move-m-m
Left bank: (c c)
Right bank: (m m m c b)



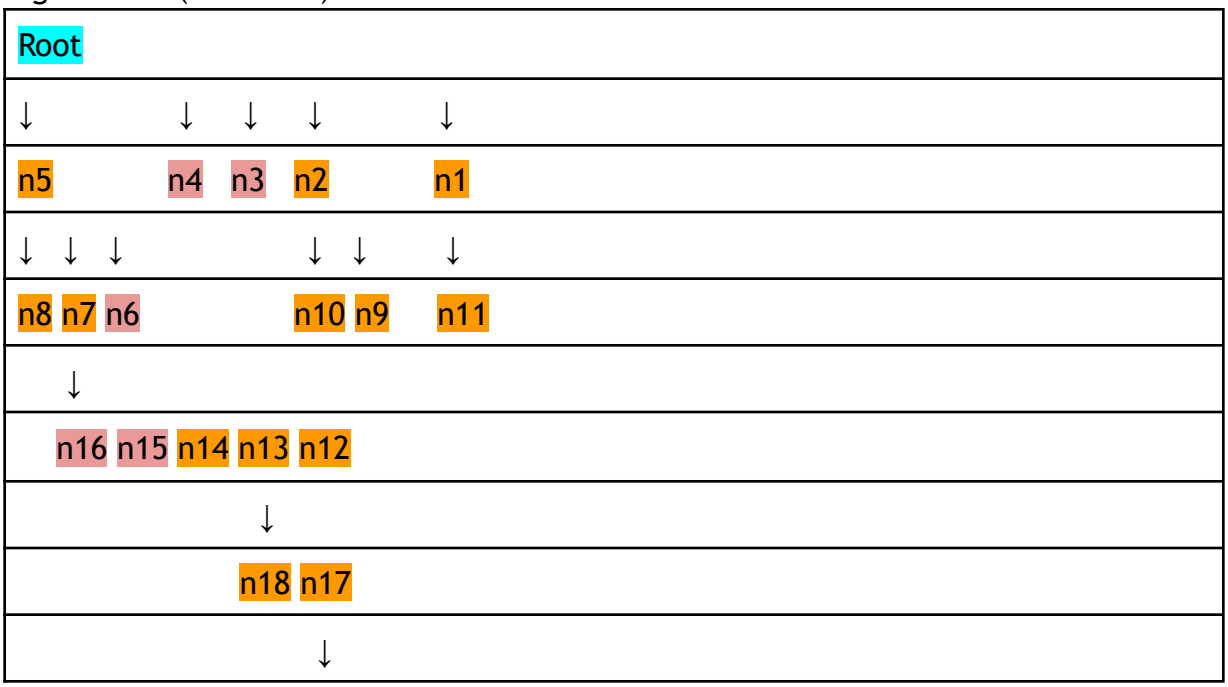


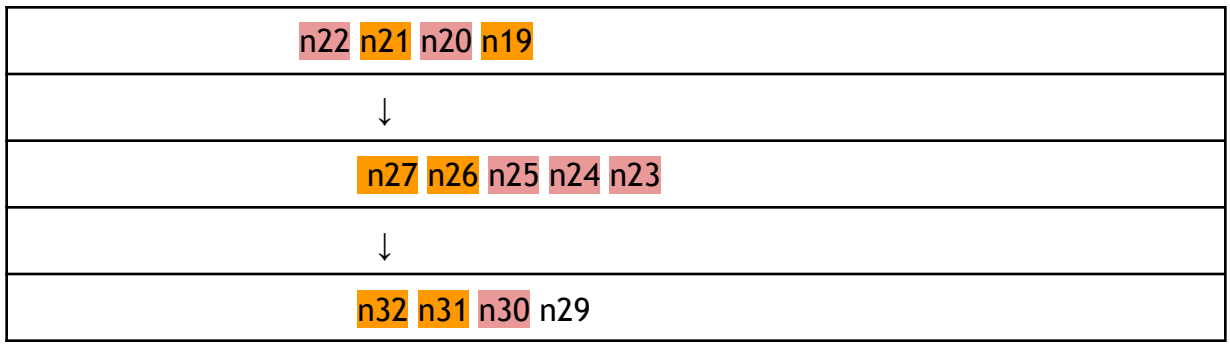
Node: n30
Parent: n27
Operator: move-m
Left bank: (m c c)
Right bank: (m m c b)



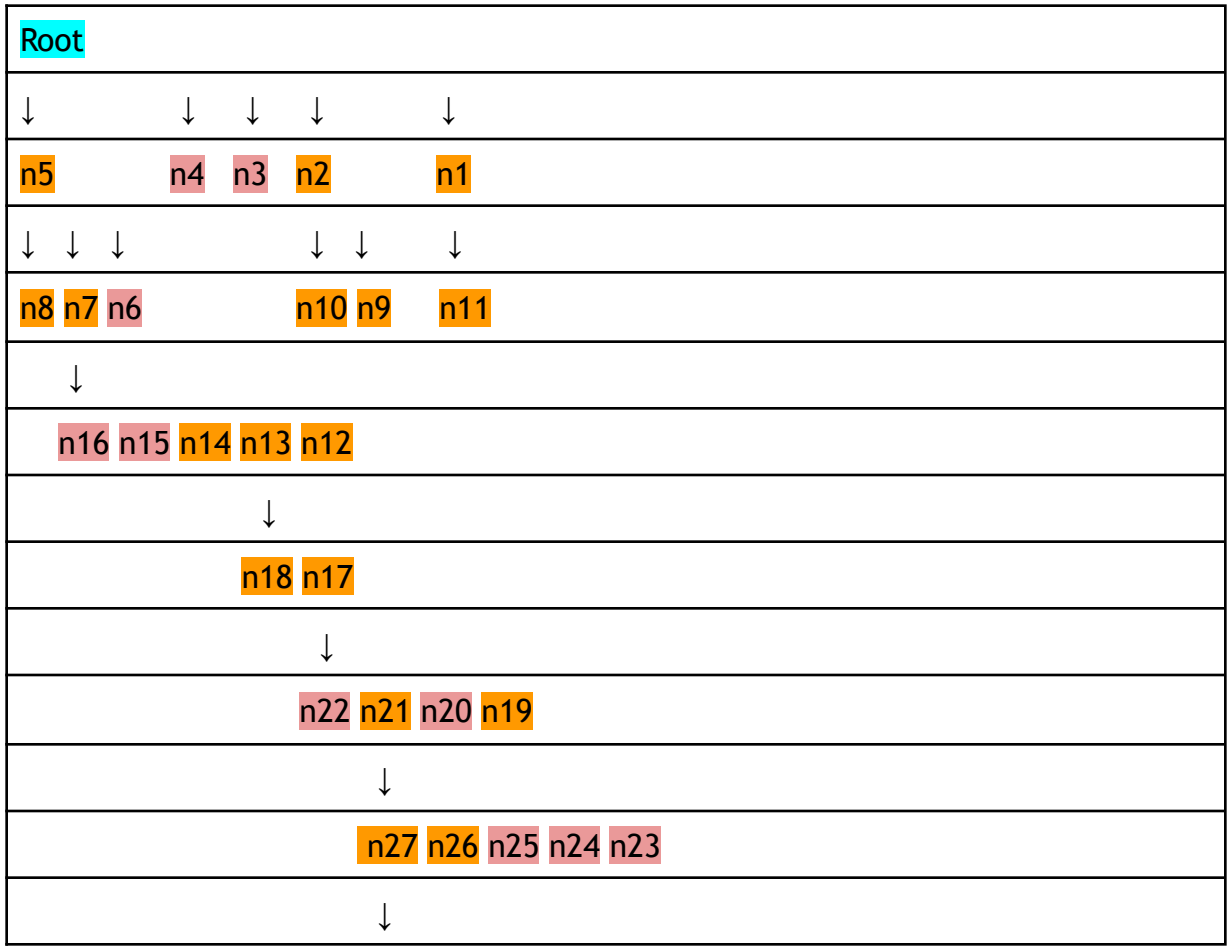


Node: n29
Parent: n27
Operator: move-c-c
Left bank: (m m)
Right bank: (m c c c b)



