Stack Usage

VE281 - Data Structures and Algorithms, Xiaofeng Gao, Autumn 2019

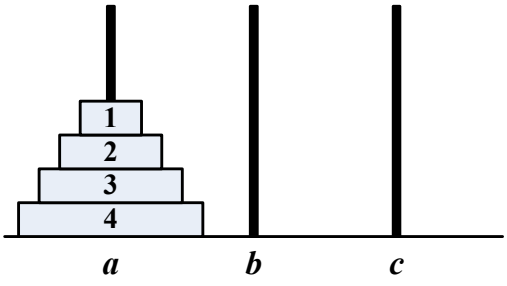
Hanio Tower

1. **Pseudocodes**

Algorithm 1: Hanoi(n, a, b, c)

Input: a, b and c. n disks
Output: Moving strategy from a to c.

```
1 if n = 1 then
2 |   Move(a, 1, c);
3 else
4 |   Hanoi(n − 1, a, c, b);
5 |   Move(a, n, c);
6 |   Hanoi(n − 1, b, a, c);
```



Hanoi Tower

2. **Function Operations**

Hanoi(4, a, b, c)
- ⊢ Hanoi(3, a, c, b)
  - ⊢ Hanoi(2, a, b, c)
    - ⊢ Hanoi(1, a, c, b)
      - ⌞ Move(a, 1, b)
    - ⊢ Move(a, 2, c)
    - ⌞ Hanoi(1, b, a, c)
      - ⌞ Move(b, 1, c)
  - ⊢ Move(a, 3, b)
  - ⌞ Hanoi(2, c, a, b)
    - ⊢ Hanoi(1, c, b, a)
      - ⌞ Move(c, 1, a)
    - ⊢ Move(c, 2, b)
    - ⌞ Hanoi(1, a, c, b)
      - ⌞ Move(a, 1, b)
- ⊢ Move(a, 4, c)
- ⌞ Hanoi(3, b, a, c)
  - ⊢ Hanoi(2, b, c, a)
    - ⊢ Hanoi(1, b, a, c)
      - ⌞ Move(b, 1, c)
    - ⊢ Move(b, 2, a)
    - ⌞ Hanoi(1, c, b, a)
      - ⌞ Move(c, 1, a)
  - ⊢ Move(b, 3, c)
  - ⌞ Hanoi(2, a, b, c)
    - ⊢ Hanoi(1, a, c, b)
      - ⌞ Move(a, 1, b)
    - ⊢ Move(a, 2, c)
    - ⌞ Hanoi(1, b, a, c)
      - ⌞ Move(b, 1, c)

Stack Usage:
- ⊢ H(4, a, b, c)
- ⊢ H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(4, a, b, c)
- ⊢ H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, c, a, b), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, c, a, b), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, c, a, b), H(3, a, c, b), H(4, a, b, c)
- ⊢ H(2, c, a, b), H(3, a, c, b), H(4, a, b, c)
- ⊢ ∅
- ⊢ H(4, a, b, c)
- ⊢ H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, b, c, a), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, b, c, a), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, b, c, a), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, b, c, a), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(4, a, b, c)
- ⊢ H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, b, a, c), H(4, a, b, c)
- ⊢ H(2, a, b, c), H(3, b, a, c), H(4, a, b, c)

Parenthesis Matching

1. **Introduction**

   Stack can be used to deal with the **Parenthesis Matching** problem. Actually, we can implement a formula calculator with stack.

   When the calculator is doing the operation, we must compare the priority order of the operators. The relative orders are shown in table on the right side, where θ_1 and θ_2 represent the input operator and ">, <, =" represent the relative order between them.

   Every time the calculator reads an input operator θ_2, it should always be compared with θ_1 that is on the top of "StackR". If $\theta_1 < \theta_2$, then θ_2 should be pushed into "StackR". Otherwise, θ_2 should be popped out of stack. Then the calculator should fetch two numbers stored in "StackD" and carry out the operation.

Relative order between operators.

$\theta_1 \backslash \theta_2$	+	−	∗	/	(	)	#
+	>	>	<	<	<	>	>
−	>	>	<	<	<	>	>
∗	>	>	>	>	<	>	>
/	>	>	>	>	<	>	>
(	<	<	<	<	<	=	
)	>	>	>	>		>	>
#	<	<	<	<	<		=

2. **Example**

   Here we provide an example with the input "$\#(9-3)*2/(5+1)\#$". Note that the operator "#" indicates the start and end points of the formula.

No	StackR	StackD	Inputs	Operations	Explanations
0			$\underline{\#}(9-3)*2/(5+1)\#$	push(StackR, #)	Push starting # into stack.
1	#		$\underline{(}9-3)*2/(5+1)\#$	push(StackR, ()	StackR top, "#" < "(". Push "(".
2	#(		$\underline{9}-3)*2/(5+1)\#$	push(StackD, 9)	Number 9 in StackD.
3	#(	9	$\underline{-}3)*2/(5+1)\#$	push(StackR, −)	"(" < " − ". Push " − ".
4	#(−	9	$\underline{3})*2/(5+1)\#$	push(StackD, 3)	Number 3 in StackD.
5	#(−	9 3	$\underline{)}*2/(5+1)\#$	operate(9 − 3)	" − " > ")". Execute "9 − 3"
6	#(	6	$\underline{)}*2/(5+1)\#$	pop(StackR)	Eliminate a pair of parentheses.
7	#	6	$\underline{*}2/(5+1)\#$	push(StackR, ∗)	"#" < " ∗ ". Push " ∗ ".
8	#∗	6	$\underline{2}/(5+1)\#$	push(StackD, 2)	Number 2 in StackD.
9	#∗	6 2	$\underline{/}(5+1)\#$	operate(6 ∗ 2)	" ∗ " > "/". Execute "6 ∗ 2"
10	#	12	$\underline{/}(5+1)\#$	push(StackR, /)	"#" < "/". Push "/".
11	#/	12	$\underline{(}5+1)\#$	push(StackR, ()	"#" < "(". Push "(".
12	#/(	12	$\underline{5}+1)\#$	push(StackD, 5)	Number 5 in StackD.
13	#/(	12 5	$\underline{+}1)\#$	push(StackR, +)	"(" < " + ". Push " + ".
14	#/(+	12 5	$\underline{1})\#$	push(StackD, 1)	Number 1 in StackD.
15	#/(+	12 5 1	$\underline{)}\#$	operate(5 + 1)	" + " > ")". Execute "5 + 1"
16	#/(	12 6	$\underline{)}\#$	pop(StackR)	Eliminate a pair of parentheses.
17	#/	12 6	$\underline{\#}$	operate(12/6)	"/" > "#". Execute "12/6"
18	#	2	$\underline{\#}$	pop(StackD)	Read "#", end.