Computing complexity for the "Towers of Hanoi" problem

hanoi(3,1,2,3)

hanoi(2,1,3,2) move(1,2) hanoi(2,3,2,1)

h(1,1,2,3) h(1,2,3,1) h(1,3,1,2) h(1,1,2,3)

move(1,3) move(3,2)

$$C(0) = 0$$

$$C(n) = 2C(n-1) + 1$$

```
hanoi(n, from, to, using) {
  if (n > 0) {
    hanoi(n-1, from, using, to)
    move(from, to)
    hanoi(n-1, using, to, from)
  }
}
```

param: n
unit op: move

$C(0) = 0$

$C(n) = 2C(n-1) + 1$

$D(n) = C(n) + 1$

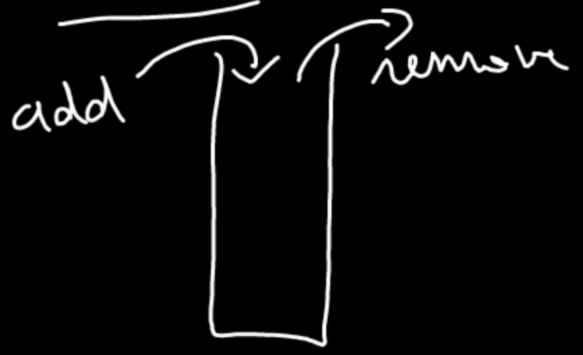
$C(n) + 1 = 2\left(C(n-1) + 1\right)$

$D(0) = 1$

$D(n) = 2D(n-1)$

$D(n) = 2^n \Rightarrow \boxed{C(n) = 2^n - 1} = \Theta(2^n)$

Stack:



LIFO

isEmpty: test. } the stack is empty

top: returns the topmost element

push: adds an elt on top of the stack

pop: removes the topmost elt

ADT Stack

Use Boolean, Element

Operations

* new: → Stack

isEmpty: Stack → Boolean

top: Stack → Element

* push: Stack × Element → Stack

pop: Stack → Stack

Precondition

top(s) iff $\neg$ isEmpty(s) ≡ F

pop(s) iff s ≠ new

Axioms

A_1 isEmpty (new) $\equiv$ T

A_2 isEmpty (push(s,e)) $\equiv$ F

A_3 top(push(s,e)) $\equiv$ e

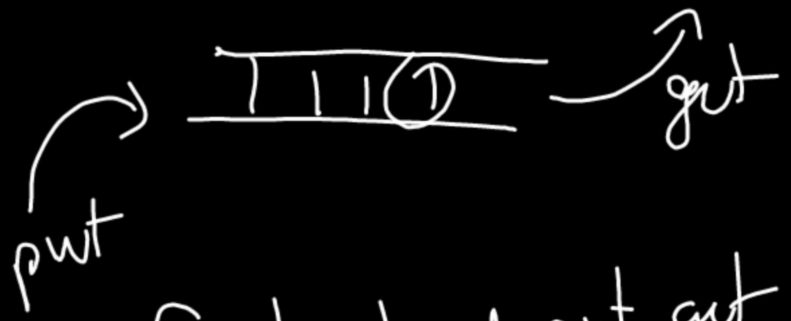
A_4 pop(push(s,e)) $\equiv$ s

top(pop (push (push (new, e_1), e_2)))

A_4 top(push (new, e_1))

A_3 e_1

e_1



isEmpty, head, put, get

ADT Queue
Use Boolean, Element

Operations

* new : → Queue
isEmpty : Queue → Boolean
head : Queue → Element
* put : Queue × Element → Queue
get : Queue → Queue

Preconditions

get(q) iff isEmpty(q) ≡ F
head(q) iff q != new

Axioms

A_1 isEmpty(new) $\equiv$ T

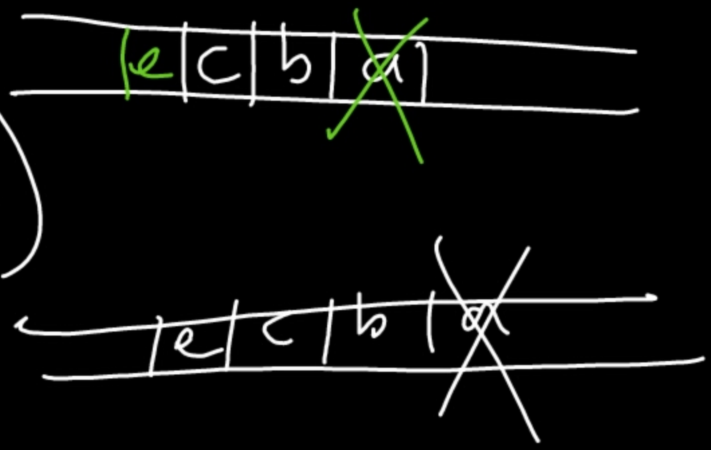
A_2 isEmpty(put(q,e)) $\equiv$ F

A_3 head(put(new,e)) $\equiv$ e

A_4 head(put(q,e)) $\equiv$ head(q) $\leftarrow$ q != new

A_5 get(put(new,e)) $\equiv$ new

A_6 get(put(q,e)) $\equiv$ put(get(q),e)



Lists.