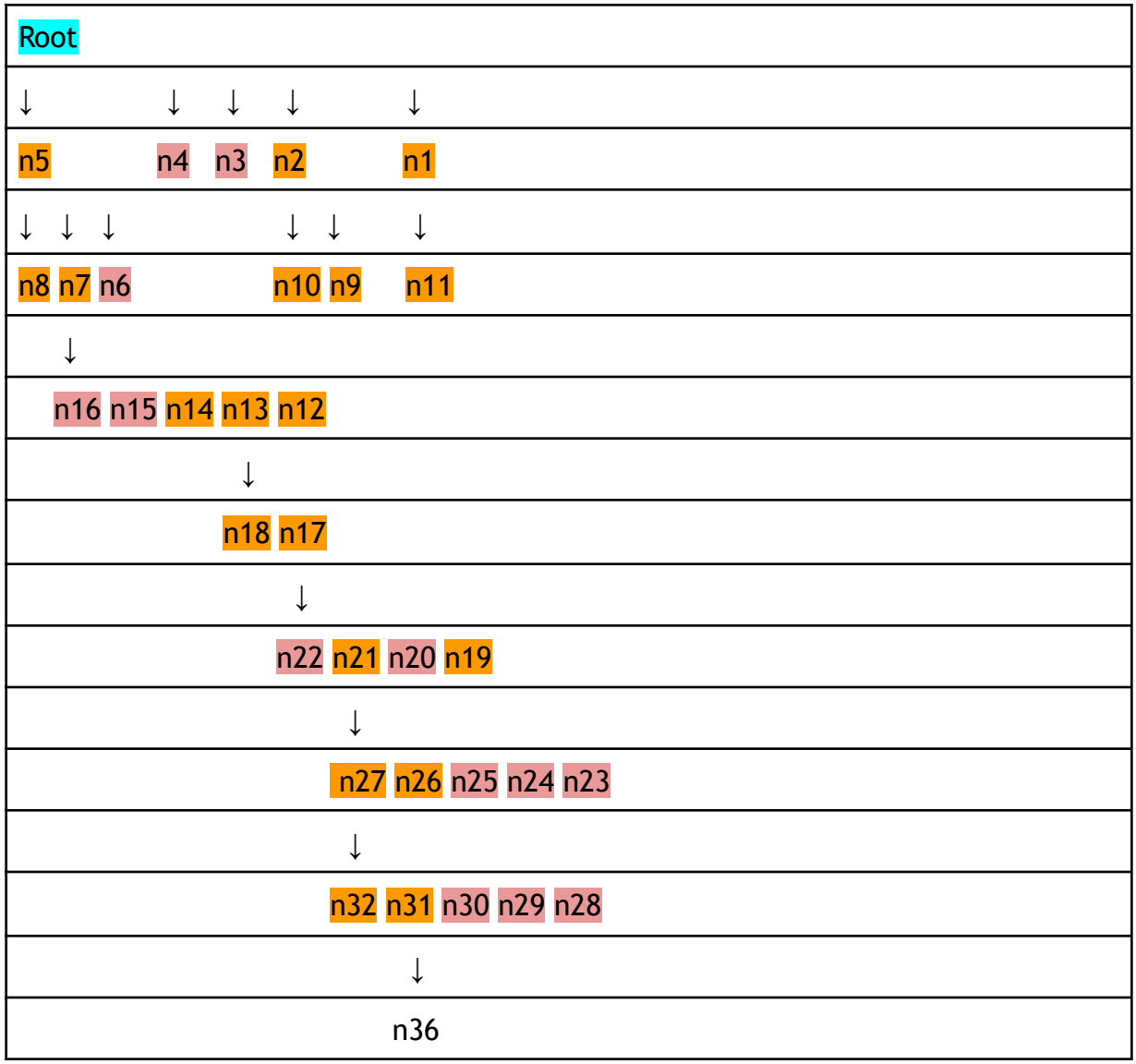


Node: n28
Parent: n27
Operator: move-c
Left bank: (m m c)
Right bank: (m c c b)

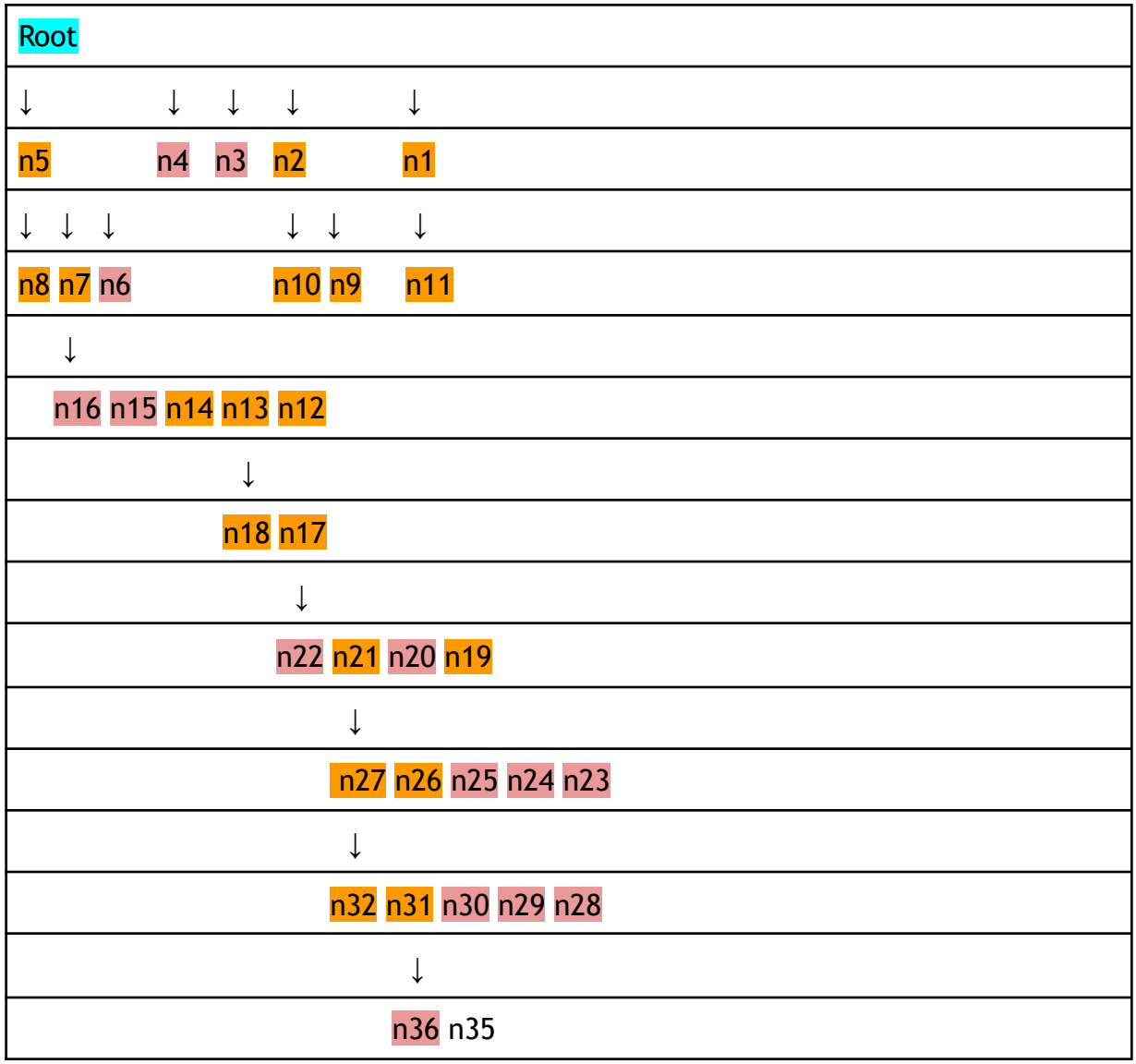


n32 n31 n30 n29 n28

Node: n36
Parent: n31
Operator: move-c-m
Left bank: (m c c c b)
Right bank: (m m)

