Useful Information

DIJKSTRA(G, w, s)
INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
$Q = G.V$
while $Q \neq \emptyset$
  u = EXTRACT-MIN(Q)
  $S = S \cup \{u\}$
  for each $v \in G.Adj[u]$)
    RELAX(u, v, w)

FLOYD-WARSHALL(W, n)
let $D^{(0)} = W$
for $k = 1$ to n
  let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
  for $i = 1$ to n
    for $j = 1$ to n
      $d_{ij}^{(k)} = min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $D^{(n)}$

DFS(G)
for each $u \in G.V$
  u.color = WHITE
time = 0
for each $u \in G.V$
  if u.color == WHITE
    DFS-VISIT(G,u)

DFS-VISIT(G, u)
time = time +1
$u.d$ = time
u.color = GRAY
for each $v \in G.Adj[u]$
  if v.color == WHITE
    DFS-VISIT(G, v)
u.color = BLACK
time = time +1
$u.f$ = time

BFS(V, E, s)
for each $u \in V - s$
  $u.d = \infty$
$s.d = 0$
$Q = \emptyset$
ENQUEUE(Q, s)
while $Q \neq \emptyset$
  u =DEQUEUE(Q)
  for each $v \in G$.Adj$[u]$
    if $v.d == \infty$
      $v.d = u.d + 1$
      ENQUEUE(Q, v)

QUICKSORT(A, p, r)
1 if $p < r$
  q =PARTITION(A, p, r)
  QUICKSORT$(A, p, q-1)$
  QUICKSORT$(A, q+1, r)$

INIT-SINGLE-SOURCE(G, s)
for each $v \in G.V$
  $v.d = \infty$
  $v.\pi = NIL$
$s.d = 0$

RELAX(u, v, w)
if $v.d > u.d + w(u, v)$
  $v.d = u.d + w(u, v)$
  $v.\pi = u$

MEMOIZED-CUT-ROD(p, n)
let $r[0..n]$ be a new array
for $i = 0$ to n
  $r[i] = -\infty$
return MEMOIZED-CUT-ROD-AUX(p, n, r)

MEMOIZED-CUT-ROD-AUX(p, n, r)
if $r[n] \geq 0$
  return $r[n]$
if $n == 0$
  $q = 0$
else $q = -\infty$
  for $i = 1$ to n
    q=max$(q, p[i]$+MEMOIZED-CUT-ROD-AUX$(p, n-i, r))$
$r[n] = q$
return q

PRIM(G, w, r)
$Q = \emptyset$
for each $u \in G.V$
  u.key = inf
  $u.\pi$ = NIL
  INSERT(Q, u)
DECREASE-KEY$(Q, r, 0)$ //r.key = 0
while $Q \neq \emptyset$
  u = EXTRACT-MIN(Q)
  for each $v \in G$.Adj$[u]$
    if $v \in Q$ and $w(u, v) < v$.key
      $v.\pi = u$
      DECREASE-KEY$(Q, v, w(u, v))$

KRUSKAL(G, w)
$A = \emptyset$
for each vertex $v \in G.V$
  MAKE-SET (v)
sort the edges of $G.E$ by weight w
for each (u, v) taken from the sorted list
  if FIND-SET$(u) \neq$FIND-SET(v)
    $A = A \cup \{(u, v)\}$
    UNION(u, v)
return A

HUFFMAN(C)
$n = |C|$
$Q = C$
for $i = 1$ to $n - 1$
  allocate a new node z
  z.left = x = EXTRCT-MIN(Q)
  z.right = y = EXTRCT-MIN(Q)
  z.freq = x.freq + y.freq
  INSERT(Q, z)
return EXTRACT-MIN(Q)

- The (simplified) Master Method for when $a \geq 1$, $b > 1$ and $c \geq 0$:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ aT(n/b) + \Theta(n^c) & \text{otherwise} \end{cases}$$

  1. if $\log_b a > c$ then $T(n) = \Theta(n^{\log_b a})$
  2. if $\log_b a = c$ then $T(n) = \Theta(n^c \log(n))$
  3. if $\log_b a < c$ then $T(n) = \Theta(n^c)$

- The Master Method for when $a \geq 1$, $b \geq 2$ and $f(n) \geq 0$:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ aT(n/b) + f(n) & \text{otherwise} \end{cases}$$

  1. if $f(n)$ is $O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{log_b a} \log^k n)$ constant $k \geq 0$ then $T(n) = \Theta(n^{\log_b a} \log^{k+1}(n))$
  3. if $f(n)$ is $\Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$

- Asymptotic Formulas:
  - $O(g(n)) = \{f(n)$: there exists positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$
  - $\Omega(g(n)) = \{f(n)$: there exists positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$
  - $\Theta(g(n)) = \{f(n)$: there exists positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$
  - $o(g(n)) = \{f(n)$: for all positive constants c there exists a c constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$
  - $\omega(g(n)) = \{f(n)$: for all positive constants c there exists a c constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$

MEMORIZED-MATRIX-CHAIN(p)

```
n = p.length - 1
let m[1..n, 1..n] be a new table
for i = 1 to n
  for j = i to n
    m[i, j] = ∞
return LOOKUP-CHAIN(m, p, 1, n)
```

LOOKUP-CHAIN($m, p, 1, n$)

```
if m[i, j] < ∞
  return m[i, j]
if i == j
  m[i, j] = 0
else fork = i to j - 1
  q = LOOKUP-CHAIN(m, p, i, k)
    + LOOKUP-CHAIN(m, p, k + 1, j) + p_{i-1} p_k p_j
  if q < m[i, j]
    m[i, j] = q
return m[i, j]
```

LCS-LENGTH(X, Y, m, n)

```
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
  c[i, 0] = 0
for j = 0 to n
  c[0, j] = 0
for i = 1 to m
  for j = 1 to n
    if x_i = y_j
      c[i, j] = c[i - 1, j - 1] + 1
      b[i, j] = "↖"
    else if c[i - 1, j] ≥ c[i, j - 1]
      c[i, j] = c[i - 1, j]
      b[i, j] = "↑"
    else
      c[i, j] = c[i, j - 1]
      b[i, j] = "←"
return c and b
```