Analysis, Big O and Growth of Functions

Zach Neveu

May 27, 2019

1 Book Keeping

- Reading posted
- Lab 1 available

2 Analysis of Algorithms

Problem: a general description of input parameters and the properties that an optimal solution should have

Instance: a specific example of a problem with all parameters specified

- Example: Given a weighted graph, find the cheapest Hamiltonian Cycle (TSP)
- A "problem" can have many instances

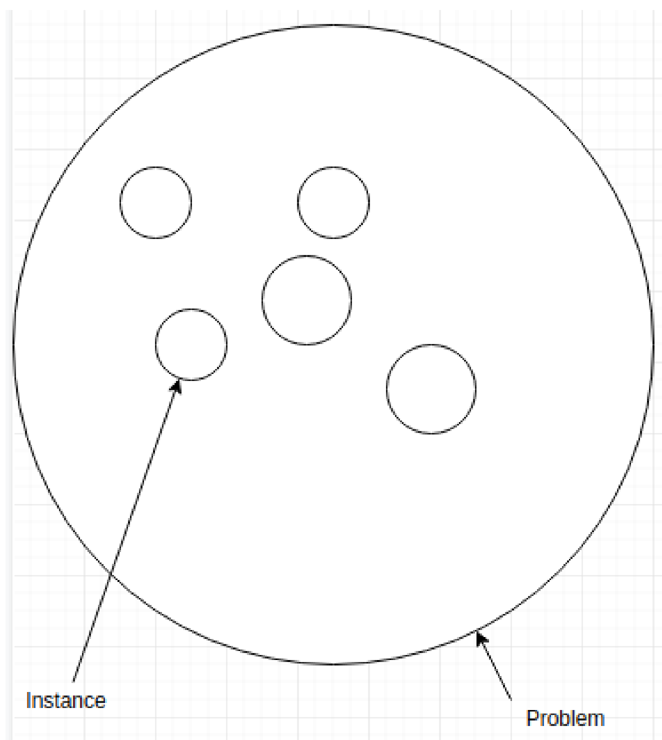

Figure 1: instance_problem

- An algorithm solves all instances of problem

- Many algorithms, what is most efficient?
- What is efficient?
  - Memory
  - Time
  - CPU cycles
  - Disk Space
  - I/O bandwidth
  - Power
- Efficiency usually defined as using smallest time
- Index runtimes by instance size
- "Instance Size" not always well defined - can have multiple params (edges, nodes)

3 Example: Insertion Sort

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
$A[i+1] \leftarrow key$	c_8	$n-1$

Figure 2: Cor p.24

- Best case: already sorted. $T(j) = 1, T(n) = an + b \rightarrow$ linear
- Worst case: reverse sorted: $T(j) = j, T(n) = \frac{n(n+1)}{2} \approx an^2 + bn + c \rightarrow$ quadratic

  Time Complexity Function: The largest amount of time for an algorithm needed to solve the problem for a given instance size.
- Even Time-Complexity function considered too complicated for daily use
- Asymptotic notation used instead

4 Asymptotic Notation

For a given function $g(n)$, $O(g(n)) = f(n)$ there exist positive constants k and n_0 such that $f(n) \leq Kg(n)$ for all $n \geq n_0$
Less formally: $O(g(n))$ is the set of functions that are asymptotically less than $g(n)$ for large n.

Example

I claim that $f(n) = an^2 + bn + c = O(n^2)$. If so, then there should exist positive constants k and n_0 such that

$$an^2 + bn + c \leq kn^2$$
$$a + b/n + \frac{c}{n^2} \leq k$$
$$k = a + 1$$
$$n_0\ is\ intersection$$

Summary

- For insertion sort, worst case runtime (time complexity function) is $an^2 + bn + c$ so the complexity is $O(n^2)$

- Also $O(n^3), O(n^4)$ etc.
- Worst case runtime is $O(n^2)$
- Worst case runtime **itself** is upper bound on run time
- $O(n^2)$ is then an upper bound on the general runtime as well!

  Polynomial-time Algorithm: an algorithm whose time complexity function is $O(p(n))$ for some polynomial $p(n)$

  Exponential-time Algorithm: an algorithm that is not polynomial time

EXPONENTIAL VERY BAD