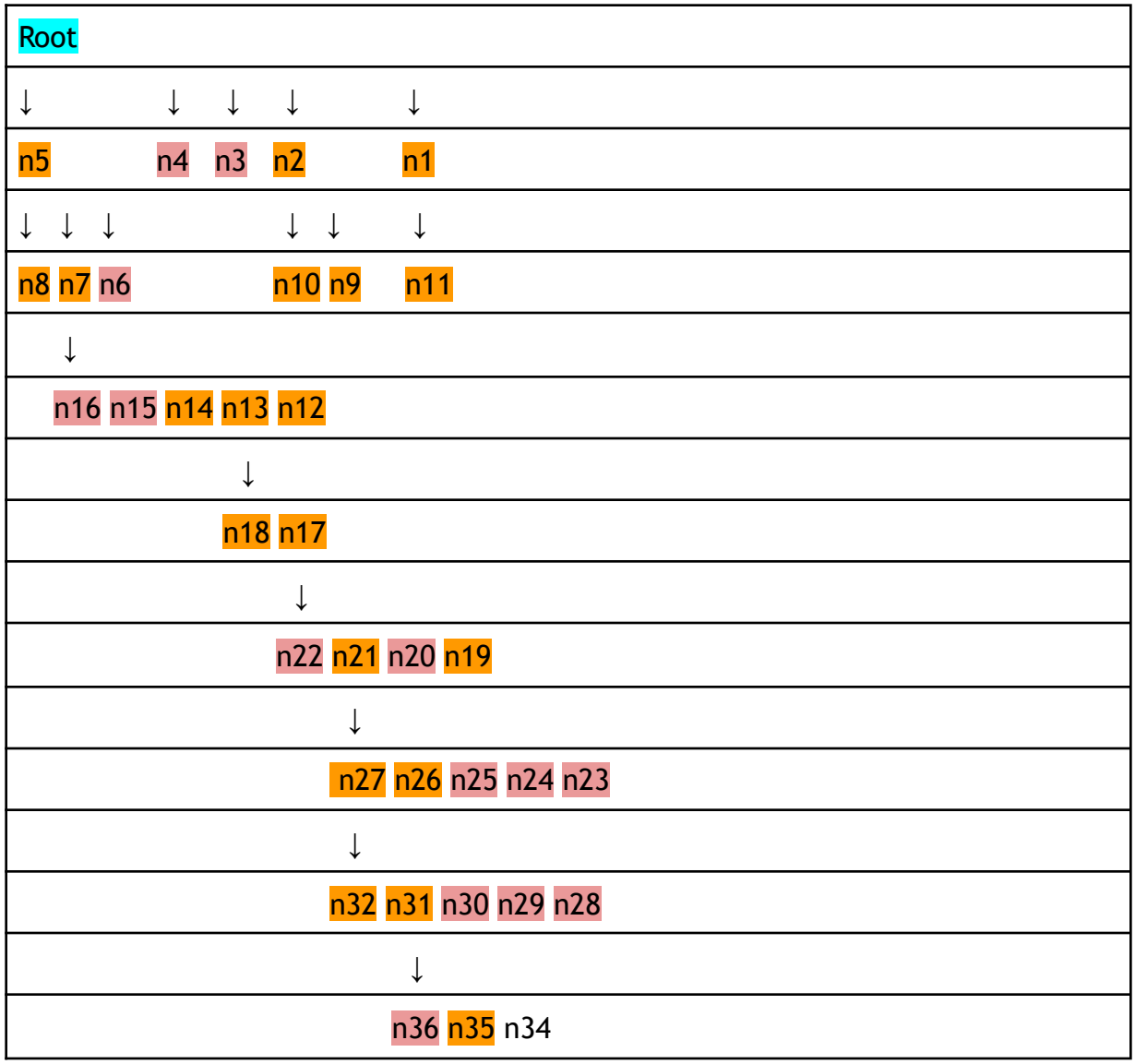


Node: n35
Parent: n31
Operator: move-m-m
Left bank: (m m c c b)
Right bank: (m c)

