Joshua Eckels · CSSE230

Sums

Geometric: $\sum_{i=0}^{n} a^i = 1 + a + a^2 \dots + a^n = \frac{1-a^{n+1}}{1-a}$

arithmetic: $\sum_{i=1}^{n} i = 1+2+3\dots+n = \frac{n(n+1)}{2}$

square: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Logs

$b^c = a^{c * \log_a(b)}$

$a^{\log_b(n)} = n^{\log_b(a)}$

Runtime

$f(n)$ is $O(g(n))$ iff $f(n) \le c\,g(n)$ on $n \ge n_0$

Max cont. Subseq. Sum

$\theta(n)$ $O(n)$ → upper $\Omega(n)$ → lower

Obs 1: $S_{i,j}$ cannot begin with prefix $S_{i,k}$ whose sum is negative

$\theta(n) \to O(n)$ & $\Omega(n)$

Obs 2: Every sum bordering a MCSS <0 MCSS <0 must be <0

Obs 3: $S_{i,j}$ just became negative, no MaxCSS starting in between $i+1$ and j

$n^3 + 20n + 1 \le cn^3$

$1 + \frac{20}{n^2} + \frac{1}{n^3} \le c$

as $n \to \infty$, $c \to 0$

$n \ge 1$

$c \ge 22$

ADT

ADT structure	find	insert/rem	comments
Array	O(n)	can't	O(1) access pos
Stack	O(1) top	O(1) top	imp → array
Queue	O(1) front	O(n)	FIFO
ArrayList	O(n), O(log n)	O(n)	add O(1), O(n) w
LinkedList	O(n)	O(1)	O(n) to find pos
HashSet/Map	O(1)	O(1), O(n)	not sorted
TreeSet/Map	O(log(n))	O(log n)	traverse in sorted order
Priority Queue	O(1)	O(log n)	find smallest only
Search tree	O(log n)	O(log n)	only if balanced, or O(n)

BST

T_L: BST or null

R: root

T_R

In order: $T_L \to R \to T_R$

Pre order: $R \to T_L \to T_R$

Post order: $T_L \to T_R \to R$

level order →

height(null) = -1

AVL

$S_h = 1 + S_{h-1} + S_{h-2}$

$S_h = F_{h+3} - 1$

h	0	1	2	3	4	5
S_h	1	2	4	7	12	20
F_h	0	1	1	2	3	5	8	13	21

$\frac{1}{\sqrt{5}}\phi_1^{h+3} - 1 \le N(t) \le 2^{h(t)+1} - 1$

$\lceil \log_2(N+1) \rceil - 1 \le h(t) \le \lfloor \log_{\phi_1}(\sqrt{5}(N(t)+1)) \rfloor - 3$

$\log_2 N - 1 \le h \le 1.44 \log_2 N - 1.33$

Hash table

$\lambda = \frac{n}{m} = \frac{\#\text{ of items}}{\text{table (array) cap.}}$

String: $x = 31x + y$

$A_3x^3 + A_2x^2 + A_1x^1 + A_0x^0 =$

$(((A_3)x + A_2)x + A_1)x + A_0$

guaranteed no duplicate probing for quadratic if $\lambda \le 0.5$, m = prime

Runtime: $O(\lambda)$

Expected # probes: $\frac{1}{1-\lambda}$ or $\frac{1}{2}\left(1 + \frac{1}{(1-\lambda)^2}\right)$ w/clustering

linear probe: $H+1, H+2 \dots$

quadratic: $H+1^2, H+2^2 \dots$

Red-black trees

a. BST

b. Red or black

c. root is black

d. No 2 successive red nodes

e. Every path ~ same # black nodes

f. use top-down: flip first, rotate once at most

worst case:

- n nodes
- $\lfloor \frac{n+1}{2} \rfloor$ black nodes
- fall to $\frac{h+1}{2}$
- h

shortest path is $\lfloor \frac{h+1}{2} \rfloor$ black nodes

$2^{h/2} - 1 \le N(t) \le 2^{h(t)+1} - 1$

$\lceil \log_2(N(t)+1) \rceil - 1 \le h(t) \le 2\log_2(N(t)+1)$

Priority Queue

	insert	find min	delete MIN
unsorted array	$\theta(1)/(n)$	$\theta(n)$	$\theta(n)$
reverse sorted array	$\theta(n)$	$\theta(1)$	$\theta(1)/(n)$
AVL tree	$\theta(\log n)$		
Binary Heap	$\theta(\log n)/(n)$	$\theta(1)$	$\theta(\log n)/(n)$

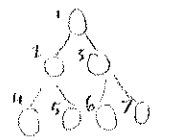
Binary Heap

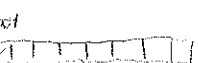
left = 2i, right = 2i+1

parent = $\lfloor i/2 \rfloor$

insert → at end → perc up

Delete → delete root → move last element to hole → percolate down





Sort

	A (insert into data structure)	B (extract)
Treesort	insert into AVL tree $\theta(N \log N)$	inorder traversal $\theta(N)$
Naive Heapsort	Start w/empty heap, .insert() all items $\theta(N \log N)$	run deleteMin N times → copy to array $\theta(N \log N)$
Heapsort w/buildHeap	Fix heap order w buildHeap $\theta(N) = N - H + 1$ ✗	$\theta(N \log N)$

