Time Complexity

Define hypothetical model machine

- Single vs multi processor
- Read / write speed to memory
- 32-bit vs 64-bit architecture
- ...

Running time / rate of growth of time

- Asymptotic notation: Big Oh, Omega, Theta
    - `f(n) = O(g(n))` means `c * g(n)` is an upper bound on `f(n)`.
    - `f(n) = Ω(g(n))` means `c * g(n)` is a lower bound on `f(n)`.
    - `f(n) = Θ(g(n))` means `c1 * g(n)` is an upper bound on `f(n)` and `c2 * g(n)` is a lower bound on `f(n)`.
- We analyse time complexity of a algorithm for very large input size.
    - Drop lower order terms
    - With Big Oh we discard multiplication of constants: `n^2` and `1000 * n^2` are treated identically.
    - Measure all fragments
    - Pick complexity of condition which is the worst case
    - `O(f(n)) + O(g(n))` => `O(max(f(n), g(n)))` => `n^3 + n^2 + n + 1 = O(n^3)`
    - `O(f(n)) * O(g(n))` => `O(f(n) * g(n))`

Space Complexity

- It is a measure of how efficient your code is in terms of the largest memory use by the program when it runs.
- It analysis happens almost in the same way time complexity analysis happens.

Examples

Growth rates

- `1` < `log(n)` < `n` < `n * log(n)` < `n^2` < `n^3` < `2^n` < `n!`
    - Constant functions
    - Logarithmic: binary search, arises in any process where things are repeatedly halved or doubled
    - Linear: traversing every item in an array
    - Super-linear: quick sort, merge sort
    - Quadratic: going thru all pairs of elements, insertion sort, selection sort
    - Cubic: enumerating all triples of items
    - Exponential: enumerating all subsets of n items
    - Factorial: generating all permutations or orderings of n items

Recursion Complexity Analysis

- Let `T(n)` be the recursive function.

- Bellow are examples of inner calls of `T(n)`, their complexity, and example algorithm that uses them:
    - `T(n) = T(n/2) + O(1)`, `O(log(n))`, binary search
    - `T(n) = T(n-1) + O(1)`, `O(n)` , sequential search
    - `T(n) = 2 * T(n/2) + O(1)`, `O(n)`, tree traversal
    - `T(n) = 2 * T(n/2) + O(n)`, `O(n * log(n))`, merge sort, quick sort
    - `T(n) = T(n-1) + O(n)`, `O(n^2)`, selection sort
- Geometric progression:

```
a(n) = a(1) * q^(n-1)
S(n) = a(1) * (q^n-1) / (q-1)

If it converges:
S(inf) = a(1) / (1-q)
```

Combinatorics

- All pairs: `O(n^2)`
- All triples: `O(n^3)`
- Number of all permutations: `n!`
- `n` over `k`: number of combinations for choosing `k` items from a set of `n`
    - `(n over k) = n! / (k! * (n-k)!)`
- Number of subsets from a set of `n`: `2^n`
    - Subset = any unique set of `k` elements from `n`, including the empty set).
    - For example: `set={1, 2, 3}`, `subsets={}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}` (ordering in a set doesn't matter).

Handy Formulas

- `1 + 2 + ... + n = n * (n + 1) / 2`
- `x + x/2 + x/4 + ... = 2x`