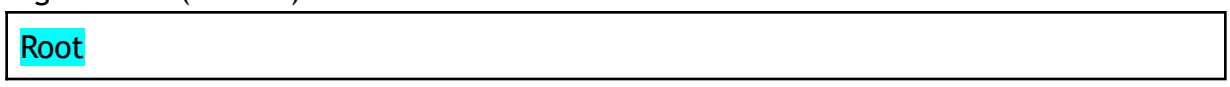


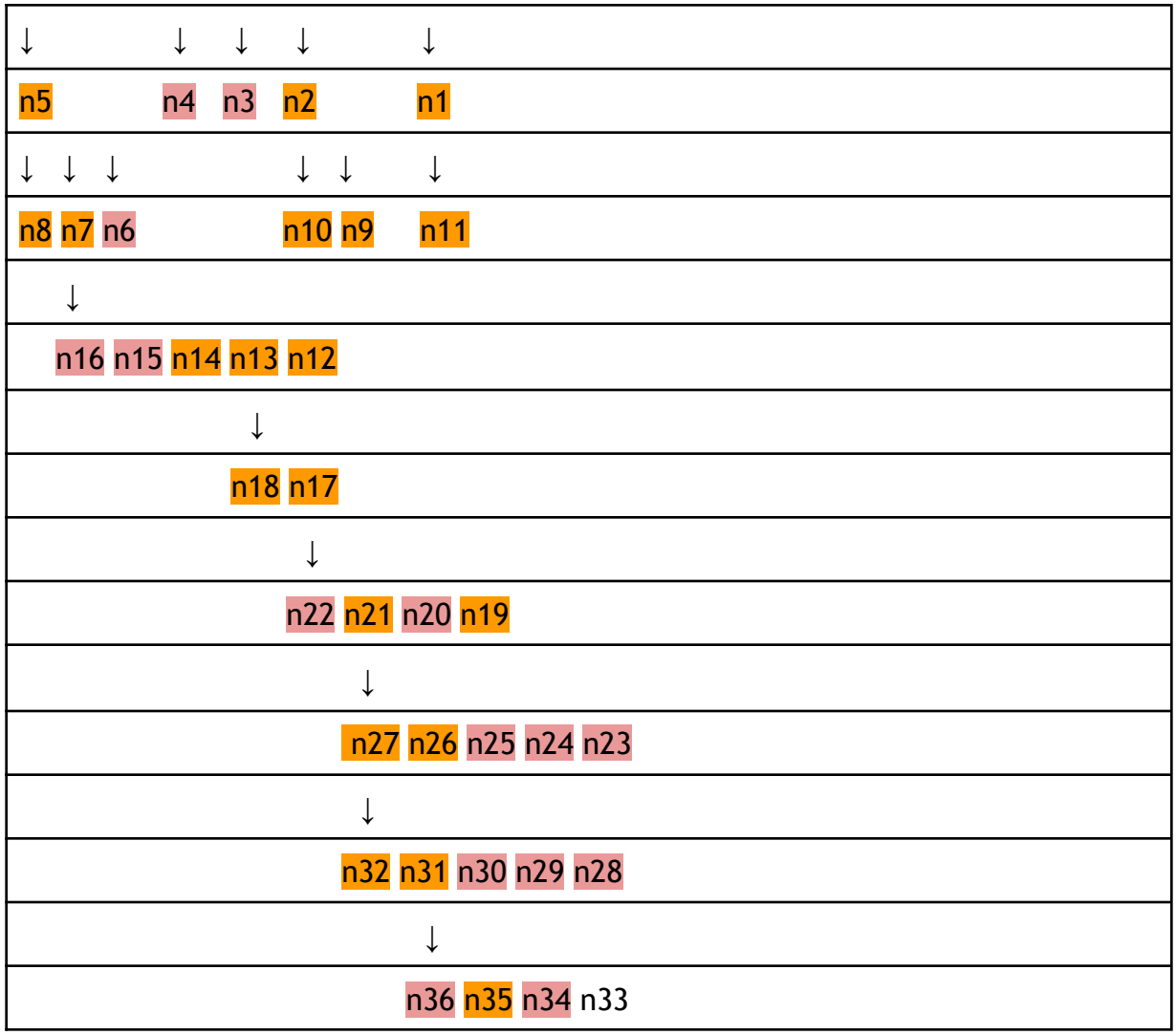
Node: n34
Parent: n31
Operator: move-m

Left bank: (m c c b)
Right bank: (m m c)

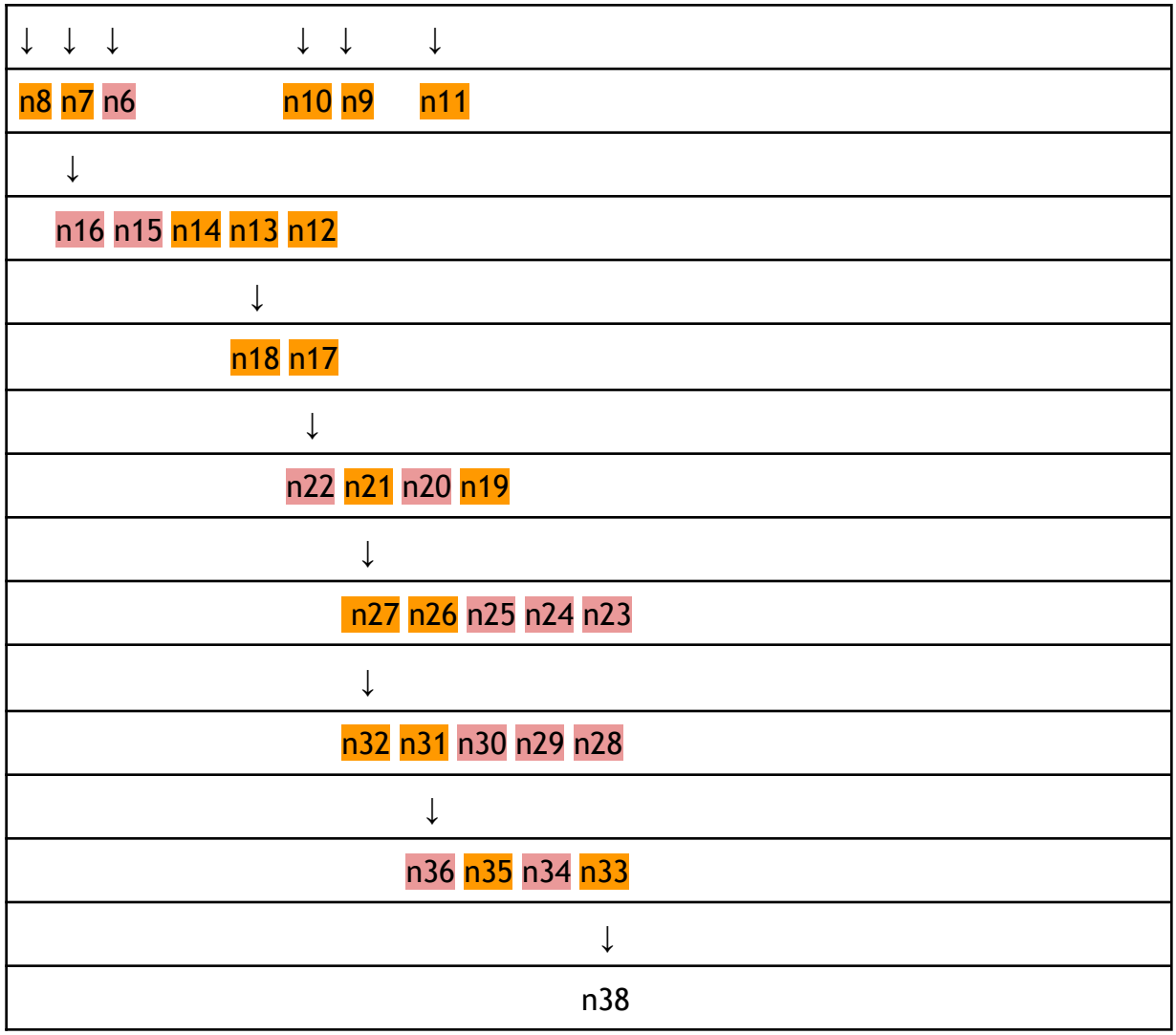


Node: n33
Parent: n31
Operator: move-c
Left bank: (c c c b)
Right bank: (m m m)

