Lecture 2: Asymptotic Complexity + Worst-case analysis

Motivating Q: What does it mean for an algorithm to be "efficient"?

Proposal: When implemented, it runs quickly on input instances

Proposal: Achieves qualitatively better worst-case performance, at an analytical level, than brute-force search

☆ Proposal: An algorithm is efficient if it has a polynomial running time.

⇒ Why care about "worst-case" analysis?

↳ Want to know how bad it could get
↳ Average / Expected case is important, but defining "random" or "expected" input is often very difficult.
↳ In practice, actual polynomial solutions often tend to be lower-order

e.g. $\lg(n)$, $n\lg(n)$, n^2, n^3

↳ Problems where we know of no current poly. solutions tend to be difficult in practice

↳ This defn. "really works"

↳ Allows us to express that there is no efficient algorithm for a specific problem.

Asymptotic Order of Growth

- running time of algo on inputs of size n grows at a rate proportional to $f(n)$
- Let $T(n)$ be a function - the worst-case running time of an algo. on an instance of size $n > 0$.
- Given function $f(n)$, we say:

  $\boxed{T(n) \text{ is } O(f(n))}$ if, for sufficiently large n, $T(n)$ is bounded above by a constant multiple of $f(n)$.

  This is often written as:

  $$\boxed{T(n) = O(f(n)) \quad \text{or} \quad T(n) \in O(f(n))}$$

abuse of notation, but most common.

Def: $T(n) \in O(f(n))$ if $\exists\, c > 0$ and $n_0 \geq 0$ such that $\forall\, n \geq n_0$, $T(n) \leq c \cdot f(n)$.

Here, $T(n)$ is asymptotically upper-bounded by $f(n)$.

E.g. Assume $T(n) = pn^2 + qn + r$ for constants p, q, r.

Then, any such $T(n)$ is $O(n^2)$ ... why?

Proof: $T(n) = pn^2 + qn + r \leq pn^2 + qn^2 + rn^2 = (p+q+r)n^2$

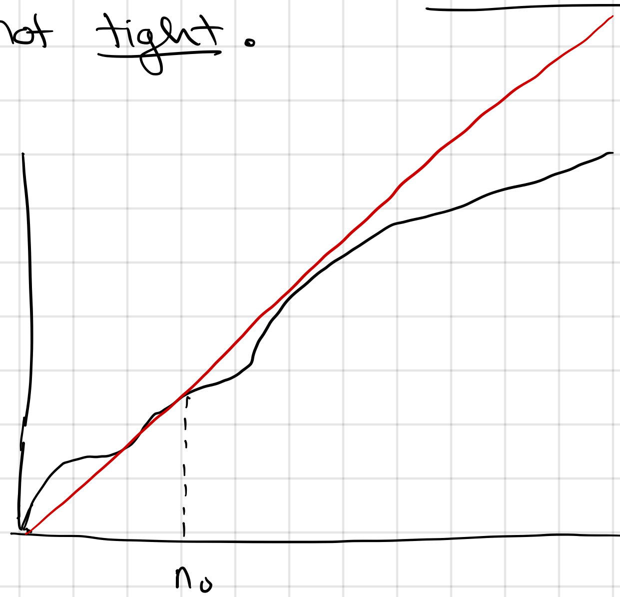
for all $n \geq 1$.

Hence, if we set $c = p+q+r$, we have that

$T(n) \leq c \cdot n^2 \quad \forall\, n_0 \geq 1$

<u>Note</u>: $T(n)$ is <u>also</u> $O(n^3)$, $O(n^4)$ etc. However, these bounds are <u>less useful</u> because they are not <u>tight</u>.

Visual Eg.



Assuming no "funny business" here

$T(n) \in O(f(n))$

Another def.

$$f(n) = O(g(n)) \iff \overline{\lim_{n\to\infty}} \frac{|f(n)|}{g(n)} < \infty$$

Notation: $\overline{\lim_{n\to\infty}}\, x_n$ is the limit superior

$O(\cdot)$ notation tells us about asymptotic <u>upper bounds</u>

Q: What about lower bounds?

- What if we want to show that an $O(\cdot)$ upper bound (e.g. $O(n^2)$) is the best possible?
- Want to show that for sufficiently large n, $T(n)$ is at least as large as some constant multiple of $f(n)$.
- We write this as $T(n) \in \Omega(f(n))$ or $T(n) = \Omega(f(n))$.

Def: $T(n)$ is $\Omega(f(n))$ if $\exists\ \epsilon > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0$, $T(n) \geq \epsilon \cdot f(n)$.

Here $T(n)$ is <u>asymptotically lower bounded</u> by f.

E.g. Let $T(n)$ as above be $pn^2 + qn + r$ for const. p, q, r. Then $T(n) \in \Omega(n^2)$ ... why?

Proof: $T(n) = pn^2 + qn + r \geq pn^2$ for $n \geq 1$.