$\sum_{k=1}^{H} 2^{H-k} \cdot K$

Graphs

$\binom{n}{k} = \frac{n!}{k!(n-k)!}$

size: $N = |V|$

$0 \le |E| \le \binom{|V|}{2}$

$0 \le E \le \frac{(V)(V-1)}{2}$ → undirected

$0 \le E \le (V)(V-1)$ → directed

$N-1 \le E \le \frac{N(N-1)}{2}$ → connected

$\sum \deg(v) = 2|E|$ → undirected

connected: path exists from V

tree: connected acyclic: $|V|-1 = |E|$

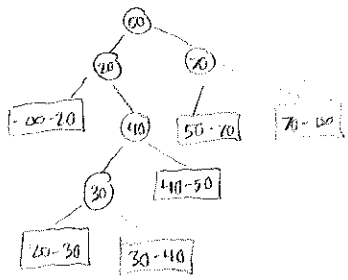
strongly connected: 2 nodes are mutually reachable

Component: every node reachable from every other node

	Runtime	Adj Matrix	Adj List
	degree(V)	$\theta(N)$	$\theta(1)$
	delete Edge	$\theta(1)$	$\theta(N)$ or $\theta(\deg(u) + \deg(v))$
space efficiency		$\theta(N^2)$	$\theta(E)$

$E = \theta(N^2), \Omega(N)$ $\quad N-1 \le E \le N(N-1)$

Extended Binary Tree (EBT)



$EN(T) = IN(T) + 1$

$EPL(T) = IPL(T) + 2 \cdot IN(T)$

$EPL(T) = EN(T_L) + EN(T_R) + EPL(T_L) + EPL(T_R)$

Recurrence

Runtime = (Recursive runtime) + (non-recursive runtime)

forward substitution

$$\begin{cases} T(0) = 0 \\ T(N) = 2 + T(N-1) \end{cases}$$

$T(0) = 0$
$T(1) = 2$
$T(2) = 4$
$T(3) = 6$
$T(N) = 2N = \theta(N)$

backward substitution

$$\begin{cases} T(0) = 1 \\ T(N) = 2T(N-1) \end{cases}$$

$T(N) = 2T(N-1)$
$= 2(2T(N-2))$
$= 2^3 T(N-3)$
$= 2^N (T(N-N))$
$= 2^N T(0) = 2^N$

telescoping

$T(1) = 1 \quad T(N) = T(N-1) + 2$

	i
$T(N) = T(N-1) + 2$	0
$T(N-1) = T(N-2) + 2$	1
$T(N-2) = T(N-3) + 2$	2
$T(N-k) = T(N-(k+1)) + 2$	k

$T(1) = 1$ ✗ $T(1) = T(N-(2k+1))$

$\Sigma lhs = \Sigma rhs$

$K = \frac{N-3}{2}$

$T(N) = \sum_{i=0}^{k} 2 + 1$

Master theorem

$T(N) = a\,T(N/b) + \theta(N^k)$

$$T(N) = \begin{cases} \theta(N^{\log_b a}) & a > b^k \rightarrow \text{dominated by leaves} \\ \theta(N^k \log N) & a = b^k \rightarrow \text{works stays the same} \\ \theta(N^k) & a < b^k \rightarrow \text{dominated by root} \end{cases}$$

QuickSort

1. select pivot point
2. while A[i] < pivot {i++}
3. while A[j] > pivot {j--}
4. swap i and j
5. break when $j <= 0$
6. swap pivot into j location

merge

1. merge $(A[0..\frac{n}{2}])$
2. merge $(\frac{n}{2}+1 \ldots n-1)$
3. merge 2 halves

Selection

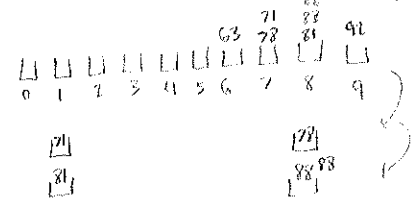
1. find max, put in A[i-1]
2. recursively sort A[0...n-2]

Insertion

1. Recursively sort A[0...n-1]
2. Insert A[n-1] into ...

Bucket sort

78, 92, 81, 88, 63, 71, 88 → bucket by LSD recursively

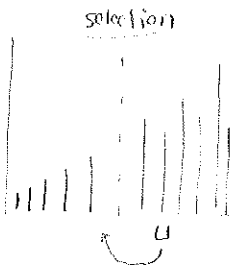


go until all buckets have only 1 value in them

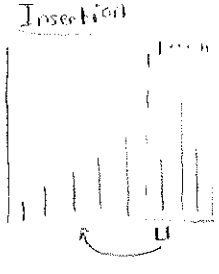
* Sorts by value

Radix sort

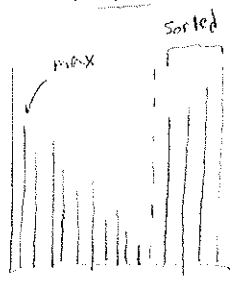
1. compute sizes of each bucket based on count of LSD's
2. Compute index of the end of each bucket
3. fill in buckets right to left
4. when buckets are full, repeat for next most sig digit
5. #bucketing steps: $k = \lfloor \log_b M \rfloor + 1$
   - → b = base (eg base 10 for decimal)
   - → M = max value of input array
6. O(kn), good when range is small, lots of repetition

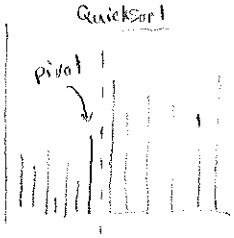


- select from unsorted and place at end of sorted
- loops over unsorted part



- inserts 1st item of unsorted into correct spot in sorted
- loops over sorted part





	Best	worst	Avg
selection	$\theta(n^2)$	n^2	n^2
insertion	n	n^2	n^2
quick	$n \log n$	n^2	$n \log n$
merge	$n \log n$	$n \log n$	$n \log n$
bucket	nk	n^2	nk
radix	n	n^2	$n \log n$

(nk)