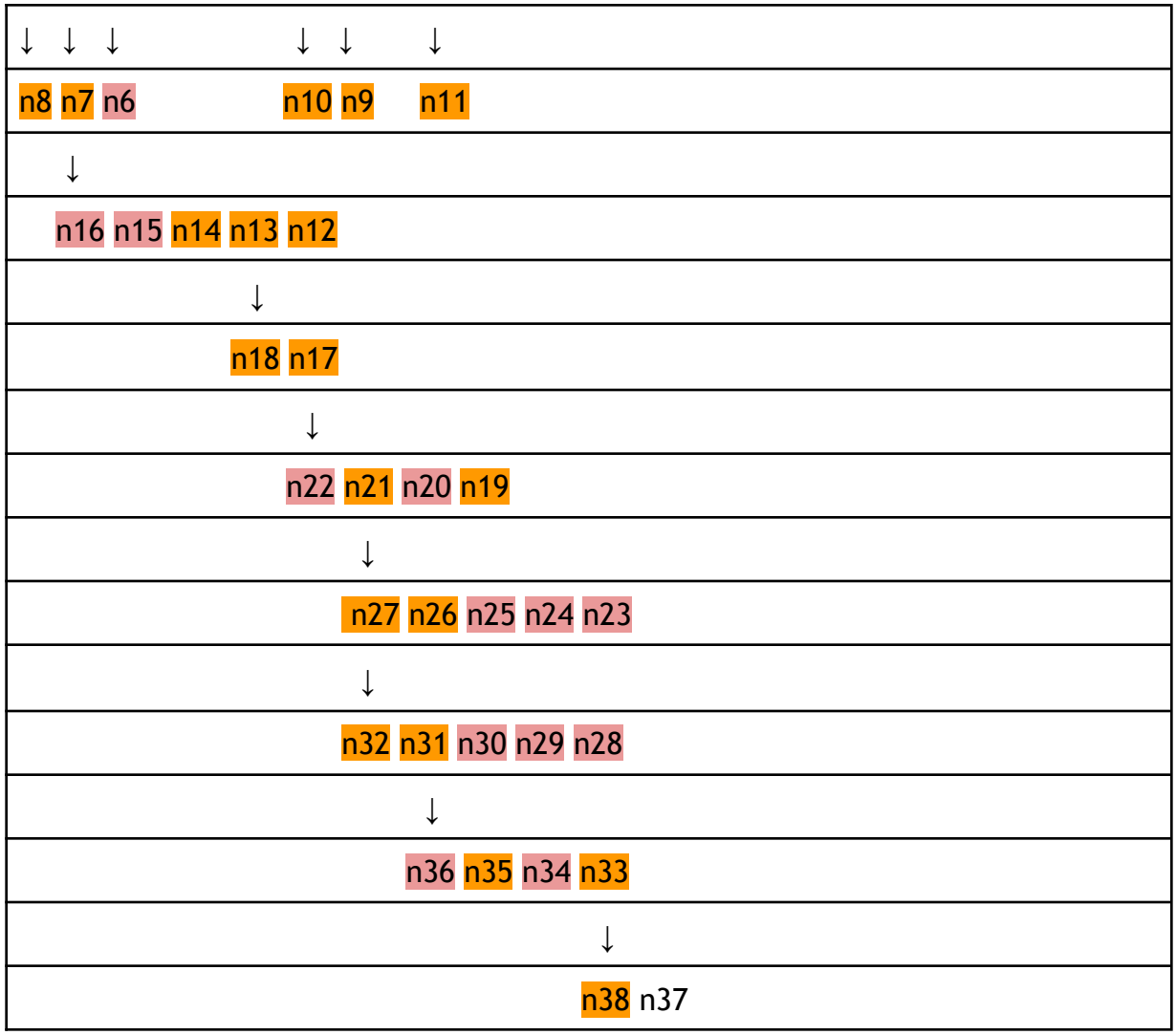




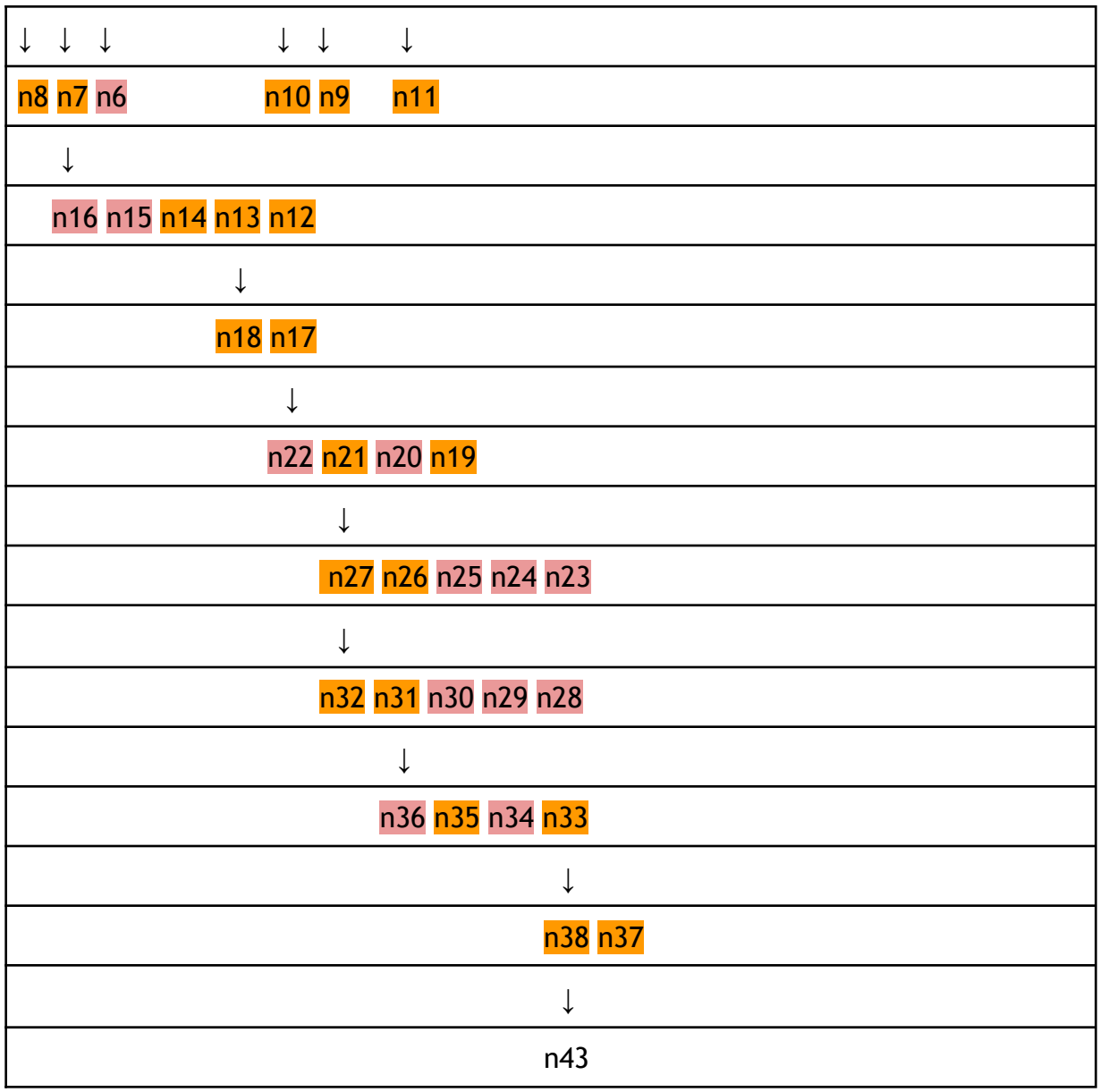
Node: n38
Parent: n33
Operator: move-c-c
Left bank: (c)
Right bank: (m m m c c b)



Node: n37
Parent: n33
Operator: move-c
Left bank: (c c)
Right bank: (m m m c b)

