```
BinarySearch(A[0..N-1], value, low, high) // Worst case O(logn)
{
  if (high < low)
      return not_found // value would be inserted at index "low"
  mid = (low + high) / 2
  if (A[mid] > value)
      return BinarySearch(A, value, low, mid-1)
  else if (A[mid] < value)
      return BinarySearch(A, value, mid+1, high)
  else
      return mid
}

minMax(A[0..N-1]) // Worst case O(n)
{
  if |A| = 1 return min=max=A[0]
  halve A into two subsets A1 & A2
  (min_1,max_1) = minMax(A1)
  (min_2,max_2) = minMax(A2)
  If min_1 ≤ min_1 then min = min_1
  Else min = min_2
  If min_1 ≥ max_2 then max = max_1
  Else max = max_2
}

mergeSort(A[0..N-1]) // Worst case O(nlogn)
{
  if |A| = 1 or 0 return A
  Q1 = mergesort(firstHalf(A))
  Q2 = mergesort(secondHalf(A))
  return merge(Q1, Q2)
}

solveHanoi(disk, source, dest, spare) // Worst case O(2^n)
{
  if disk = 0 then
   move disk from source to dest
  else
  solveHanoi(disk-1, source, spare, dest)
  move disk from source to dest
  solveHanoi(disk-1, spare, dest, source)
}
```

Dynamic Programming:
Product-Sum optimization formula: j is the number of elements and v_j is the value at j.

$$OPT[j] = \begin{cases} \max\{OPT[j-1] + v_j, OPT[j-2] + v_j * v_{j-1}\} & \text{if } j \geq 2 \\ v_1 & \text{if } j = 1 \\ 0 & \text{if } j = 0 \end{cases}$$

```
function coinChange(V, A) // Runtime: O(A*n) minimum number of coins to form change
{
  array total[A]
  set_all_values_greater_than_0_to_infinity(total)

  for i = 0 to length(V), i++
    for j = 0 to A-1, j++
      if V[i] <= j
        total[j] = min(total[j], 1+total[j-V[i]]) //DP opt formula
  return total[A-1]
}
```

Master Theorem

$$T(n) \leq \begin{cases} C \text{ if } n \leq 1 \\ a\,T(\frac{n}{b}) + f(n) \text{ if } n > 1 \end{cases} \qquad f(n) = n^d$$

$O(n^d)$ $d > \log_b(a)$

$O(n^d \log(n))$ $d = \log_b(a)$

$O(n^{\log_b a})$ $d < \log_b(a)$

Master Theorem

$$T(n) \leq \begin{cases} C \text{ if } n \leq 1 \\ aT(n-b) + f(n) \text{ if } n > 1 \end{cases} \qquad f(n) = n^d$$

$O(n^d)$ $a < 1$

$O(n^{d+1})$ $a = 1$

$O(n^d a^{\frac{n}{b}})$ $a > 1$

Big O Relationships

$A = \Theta(B)$ A & B have same growth rate

$A = O(B)$ A grows slower than B

$A = \Omega(B)$ A grows faster than B

ORdering from fastest to slowest

Fast Growth

Exp	$O(n^n)$
Factorial	$O(n!)$
Poly	$O(c^n)$
Exp	$O(n^c)$
Log	$O(n \log n)$
Linear	$O(n)$
Log	$O(\log n)$
Const.	$O(1)$

Slow Growth

$f(n)$	$g(n)$	$f=O(g)$	$f=\Omega(g)$	$f=\Theta(g)$
$n^{0.25}$	$n^{0.5}$	$O(g)$	—	—
n	$\log^2 n$	—	$\Omega(g)$	—
$\log n$	$\ln n$	$O(g)$	$\Omega(g)$	$\Theta(g)$
$1000n^2$	$0.0002n^2 + 1000n$	$O(g)$	$\Omega(g)$	$\Theta(g)$
$n \log n$	$n\sqrt{n}$	$O(g)$	—	—
e^n	3^n	$O(g)$	—	—
2^n	2^{n+1}	$O(g)$	$\Omega(g)$	$\Theta(g)$
2^n	2^{2^n}	$O(g)$	—	—
2^n	$n!$	$O(g)$	—	—
$\lg n$	$\sqrt{n}$	$O(g)$	—	—
$\log(n^3)$	$\log n + 5$	$O(g)$	$\Omega(g)$	$\Theta(g)$
4^n	$2^{2n} = 4^n$	$O(g)$	$\Omega(g)$	$\Theta(g)$
4^n	2^{n+1}	—	$\Omega(g)$	—
4^n	2^{n^2}	$O(g)$	—	—

$$\log_b a = \frac{\log a}{\log b}$$

1. Show $\log(n!) = O(n \log n)$

$\log(n!) = \log(n) + \log(n-1) + \log(n-2) + \log(n-3) \ldots \log(1)$ The sum of this is $< n \log n$, $\therefore$ $\log(n!) = \Theta(n \log n)$ $= O(n \log n)$

2. True/False: If $f_1(n) = O(g(n))$ and $f_2(n) = O(g(n))$ then $f_1(n) = \Theta(f_2(n))$

False $f_1(n) = 3^n$ $f_2(n) = 4^n$ $g(n) = 6^n$

3. True/False: $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_1(n) = O(\max(g_1(n), g_2(n)))$

True Let c_1 & c_2 be 2 constants ; for a large n $f_1(n) \le c_1 g_1(n)$ $f_2(n) \le c_2 g_2(n)$; $f_1(n) + f_2(n) \le c_1 g_1(n) + c_2 g_2(n)$; $f_1(n) + f_2(n) \le c_1 + c_2 \max(g_1(n), g_2(n))$

4. Sort as many classes into fewest rooms possible with respect to time (greedy)
- sort classes in increasing order of start time
- for each class, if start is compatable with last class' finish time in same room K schedule class
- else if not compatable, make new room $K = K + 1$
- repeat until all scheduled

5. Knapsack

6. Fractional Knapsack

7. Rod Cutting

8. Machine Scheduling, complete n tasks that have start s_i and end f_i time using minimum machines.

9. Along the trip to hotel n, there are $a_1 < a_2 < \cdots < a_n$ hotels you can stop at along the way. Travel penalty is $(200 - x)^2$. Find algo: minimum penalty for optimum hotel stop sequence.

10. $p_1, p_2 \ldots p_n$ projects with duration $d_1, d_2 \ldots d_n$ and pay $f_1, f_2 \ldots f_n$. No pay for p_i incomplete by d_i, no multitasking. Select subset S of proj that will maximize f earned in d days.

11. Find cheapest sequence to rent canoe from post $1, 2, \ldots n$. $R[i,j]$ gives cost to rent canoe from post i to j.