Lecture 1: preliminaries, RAM, amortized complexity

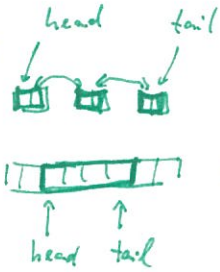
introduction - see www

Preliminaries

DS: { data + operations (queries / updates) "black-box" interface
implementation ⇒ complexity of operations

static DS (only queries) / dynamic

Examples: • Queue: (sequence of items) Enqueue(x) O(1) implementations:
Dequeue O(1) linked list
IsEmpty O(1) array



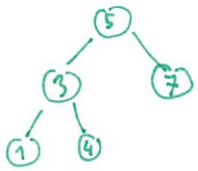
cyclically indexed

(• Stack: Push(x) Pop IsEmpty similar)

• Set: (of keys) — is there data? →

	linked list	array	sorted array	bin. search tree	hash table
Insert(x)	O(n)	O(1)	O(n)	O(log n)	O(1)
Delete(x)	O(n)	O(n)	O(n)	O(log n)	O(1)
Find(x)	O(n)	O(n)	O(log n)	O(log n)	O(1)
Build($x_1 \dots x_n$)	O(n)	O(n)	O(n log n)	O(n log n)	O(n)

BST:



⇑ average complexities

• modification: key + value (dictionary)

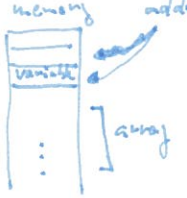
• ordered set:

	linked list	array	sorted array	bin. search tree	hashing
MIN / MAX	O(n)	O(n)	O(1)	O(log n)	O(n)
SUCC(x)	O(n)	O(n)	O(log n)	O(log n)	O(n)
PRED(x)	O(n)				

⇑ also also.

Model of computation

RAM - memory addresses (pointers)



- time = # instructions performed
- space = # cells used (between the smallest and largest accessed address)

~~empty cells between used cells~~

- memory allocator
- standard arithm. + logic instructions (comparisons)

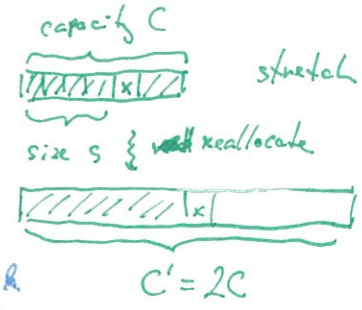
assumption: all integers used ≤ poly(len(I), max(I))

Amortized complexity – operations first

Aggregation method

Example: flexible array: APPEND(x)

$\Theta(n)$ worst-case time

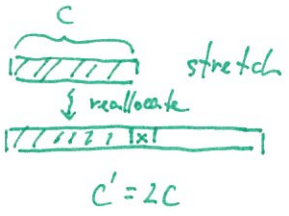


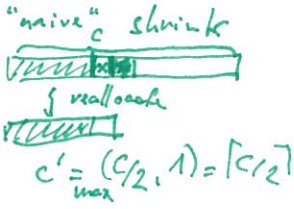
n items $\Rightarrow$ final capacity 2^k, $2^{k-1} < n \leq 2^k$

(aggregated) total time $= 2^0 + 2^1 + \dots + 2^{k-1} = 2^k - 1 = \Theta(n) \Rightarrow \Theta(1)$ amortized time

Accounting method

Ex. shrinkable array: APPEND(x) (stack) REMOVELAST



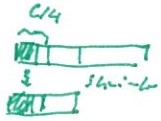


"naive" shrink if $n < C/2$ does not work:

for $n = C$ + APPEND(x), REMOVELAST 2x, APPEND(x')

$\Rightarrow \Theta(n)$ amortized time :(

"better": shrink if $n < C/4$, to $C' = C/2$



n operations, divided in blocks by reallocations

☺: after stretch/shrink we have $n = C/2 \pm 1$.

$\Rightarrow$ each block contains at least $C/4$ operations, reallocation cost $\Theta(C)$ accounted to operations in its block $\Rightarrow \Theta(1)$ amortized cost

(Total time is redistributed (accounted) between operations.)

Coin method

Ex. binary counter: INC (l-bits) ISZERO

$\Theta(l)$ worst case

☺ ith bit changes once in 2^{i-1} increments

initially 0...0, n INCs $\Rightarrow$ aggregation $= \sum_{i=1}^{l} \left\lfloor \frac{n}{2^{i-1}} \right\rfloor \leq \sum_{i=0}^{l-1} \frac{n}{2^i} < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$

$\Rightarrow \Theta(1)$ amortized time

Coin method (cont.)

paying in advance: 2 coins for each INC (one for $0 \to 1$, one as 1) — for time

$\Rightarrow \Theta(1)$ amortized time

Lecture 2

Potential method (general approach)

goal: compensate "hard" operations by "easy" ones.

n operations, real costs R_i (upper bounds), amortized costs A_i

after i steps/operations: $\Phi_i = \sum_{j=1}^{i} A_j - \sum_{j=1}^{i} R_j$ potential (bank account)

$\Phi_i \geq 0$ (never owe time)

$\Phi_0 = 0$ (naturally)

potential difference $\Delta\Phi_i = \Phi_i - \Phi_{i-1} = A_i - R_i$ — >0 saves time; $=0$ pays exact amount; <0 spends time — ith operation

Ex. binary counter: Φ_i = # of 1's in $(i)_2$

$A_i = 2$, $R_i = (\#$ of trailing 1's in $i-1) + 1$

$\Delta\Phi_i = 1 - t$ (t created, trailing 1's disappear)

dynamizable arrays: Φ_i = # of operations since the last reallocation.

$A_i = 2$ (1 for append / remove, 1 for potential)

$R_i = \begin{cases} 1 & \text{no reallocation} \\ 1 + \Theta(c) & \text{reallocation} \end{cases}$ (← idea of multiplying by a constant)

What is an amortized cost?

$A_i = R_i + \Delta\Phi_i$ — sum of real cost + change of potential

over n operations: $\sum_{i=1}^{n} A_i = \sum_{i=1}^{n} (R_i + \Phi_i - \Phi_{i-1}) = \sum_{i=1}^{n} R_i + \underbrace{\Phi_n - \Phi_0}_{\geq 0}$ — telescopic sum

$\Rightarrow$ $\{$ real costs $\}$ bounded by $\{$ amortized costs $\}$ ✓

Summary: 1) choose an appropriate potential Φ (i.e. counting something in the structure)

2) show $\Phi_n - \Phi_0 \geq 0$ (never owe time)

3) aim for $A_i = R_i + \Delta\Phi_i$ to be small

amortized vs. average complexity