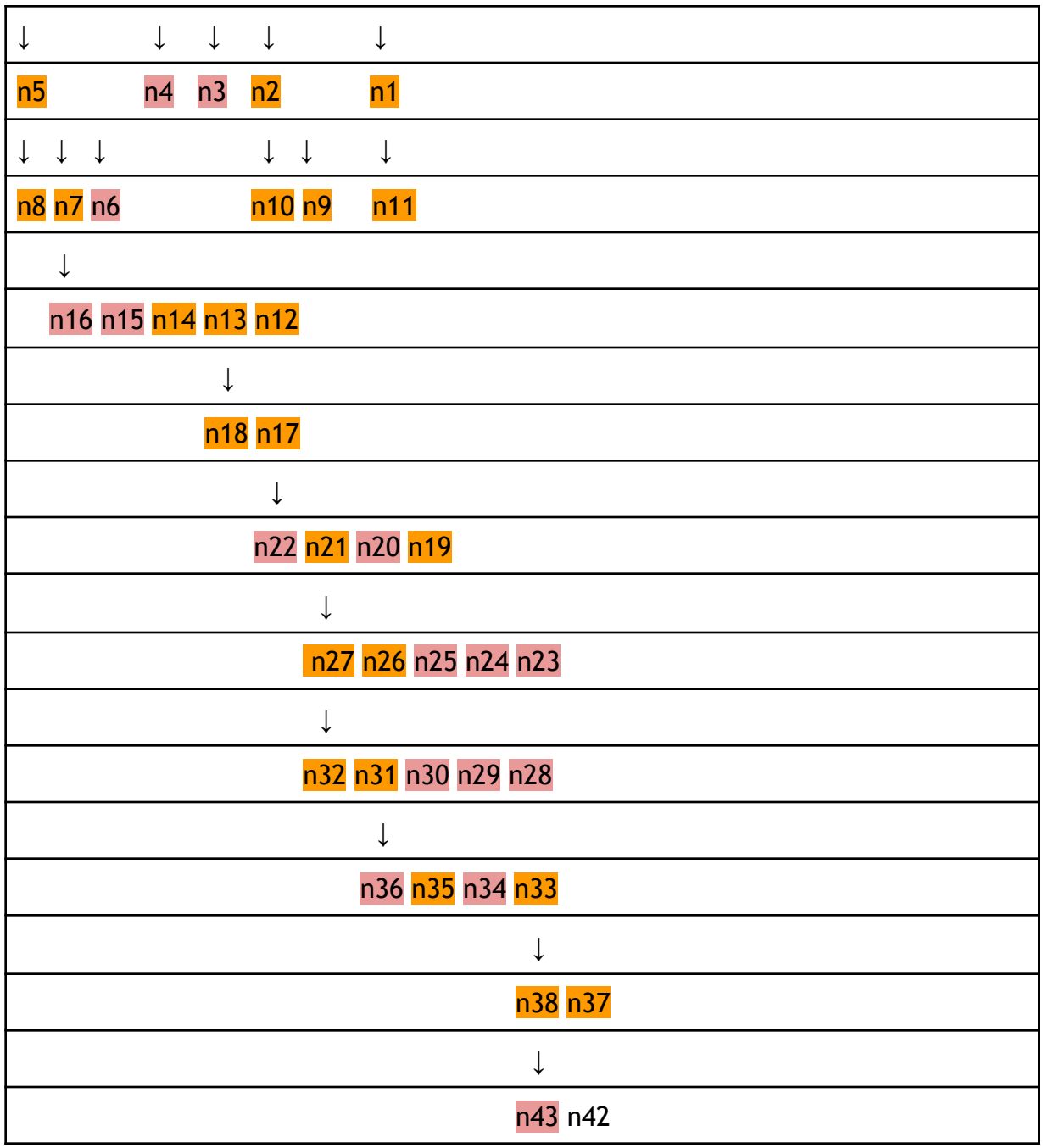


Node: n43
Parent: n38
Operator: move-c-m
Left bank: (m c c b)
Right bank: (m m c)



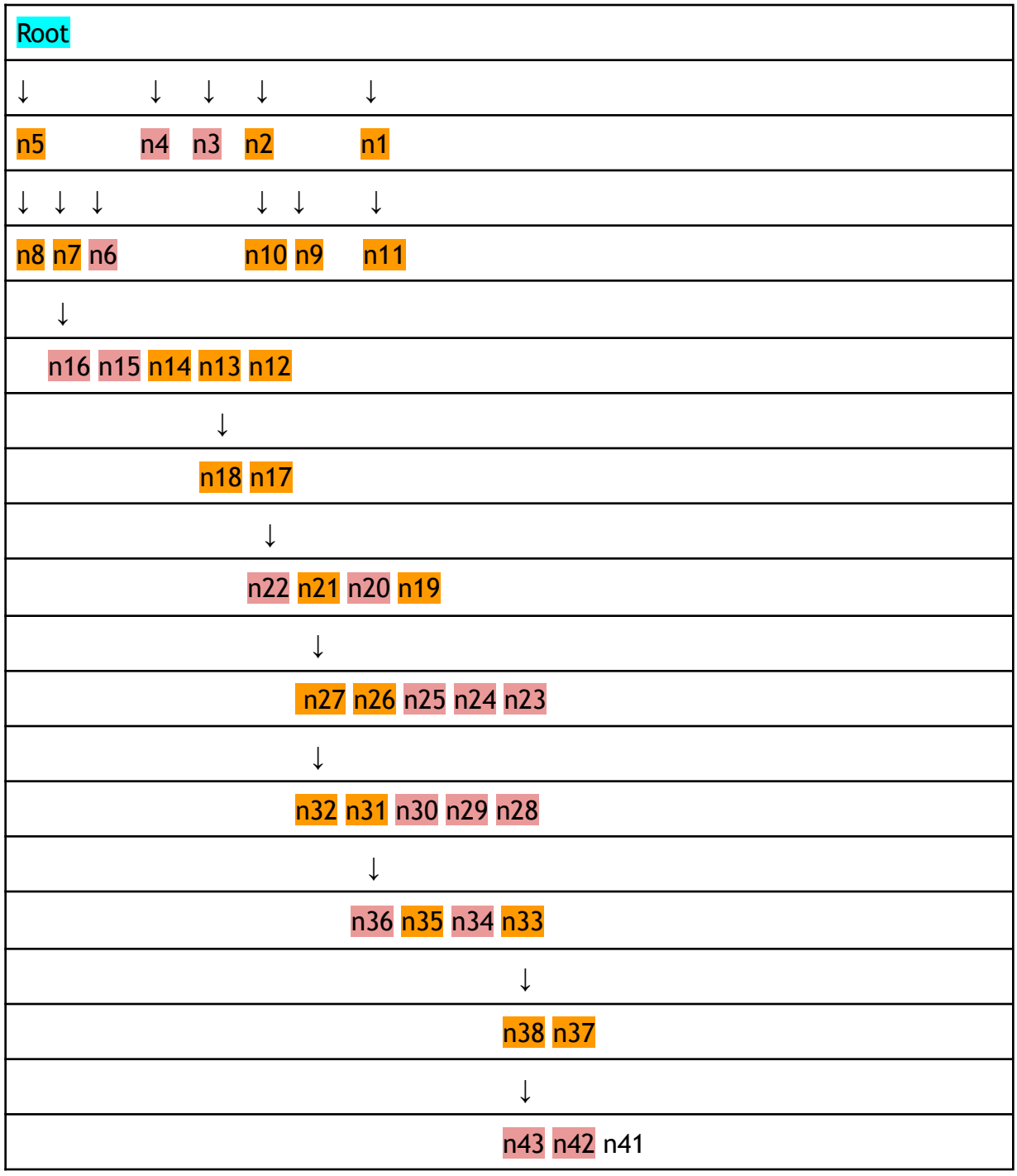
Node: n42
Parent: n38
Operator: move-m-m
Left bank: (m m c b)
Right bank: (m c c)

