Time-Complexity notation
$O = 0 \le f(n) \le cg(n), o = 0 \le f(n) < cg(n)$
$\Omega = 0 \le cg(n) \le f(n), \omega = 0 < cg(n) \le f(n)$

Master Method
Case 1: $F(n) = O\left(n^{\log_b a - \epsilon}\right)$ then, $T(n) = \Theta\left(n^{\log_b a}\right)$
Case 2: $F(n) = O\left(n^{\log_b a} \log^k(n)\right)$ then, $T(n) = \Theta\left(n^{\log_b a} \log^{k+1}(n)\right)$
Case 3: $F(n) = O\left(n^{\log_b a + \epsilon}\right)$ then, $T(n) = \Theta\left(n^{\log_b a}\right)$

Red-Black: Search=Insertion=Deletion=Color Changes $O(\log n)$, Rotations $O(1)$

Heaps
Top Down $O(n \log n)$, Bottum Up $O(n)$
Heapification: $O(n)$, Update=Extract-Minimum: $O(\log n)$, Minimum-Index=Get-Minimum=Get-Key: $O(1)$

Sorting
Bucket Sort: $O(n+m)$, space $O(m)$
Radix Sort: $O(kn)$
Countering Sort: $O(n)$
Insertion Sort: Data Movement $O(n^2)$, Comparisons $O(n \log n)$.
Selection Sort: Data Movement $O(n)$, Comparisons $O(n^2)$.
Merge Sort: $O(n \log n)$
Quick Sort: $O(n^2)$

Union Find union: $O(1)$, find: $O(\log n)$
Huffman Codes: $O(n \log n)$
Selection: $O(n)$

String Matching: $O(n+m)$

Graphs
DFS and BFS: list $O(|V| + |E|)$, Matrix $O(V^2)$
Bellman-Fort: $O(|V||E|)$, works for everything but negative cycles.
Acyclic: $O(|V| + |E|)$, doesn't have cycles, works with negative weights has to sorted in a topological order.
Dijkstra: Binary heap $O\big((|V| + |E|) \log |V|\big)$, delete $O(|V| \log |V|)$, update $O(|E| \log |V|)$
Fibonacci heap $O(|V| \log(|V|) + |E|)$, delete $O(|V| \log |V|)$, update $O(|E|)$
No heap $O(|V|^2)$, delete $O(|V||V|)$, update $O(|E|)$
Prims: $O\big((|V| + |E|) \log |V|\big)$, delete $O(|V| \log |V|)$, update $O(|E| \log |V|)$
Kruskal: $O\big((|V| + |E|) \log |V|\big)$, heap operations $O\big((|V| + |E|) \log |V|\big)$, union find $O(|E| \log |V|)$
Floyd-Warshall: $O(|V|^3)$, no negative cycle $O\big((|V|^2 |E|)\big)$, no negative weights $O(|V|^2 \log(|V|) + |V||E|)$

Max Flow:
Ford-Fulkerson: $O(|E||f|)$
Edmonds-Krap: $O(|V||E|^2)$

P, NP, NPC
P = Polynomial Time Solvable, Nondeterministic Polynomial Time Solvable.
$O(n^k)$, as long as k is a constant then, the problem is P.
A problem is called NP-Hard if every problem is reducible to that problem.
A problem is called NO-Complete if the problem belong to NP, and it is an NP-Hard problem.

Counting-Sort(A, B, k)
for i=0 **to** k **do** C[i]=0
for j=1 **to** n **do** C[A[j]0]+=1
for i=1 **to** k **do** C[i]+=C[i-1]

Compute-Next(P, m)
next[1]=0, q=0
for i = 2 **to** m **do**
    while q>0 **and** P[i]!=P[q+1] **do** q = next(q)
    if P[i]==P[q+1] **then** q++
    next[i]=q;

String-Matching(T, n, P, m) $\boldsymbol{O(n)}$
i ← 1, q ← 0
while i <= n **do**
if T[i] = P[q + 1] **then** i ← i + 1, q ← q + 1
else
    if q = 0 **then** i ← i + 1
    else q = next [q]
if q = m **then**
    print "pattern found at position" i – m
    q = next [q]

Bellman-Ford(G, w, s)
Initialize-Single-Source(G, s)
for i=1 **to** V – 1 **do**
    for each edge (u, v) that belongs to E **do**
        Realx(u, v, w)
    for each edge (u, v) that belongs to E **do**
        If d[v]>d[u]+w(u, w) **then return** False