T(n) = aT(n/2) + O(n^d)
RSA:
y = x^e mod N
x = y^d mod N
d = e mod(p-1)(q-1)

(tn tn-1 …)(b^n)(p(n))
(...)(r-b)$^{d+1}$

function multiply(x, y)
Input: Two n-bit integers x and y, where $y \geq 0$
Output: Their product

if $y = 0$: return 0
$z =$ multiply$(x, \lfloor y/2 \rfloor)$
if y is even:
    return $2z$
else:
    return $x + 2z$

function modexp(x, y, N)
Input: Two n-bit integers x and N, an integer expon

Output: $x^y \bmod N$

if $y = 0$: return 1
$z =$ modexp$(x, \lfloor y/2 \rfloor, N)$
if y is even:
    return $z^2 \bmod N$
else:
    return $x \cdot z^2 \bmod N$

function extended-Euclid(a, b)
Input: Two positive integers a and b with $a \geq b \geq 0$
Output: Integers x, y, d such that $d = \gcd(a, b)$ and $ax + by = d$

if $b = 0$: return $(1, 0, a)$
$(x', y', d) =$ Extended-Euclid$(b, a \bmod b)$
return $(y', x' - \lfloor a/b \rfloor y', d)$

function primality(N)
Input: Positive integer N
Output: yes/no

Pick a positive integer $a < N$ at random
if $a^{N-1} \equiv 1 \pmod{N}$:
    return yes
else:
    return no

$f =$ (
$f = \Omega(g)$ **means** $g = O(f)$
$f = \Theta(g)$ **means** $f = O(g)$ **and** $f = \Omega(g)$.

procedure prim(G, w)
Input: A connected undirected graph $G = (V, E)$ with edge weights w_e
Output: A minimum spanning tree defined by the array prev

for all $u \in V$:
    cost$(u) = \infty$
    prev$(u) =$ nil
Pick any initial node u_0
cost$(u_0) = 0$

$H =$ makequeue(V) (priority queue, using cost-values as keys)
while H is not empty:
    $v =$ deletemin(H)
    for each $\{v, z\} \in E$:
        if cost$(z) > w(v, z)$:
            cost$(z) = w(v, z)$
            prev$(z) = v$
            decreasekey(H, z)

procedure kruskal(G, w)
Input: A connected undirected graph $G = (V, E)$ with edge weights w_e
Output: A minimum spanning tree defined by the edges X

for all $u \in V$:
    makeset(u)

$X = \{\}$
Sort the edges E by weight
for all edges $\{u, v\} \in E$, in increasing order of weight:
    if find$(u) \neq$ find(v):
        add edge $\{u, v\}$ to X
        union(u, v)

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a\,. \end{cases}$$

procedure dijkstra(G, l, s)
Input: Graph $G = (V, E)$, directed or undirected;
    positive edge lengths $\{l_e : e \in E\}$; vertex $s \in V$
Output: For all vertices u reachable from s, dist(u) is set
    to the distance from s to u.

for all $u \in V$:
    dist$(u) = \infty$
    prev$(u) =$ nil
dist$(s) = 0$

$H =$ makequeue(V) (using dist-values as keys)
while H is not empty:
    $u =$ deletemin(H)
    for all edges $(u, v) \in E$:
        if dist$(v) >$ dist$(u) + l(u, v)$:
            dist$(v) =$ dist$(u) + l(u, v)$
            prev$(v) = u$
            decreasekey(H, v)

procedure shortest-paths(G, l, s)
Input: Directed graph $G = (V, E)$;
    edge lengths $\{l_e : e \in E\}$ with no negative cycles;
    vertex $s \in V$
Output: For all vertices u reachable from s, dist(u) is set
    to the distance from s to u.

for all $u \in V$:
    dist$(u) = \infty$
    prev$(u) =$ nil

dist$(s) = 0$
repeat $|V| - 1$ times:
    for all $e \in E$:
        update(e)