Root



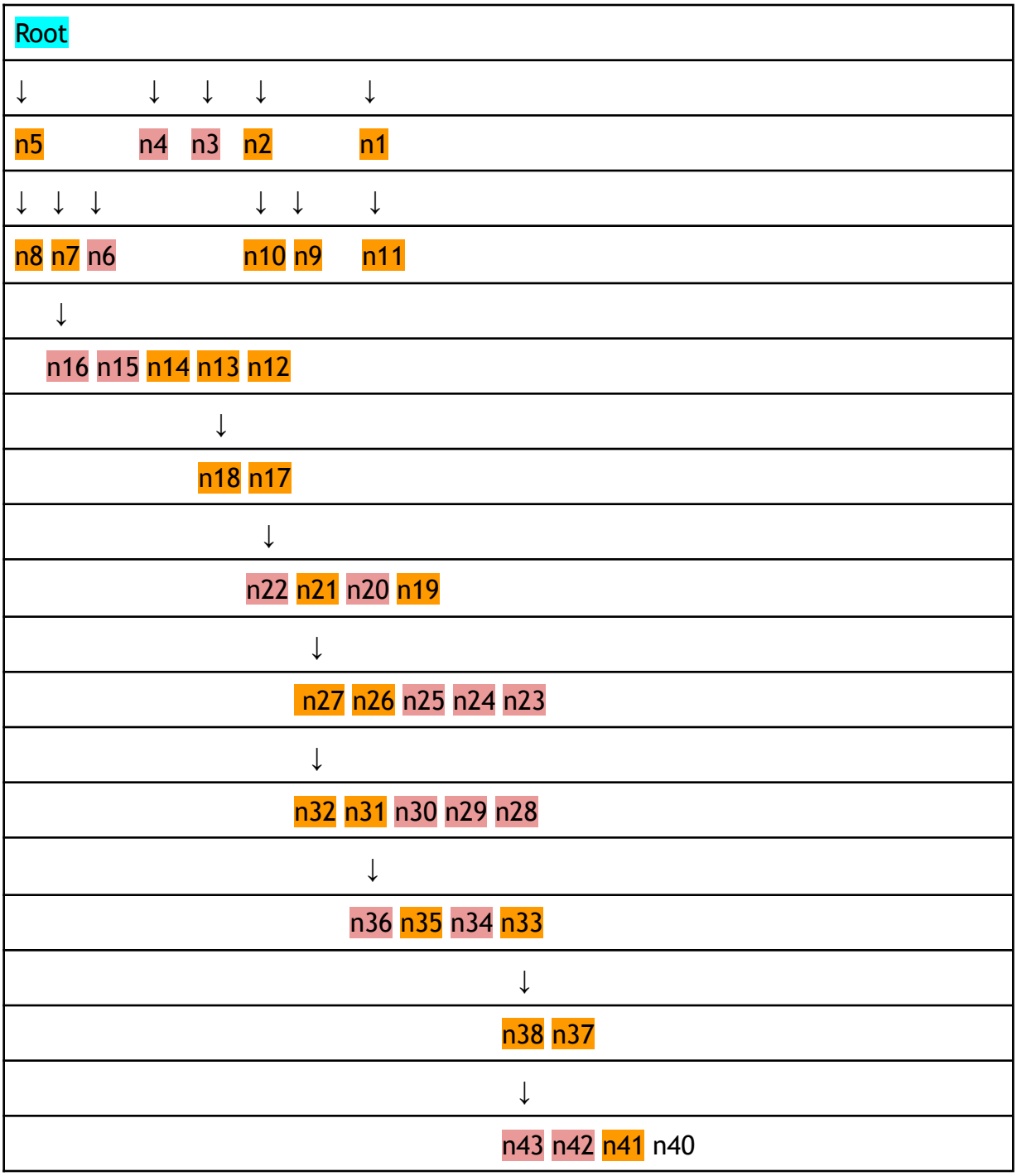
Node: n41
Parent: n38
Operator: move-m
Left bank: (m c b)

Right bank: (m m c c)

