Types of Problems

1. Sorting
   - Evaluate: no. of key comparisons
   - Stability: preserve relative order of two equal elements
   - In-place: no extra memory
2. Searching
3. String matching
4. Graph problems
   $G = (V, E)$
5. Combinatorial
6. Geometric
7. Numerical

Theoretical Analysis: Time Efficiency

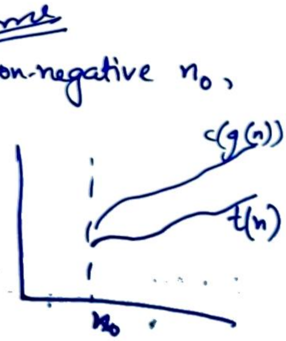
$$\boxed{T(n) \approx c_{op} \cdot C(n)}$$

order of growth: leading term, ignore constant coeff.

Asymptotic Notation

O: $t(n) \in O(g(n))$ if for any +ve c and some non-negative n_0,

$$\boxed{t(n) \le c \cdot g(n) \quad \text{for } n \ge n_0}$$



Θ: $t(n) \in \Theta(g(n))$ if

$$\boxed{c_1 \cdot g(n) \le t(n) \le c_2 \cdot g(n) \quad \text{for } n \ge n_0}$$

Ω: $t(n) \in \Omega(g(n))$ if

$$\boxed{t(n) \ge c \cdot g(n) \quad \text{for } n \ge n_0}$$

o: $\boxed{t(n) < c \cdot g(n) \quad \forall\, n \geq n_0}$

ω: $\boxed{t(n) > c \cdot g(n) \quad \forall\, n \geq n_0}$

Theorem:

$$t_1(n) \in O(g_1(n)),\ t_2(n) \in O(g_2(n))$$

$$\Rightarrow \boxed{t_1(n) + t_2(n) \in O\left(\max\{g_1(n), g_2(n)\}\right)}$$

Same for others

Method of evaluation with limits

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \begin{cases} 0, & t(n) < g(n) \longrightarrow t(n) \in O(g(n)) \\ c, & t(n) = g(n) \longrightarrow t(n) \in \Theta(g(n)) \\ \infty & t(n) > g(n) \longrightarrow t(n) \in \Omega(g(n)) \end{cases}$$

L'Hôpital's Rule: $\lim_{n\to\infty} \frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{t'(n)}{g'(n)}$

Stirling's Approximation: $n! \approx \left(\sqrt{2\pi n}\right)\left(\frac{n}{e}\right)^n$

All $\log_a(n) \in \Theta(\log n)$ no matter what the log base is

$3^n \notin \Theta(2^n)$

Analysis of non-recursive algorithms

1. Decide input size n
2. Identify main operation
3. See if main op depends only on n or not
4. Set up summation for basic op count.
5. Solve

$\sum_{u} 1 = u-l+1$

$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

$\sum_{i=0}^{n} a^i = \frac{(a^{n+1}-1)}{(a-1)}$

Analysis of recursive algorithms

1. Decide input n
2. Identify basic operation
3. Dependent on only n or not
4. Set up recurrence relation & initial conditions
5. Solve and estimate order of magnitude of soln.

Denom = 2 $\xrightarrow[\text{rule}]{\text{smoothness}}$ take n as 2^k

BRUTE FORCE

Selection Sort

ith iteration: i elements in final pos.
look for min of elements not in final pos $\longrightarrow$ swap min with A[i].

```
for i ← 0 to n-2
    min ← i
    for j ← i+1 to n-1
        if A[j] < A[min]
            min ← j
    swap A[min], A[i]
```

P.T.O

Bubble Sort.

For ith iteration, final i elements are in final pos
compare every two adjacent eles → swap.

```
for i ← 0 to n-2
    for j ← 0 to n-2-i
        if A[j] > A[j+1]
            swap A[j], A[j+1]
```

Sequential Search

String Matching

Pattern P, String S : if cur char not equal, shift to right by 1
(length = m) (length = n)

```
for i ← 0 to n-m
    j ← 0
    while S[i+j] == P[j]
        j ← j+1
    if j = m return 1
return -1
```

TSP:
(n-1)! permutations (keeping origin city fixed) } shortest Hamiltonian circuit in weighted connected graph

Knapsack:
2^n : no. of subsets for n elements. } most valuable subset
$\Omega(2^n)$

Assignment

n people, each assigned to 1 job each.
Jobs are unique
Cost of i to j : c[i, j]

Minimise total cost.
n! : total permutations of all n people into n jobs.