* access by position (1 → length)
* add/remove at any position

df: a list is either empty, or made of (cons)
- a first element
- a rest (list)

(A B C D)

A
(B C D)

ADT List

Use Element

Operations

* new : $\longrightarrow$ List

first : List $\nrightarrow$ Element

rest : List $\nrightarrow$ List

* cons : Element $\times$ List $\longrightarrow$ List

Preconditions

first(l) iff $l \not\equiv new$

rst (l) ———————

Axioms

A1. first (cons (e, l)) $\equiv$ e

A2 rest (cons (e, l)) $\equiv$ l

e A

l (B C D)

cons (e, l) $\longrightarrow$ (A B C D)

cons (A, cons (B, cons (C, cons (D, new))))

ADT extension List

Use Boolean, Natural

Operations

isEmpty : List → Boolean

length : List → Natural

contains : List × Element → Boolean

Axioms

A_1 isEmpty(new) ≡ T

A_2 isEmpty(cons(e, l)) ≡ F

A_3 length(new) ≡ 0

A_4 length(cons(e, l)) ≡ length(l) + 1

A_4^* length(l) ≡ length(rest(l)) + 1

A_5 contains(new, e) ≡ F

A_6 contains(cons(e, l), e) ≡ T

A_6^* contains(l, first(l)) ≡ T

A_7 contains(cons(e_1, l), e_2) ≡ contains(l, e_2)

A_7^* contains(l, e) ≡ contains(rest(l), e)

contains((A B C), B)

A_7^* contains((B C), B)

A_6^* T

contains((A B C), D)

A_7^* contains((B C), D)

A_7^* contains((C), D)

A_7^* contains(new, D)

A_5 F

2nd extension with operations nth (nth(⟨A B C D⟩,3) ≡ C)

ADT extension List

Operations

nth: List x Natural → Elt

insert: List x Element x Natural → List

removeElt: List x Element → List

remove: List x Natural → List

insert (insert((A B C), D, 2) ≡ (A D B C)

removeElt(removeElt((A B C), B) ≡ (A C)

remove (remove((A B C), 2) ≡ (A C)

Preconditions

nth (l, n) iff $1 \leq n \leq \text{length}(l)$

insert (l, n, e)

remove (l, n)

+1

Axioms

A_2 $nth(cons(e, l), n) \equiv nth(l, n-1)$

A_2^* $nth(l, n) \equiv nth(rest(l), n-1)$

A_1 $nth(cons(e, l), 1) \equiv e$

A_1^* $nth(l, 1) \equiv first(l)$

$nth((A\ B\ C\ D), 3)$

A_2^* $nth((B\ C\ D), 2)$

A_2^* $nth((C\ D), 1)$

A_1^* $first((C\ D)) \equiv C$

$nth(cons(A, cons(B, cons(C, cons(D, nil)))), 3)$

A_2 $nth(cons(B, cons(C, cons(D, nil))), 2)$

$\vdots$

A_3^* insert $(l, e, 1) \equiv$ cons (e, l)

A_4^* ~~insert $(l, e, n) \equiv$ insert (rest (l), e, n-1)~~

A_4^* insert $(l, e, n) \equiv$ cons(first (l), insert (rest (l), e, n-1))

insert $((A\ B\ C), D, 3)$

A_4^* insert $((B\ C), D, 2)$

A_4^* insert $((C), D, 1)$

A_3^* cons $(D, (C)) \equiv (D\ C)$

no

A_4^* cons $(A,$ insert $((B\ C), D, 2))$

A_4^* cons $(A,$ cons $(B,$ insert $((C), D, 1)))$

A_3^* cons $(A,$ cons $(B,$ cons $(D, (C))))$

$(D\ C)$

$(B\ D\ C)$

$(A\ B\ D\ C)$

A_5 removeElt(newS, e) ≡ newS

A_6 removeElt(cons(e, l), e) ≡ removeElt(l, e)

A_6^* removeElt(l, first(l)) ≡ removeElt(rest(l), first(l))

A_7 removeElt(cons(e_1, l), e_2) ≡ cons(e_1, removeElt(l, e_2))

A_7^* removeElt(l, e) ≡ cons(first(l), removeElt(rest(l), e))

A_8^* remove(l, 1) ≡ rest(l)

A_9^* remove(l, n) ≡ cons(first(l), remove(rest(l), n − 1))

removeElt((A B C), B)

A_7 cons(A, removeElt((B C), B))

A_6^* cons(A, removeElt((C), B))

A_7 cons(A, cons(C, removeElt(newS, B)))

A_5 cons(A, cons(C, newS)) ≡ cons(A, (C))

≡ (A C)

Complexity of remove?

param : n (position in list)

unit op: cons

A_0: $C(1) = 0$

A_j: $C(n) = C(n-1) + 1$

$= ((C(n-2)) + 1) + 1$

$= C(n-2) + 2$

$= C(n-3) + 3$

$\vdots$

$= C(n-i) + i$

$= (0) + n - 1$

$= \mathcal{O}(n)$

param : n (length of list)

typ: all positions are equally likely

$p(i) = \frac{1}{n}$

$D(n) = \sum_{i=1}^{n} p(i)\, C(i)$

$= \frac{1}{n} \sum_{i=1}^{n} i - 1$

$= \frac{1}{n} \cdot \frac{(n-1)\,n}{2} = \frac{n-1}{2}$

$= \mathcal{O}(n)$

next: define an extension with operation merge (if is sorted)