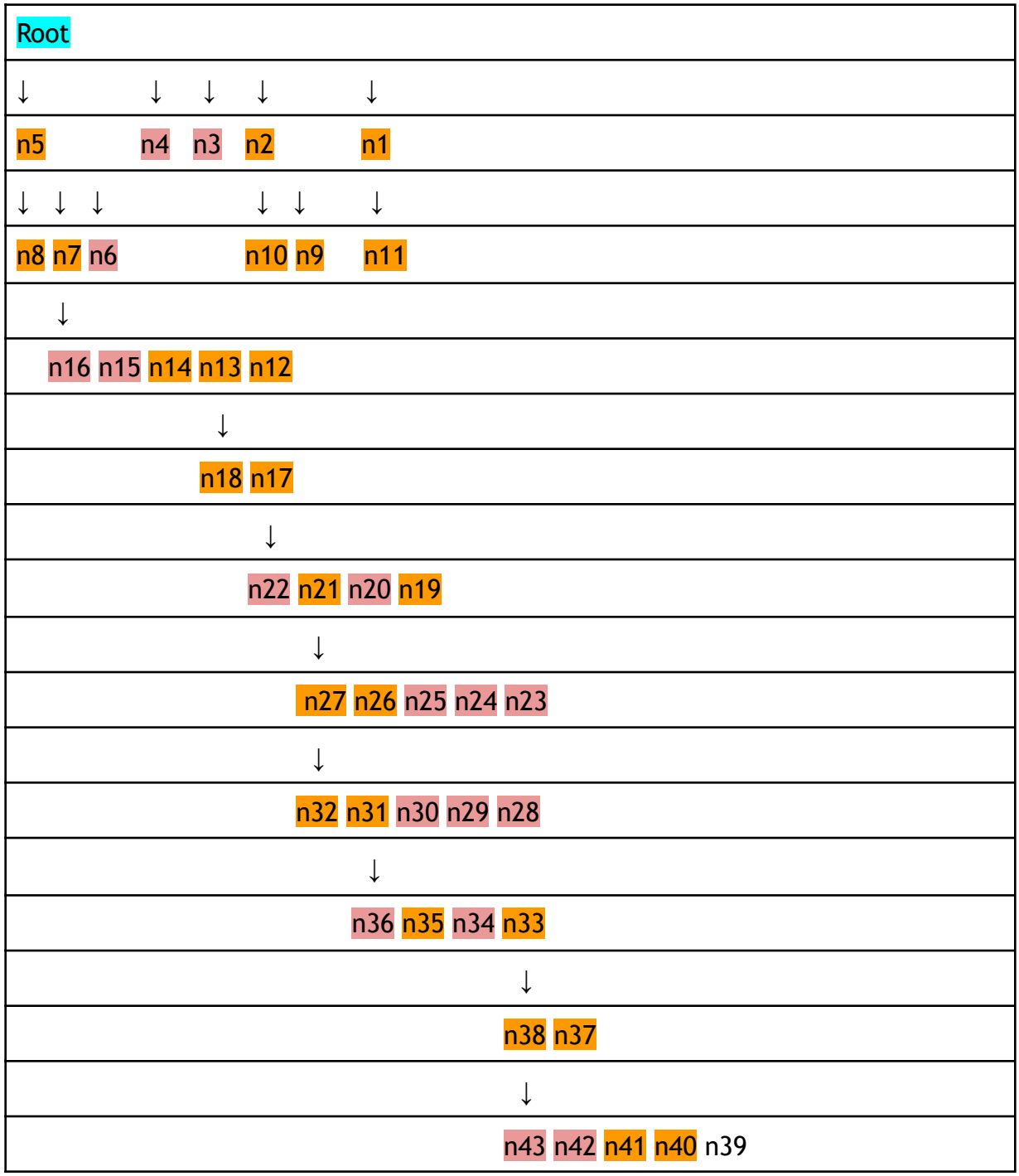


Node: n40

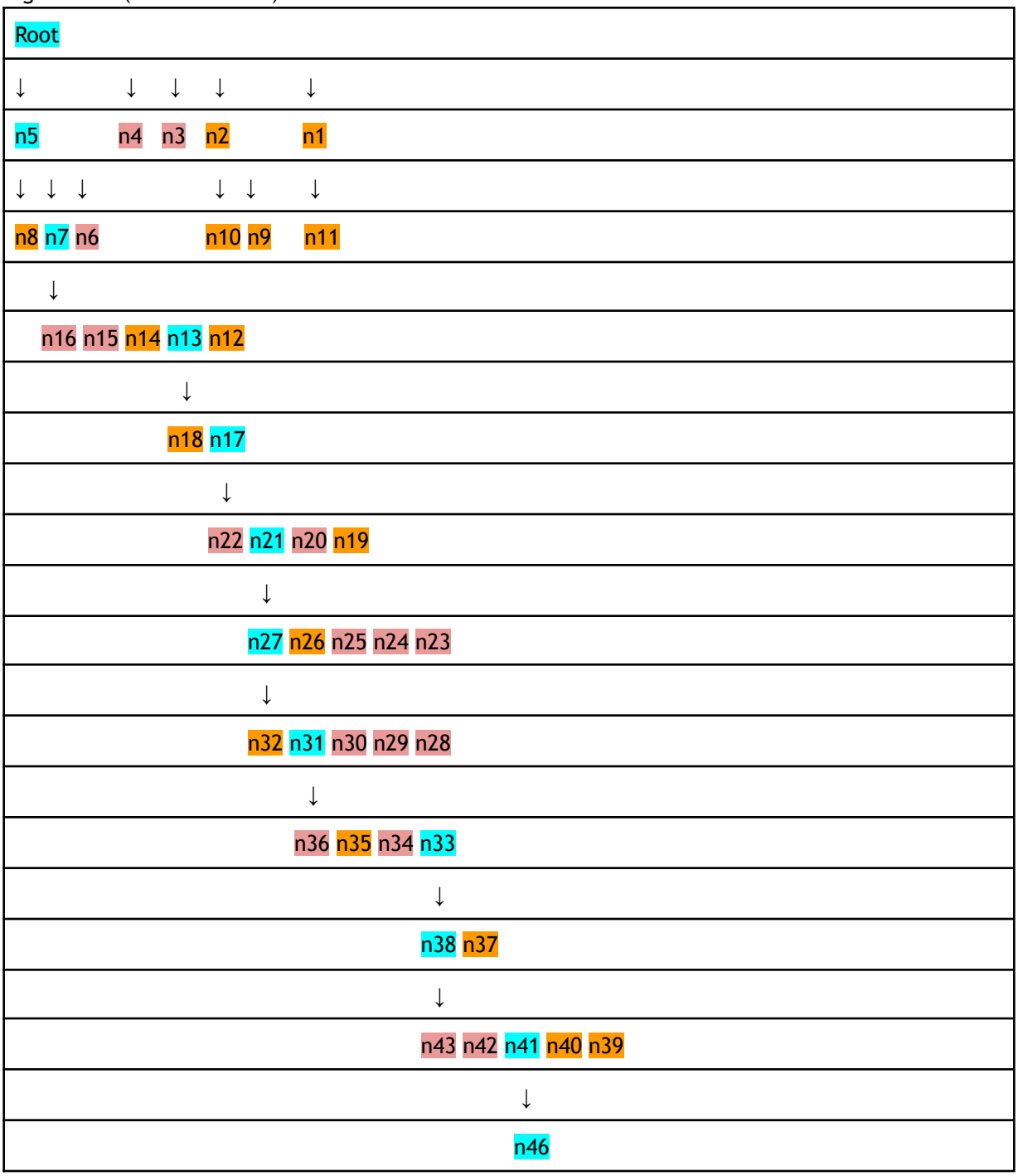
Parent: n38
Operator: move-c-c
Left bank: (c c c b)
Right bank: (m m m)



Node: n39
Parent: n38
Operator: move-c
Left bank: (c c b)
Right bank: (m m m c)



Node: n46
Parent: n41
Operator: move-c-m
Left bank: ()
Right bank: (m m m c c c b)



“Unexplored list / Explored list” Demo

Because there are countless lines in between the input and the solution, the majority of them are skipped.

```
[]> (mc 'ux)

>>> Unexplored list
ROOT
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> Explored list

>>> Unexplored list
N5 ROOT MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C)    boat: B
N4 ROOT MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C C C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: NIL    boat: B
N3 ROOT MOVE-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: NIL    boat: B
N2 ROOT MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C)    boat: B
N1 ROOT MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C)    boat: B

>>> Explored list
ROOT
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> Unexplored list
N4 ROOT MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C C C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: NIL    boat: B
N3 ROOT MOVE-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: NIL    boat: B
```

```
N2 ROOT MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C)    boat: B
N1 ROOT MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C)    boat: B
N8 N5 MOVE-C-M
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL
N7 N5 MOVE-M
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: B
Right bank: missionaries: NIL    cannibals: (C)    boat: NIL
N6 N5 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: (M)    cannibals: NIL    boat: NIL


###
###
###


>>> Unexplored list
N46 N41 MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: NIL    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
N45 N41 MOVE-M
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C)    boat: B
N44 N41 MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: NIL    boat: NIL
Right bank: missionaries: (M M)    cannibals: (C C C)    boat: B
N48 N39 MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: NIL    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
N47 N39 MOVE-C
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C)    boat: B

>>> Explored list
N39 N38 MOVE-C
Current bank: LEFT
Left  bank: missionaries: NIL    cannibals: (C C)    boat: B
Right bank: missionaries: (M M M)    cannibals: (C)    boat: NIL
N41 N38 MOVE-M
Current bank: LEFT
```

```
Left  bank: missionaries: (M)    cannibals: (C)    boat: B
Right bank: missionaries: (M M)    cannibals: (C C)    boat: NIL
N38 N33 MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C C)    boat: B
N33 N31 MOVE-C
Current bank: LEFT
Left  bank: missionaries: NIL    cannibals: (C C C)    boat: B
Right bank: missionaries: (M M M)    cannibals: NIL    boat: NIL
N31 N27 MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: NIL    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M M M)    cannibals: (C)    boat: B
N27 N21 MOVE-C-M
Current bank: LEFT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: B
Right bank: missionaries: (M)    cannibals: (C)    boat: NIL
N21 N17 MOVE-M-M
Current bank: RIGHT
Left  bank: missionaries: (M)    cannibals: (C)    boat: NIL
Right bank: missionaries: (M M)    cannibals: (C C)    boat: B
N17 N13 MOVE-C
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: B
Right bank: missionaries: NIL    cannibals: (C C)    boat: NIL
N13 N7 MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: NIL    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C C)    boat: B
N7 N5 MOVE-M
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: B
Right bank: missionaries: NIL    cannibals: (C)    boat: NIL
N1 ROOT MOVE-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C)    boat: B
N2 ROOT MOVE-C-C
Current bank: RIGHT
Left  bank: missionaries: (M M M)    cannibals: (C)    boat: NIL
Right bank: missionaries: NIL    cannibals: (C C)    boat: B
N5 ROOT MOVE-C-M
Current bank: RIGHT
Left  bank: missionaries: (M M)    cannibals: (C C)    boat: NIL
Right bank: missionaries: (M)    cannibals: (C)    boat: B
ROOT
Current bank: LEFT
Left  bank: missionaries: (M M M)    cannibals: (C C C)    boat: B
Right bank: missionaries: NIL    cannibals: NIL    boat: NIL

>>> GOT A SOLUTION!
Move (b c m) to the other bank.
Move (b m) to the other bank.
Move (b c c) to the other bank.
Move (b c) to the other bank.
```

```
Move (b m m) to the other bank.
Move (b c m) to the other bank.
Move (b m m) to the other bank.
Move (b c) to the other bank.
Move (b c c) to the other bank.
Move (b m) to the other bank.
Move (b c m) to the other bank.
NIL
```