Logarithm Rules and Properties

1. $\log_a 1 = 0$
2. $\log_a a = 1$
3. $\log_a x^y = y \log_a x$
4. $\log_a xy = \log_a x + \log_a y$
5. $\log_a \frac{x}{y} = \log_a x - \log_a y$
6. $a^{\log_b x} = x^{\log_b a}$
7. $\log_a x = \frac{\log_b x}{\log_b a} = \log_a b \log_b x$

Summations for Printing

Useful Formulae

Sum	Closed Form
$\sum_{k=0}^{n} ar^k; r \neq 0$	$\frac{ar^{n+1} - a}{r - 1}; r \neq 1$
$\sum_{k=1}^{n} k$	$\frac{n(n+1)}{2}$
$\sum_{k=1}^{n} k^2$	$\frac{n(n+1)(2n+1)}{6}$
$\sum_{k=1}^{n} k^3$	$\frac{n^2(n+1)^2}{4}$
$\sum_{k=0}^{\infty} x^k; \lvert x \rvert < 1$	$\frac{1}{1-x}$
$\sum_{k=1}^{\infty} kx^{k-1}; \lvert x \rvert < 1$	$\frac{1}{(1-x)^2}$

Summation Rules

1. $\sum_{i=l}^{u} ca_i = c \sum_{i=l}^{u} a_i$
2. $\sum_{i=l}^{u} (a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i$
3. $\sum_{i=l}^{u} a_i = \sum_{i=l}^{m} a_i + \sum_{i=m+1}^{u} a_i$, where $l \leq m < u$
4. $\sum_{i=l}^{u} (a_i - a_{i-1}) = a_u - a_{l-1}$

Don`t panic

Insertion Sort - Algorithm

1 3 7 2 0	[1 3 7]2 0
[1]3 7 2 0	[1 2 3 7] 0
[1 3]7 2 0	[0 1 2 3 7]

Starting from one end, ensure the sub-array sorted in each pivot

Selection sort - Algorithm

1 3 7 2 0	[0 1 2]3 7
[0]1 3 7 2	[0 1 2 3]7
[0 1]3 7 2	[0 1 2 3 7]

Starting from one end, ensure the sub-array sorted in each pivot

Bubble sort - Algorithm

1 3 7 2 0	1 2(0 3)7
1 3(2 7)0	1(0 2)3 7
1 3 2(0 7)	(0 1)2 3 7
1(2 3)0 7	0 1 2 3 7

Binary comparison and swap, shift each pivot by at most 1 position in an iteration

Sorting Complexities

Algorithm	Time		Space
	Best	**Worst**	**Worst**
Bubble Sort	O(n)	O(n^2)	O(1)
Insertion Sort	O(n)	O(n^2)	O(1)
Selection Sort	O(n^2)	O(n^2)	O(1)
Quicksort	O(n log(n))	O(n^2)	O(log(n))
Heapsort	O(n log(n))	O(n log(n))	O(1)
Mergesort	O(n log(n))	O(n log(n))	O(n)

Asymptotic Notations

Big-O

T(n) ∊ O(f(n)) if there are constants c > 0 and n0such that T(n) ≤c f(n) for all n ≥n0

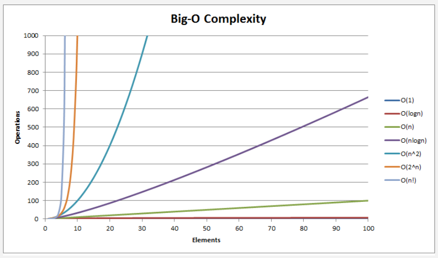
Big-Omega

T(n) ∊ Ω(f(n)) if f(n) ∊O(T(n))

Big-Theta

T(n) ∊ Θ(f(n)) if T(n) ∊O(f(n)) and T(n) ∊ Ω(f(n))

Big-O Complexity Chart



Queue ADT

Queue property

FIFO: First In First Out

Core operations

- enqueue – dequeue – is_empty

Priority Queue ADT

Queue property

Lower Priority Value Out First

Core operations

- insert –deleteMin –isEmpty

Stack ADT

Stack property

LIFO: Last In First Out

Core operations

– push – pop – top – is_empty

d-Heap ADT

child	(i-1)*d+2 ~ i*d+1
parent	⌊(i-2)/d⌋+1
root	1
next free	size+1

(Min) Heap Tree - ADT

Heap-order property

parent's key <= children's keys

Structure property

nearly complete tree

Heapify algorithm

Heapify the tree from bottom up, percolate DOWN a node as deep as needed for each node.

Binary Heap Operations Complexity:
Heapify - O(n)
Find Min - O(1)
Insert - O(log(n))
Delete - O(log(n))

Loop -> (Tail) Recursion

```
//Loop
int i = 0;
while (i < n)
    doF oo(i);
   i++;
//Recu rsion
void recDoF oo(int i, int n){
   if (i < n) {
        doF oo(i);
        rec DoFoo(i + 1, n);}
}
recDoF oo(0, n);
```

Equivalent for loop:
for (int i=0; i<n; i++) doFoo(i);

Tail Recursion -> Iteration

```
//Tail Recursion
int fact_acc (int n, int acc) {
   if (n)
        return fact_acc(n -1,
acc * n);
    return acc;
}
//Iter ation
int fact_acc (int n, int acc) {
    for (;n ;n--)
       acc = acc * n;
return acc;
}
```

Recurrence Simplification Strategy

1) Find T(n) for the base cases.

2) Expand T(n) for the general patterns.

3) Drive the recursive term to the base case.

4) Solve and represent k with n by inequality(equality).

5) Substitute back to T(n).

e.g.
T[n <= n0] = C1
T[n]= T[n/2]+C2
-> T[n] = T[n*(1/2)]+C2
-> T[n] = T[n*(1/2)^k] + k*C2

Let n*(1/2)^k <= n0 -> T(n*(1/2)^k) = C1

-> n*(1/2)^k * (2)^k <= n0* (2)^k
-> n/n0 <= (2)^k
-> log2(n/n0) <= log2((2)^k)
-> log2(n/n0) <= k

-> T[n] = C1+k*C2 = C1 + C2*log2(n/n0)

Summation Fomulas

1. $\sum_{i=l}^{u} 1 = \underbrace{1+1+\cdots+1}_{u-l+1 \text{ times}} = u - l + 1$ (l, u are integer limits, $l \le u$); $\sum_{i=1}^{n} 1 = n$
2. $\sum_{i=1}^{n} i = 1+2+\cdots+n = \frac{n(n+1)}{2} \approx \frac{1}{2}n^2$
3. $\sum_{i=1}^{n} i^2 = 1^2+2^2+\cdots+n^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{1}{3}n^3$
4. $\sum_{i=1}^{n} i^k = 1^k+2^k+\cdots+n^k \approx \frac{1}{k+1}n^{k+1}$
5. $\sum_{i=0}^{n} a^i = 1+a+\cdots+a^n = \frac{a^{n+1}-1}{a-1}$ ($a \ne 1$); $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$
6. $\sum_{i=1}^{n} i2^i = 1\cdot 2 + 2\cdot 2^2 + \cdots + n2^n = (n-1)2^{n+1}+2$
7. $\sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \cdots + \frac{1}{n} \approx \ln n + \gamma$, where $\gamma \approx 0.5772\ldots$ (Euler's constant)
8. $\sum_{i=1}^{n} \lg i \approx n \lg n$