Thus, we can set $\epsilon = p$ $(p > 0)$ and we have $T(n) \geq \epsilon \cdot n^2$ for all $n \geq 1$.

Note: As with $O(\cdot)$, a complementary issue comes up with $\Omega(\cdot)$. That is,

$n^2 \in \Omega(n^2)$ and $n^2 \in \Omega(n)$ and $n^2 \in \Omega(\lg n)$

Asymptotically Tight Bounds

What if $T(n) \in O(f(n))$ <u>and</u> $T(n) \in \Omega(f(n))$?

Then, scalings of $f(n)$ effectively "sandwich" our function $T(n)$.

Def: If $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$ then $T(n) \in \Theta(f(n))$.

Here $\Theta(f(n))$ is an <u>asymptotically tight bound</u> for $T(n)$.

Eg. Returning to $T(n) = pn^2 + qn + r$, we already showed that $T(n) \in O(n^2)$ and $T(n) \in \Omega(n^2)$

so $T(n) \in \Theta(n^2)$

Limit Bounds

$$f(n) \in O(g(n)) \iff \overline{\lim_{n\to\infty}} \frac{|f(n)|}{g(n)} < \infty$$

$$f(n) \in \Omega(g(n)) \iff \underline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} > 0$$

$$f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$$

alternatively:

$$f(n) \in \Theta(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0$$

Properties of Bounds: (2.2 - 2.3)

$O(\cdot)$ and $\Omega(\cdot)$ are transitive

- If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$ then $f(n) \in O(h(n))$
- If $f(n) \in \Omega(g(n))$ and $g(n) \in \Omega(h(n))$ then $f(n) \in \Omega(h(n))$
- If $f(n) \in \Theta(g(n))$ and $g(n) \in \Theta(h(n))$ then $f(n) \in \Theta(h(n))$

Sums (2.4–2.5)

2.4 • $f(n) \in O(h(n))$ and $g(n) \in O(h(n)) \Rightarrow [f(n) + g(n)] \in O(h(n))$

2.5 • Let K be some fixed natural number (independent of n).

If $f_i(n) \in O(h(n))$ for $1 \le i \le K$ then

$$\left(\sum_{i=1}^{K} f_i(n) \right) \in O(h(n))$$

Proof: By defn, let us have

$$f_i(n) < c_i \cdot h(n) \quad \text{for all } n \geq n_0^i$$

then

$$\sum_{i=1}^{K} f_i(n) < c_1 \cdot h(n) + c_2 \cdot h(n) + \ldots + C_K h(n) \quad \forall n \geq \max_i n_0^i$$

$$\sum_{i=1}^{K} f_i(n) < \left(\sum_{i=1}^{K} c_i \right) h(n)$$

So $\sum_{i=1}^{K} f_i(n)$ is $O(h(n))$ for $C = \left(\sum_{i=1}^{K} c_i \right)$ and $n_0 = \max_i n_0^i$

2.6 If $g(n) \in O(f(n))$ then $f(n)+g(n) \in \Theta(f(n))$

* Like 2.5, this extends to the sum of a fixed number of functions.

Some Common Asymptotic Bounds:

(1) If $f(n) = a_d n^d + a_{d-1} n^{d-1} + \dots + a_0$, then $f(n)$ is $O(n^d)$ (tight if $a_d > 0$)

(2) For every $b > 1$ and $x > 0$, $\log_b n = O(n^x)$

↳ Every logarithmic function is upperbounded by a polynomial (all polynomials grow more quickly than logarithms)

↳ $\log_a n \in \Theta(\log_b n)$ for $a, b > 1$ so, the base of the log doesn't matter (and we just write $\log n$). Why? because $\log_b(x) = \dfrac{\log_a(x)}{\log_a(b)}$

(3) For every $r > 1$ and $d > 0$, $n^d \in O(r^n)$

↳ every exponential grows faster than any polynomial.

Examples of some algorithms & their runtimes

E.g.

```
FindMax (a):
  max = a[0]
  for i=1 to n:
      if a[i] > max:
          max = a[i]

  return max
```

Q: running time?

E.g. Merging 2 sorted lists

max # of times is size of input

- Compare heads of the lists
- Select & remove largest element
- append to output

Q: running time?

E.g. $O(n \lg n)$?

- Many comparison based sorting algos
  - heapsort, mergesort

E.g. $O(n^2)$
- naive closest pair on n points
- insertion sort
- edit distance / optimal string alignment

E.g. $O(n^3)$
- naive matrix mult
- Given $S_1, \ldots, S_n$; each a subset of $\{1, 2, \ldots, n\}$ determine if any pair of sets is disjoint.

E.g. $O(n^k)$ (best known algos)
- Does a graph have an independant set of size k?

E.g. $O(2^n)$
- Find a max size independent set in graph G

$O(n!)$
- naive TSP (can be $O(n^2 \cdot 2^n)$ with Dyn prog)

E.g. Sublinear ($O(\lg n)$)
- find if value v exists in a sorted array.