Seminar 05

Thursday, May 5, 2022 2:57 PM

$x = (x_1, x_2, \dots, x_n)$ — x_i are numbers

1, 2, 3, 5, 12

k - natural numbers

2 4 10 9 8 11 7

$k = 4$

binary max-heap

heap: init(h, relation)
push - O(log(heap_size))
pop - O(log(heap_size))
top - O(1)

function sumk(x, n, k) is:
    init(h, ">=")
    for i = 1, n execute:
        push(h, x[i])
    endfor
    sum ← 0
    for i = 1, k execute
        sum ← sum + pop(h)
    endfor
    sumk ← sum
end function

Time-Complexity: $O(n \log n) + O(k \log n) = O((n+k)\log n) \sim O(n \log n)$, $k \leq n$

function sumk_v2(x, n, k) is:
    init(h, "<=")

```
for i=1,k execute:
    push(h, x[i])
endfor

for i<-k,n execute:
    if x[i] >= top(h) then:
        pop(h)
        push(h, x[i])
endfor
sum <- 0
for i<-1,k execute:
    sum <- sum + pop(h)
endfor
sumk_V2 <- sum
endfunction
```

Time Complexity: $O(k \log k) + O((n-k)\log(k)) + O(k \log k) =$

$= O((n+k)\log k) \sim O(n \log k)$

Heapify

heapify → algorithm to create a heap from an array

↘ $O(n)$

```
sum <- 0
for i<-1,k execute
    sum <- sum + pop(x)   / O(log n)   / O(k log n)
endfor
```

endfor

$O(n + k \log n)$

Evaluate an arithmetic expression
- digits, operands (1, 2, ..., 9)
- operators (*, -, /, +)
- parantheses ("(", ")")

$(2+4)*3-4=14$

infix

$24+3*4-$

postfix

I. transforming to postfix notation

stack

queue

$2*(4+3)-4/6$

Steps
- operand: add to queue
- operator: 1) move operators with higher priority from stack to queue
  As long as there is a higher priority operator on top of the stack
  2) add the operator to the stack

- open paranthesis "(" push to stack;
- closed paranthesis (")"): while the open paranthesis is not at the top of the stack move from stack to queue
  remove open paranthesis from the stack
- at the end put everything from the stack to the queue.

stack: init : queue
push
pop
top
isEmpty
/ $\Theta(1)$

isOperator(e)
isOperand(e) / $\Theta(1)$

@hasHigherPriority

length(expression) / $\Theta(1)$

```
function transform(expression) is:
    init(q)
    init(s)
    for i=1, length(expression) execute:
        el ← expression[i]
        if isOperand(el) then:
            push(q, el)
```

```
        else if isOperator(el) then:
            while ¬isEmpty(S) ∧ @ top(S) hasHigherPriority(el) ∧ :
                                    ∧ isOperator(top(S)) then:
                push(q, pop(S))
            endwhile
            push(S, el)
        else if el = "(" then:
            push(S, el)
        else:
            while ¬isEmpty(S) ∧ top(S) ≠ "(" execute:
                push(q, pop(S))
            endwhile
            if ¬isEmpty(S) then:
                pop(S)
            endif
        endif
        while ¬isEmpty(S) execute:
            push(q, pop(S))
        endwhile
    endfor
    transform ← q
endfunction
```

II. Evaluate postfix notation

$2*(4+3)-4/6 \longrightarrow 243+*461-$

```
function evaluate(postfix) is:
```

```
v   init(S)
    while ¬isEmpty(postfix) execute:
        if isOperand(top(postfix)) then:
            push(S, pop(postfix))
        else if isOperand(top(postfix)) then:
            el1 ← pop(S)
            el2 ← pop(S)
            res ← @ compute (el2, el1)
            pop (postfix)
            push(S, res)
        endif
    endwhile
    evaluate ← top(S)
endfunction
```