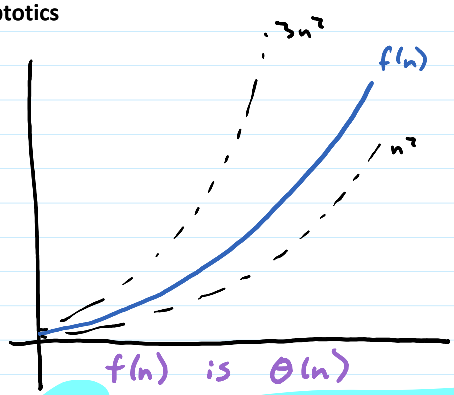
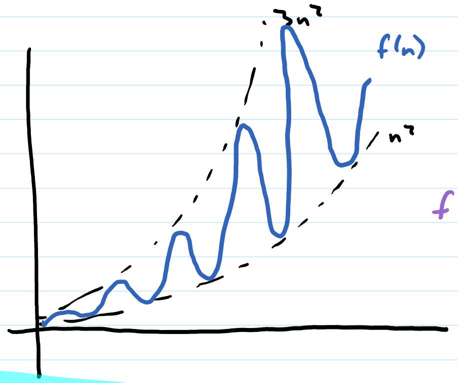
Asymptotics



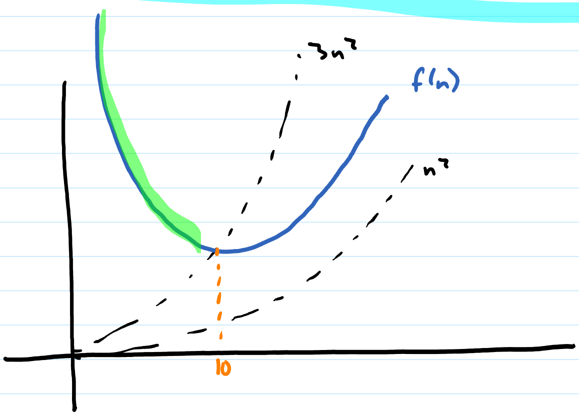
$f(n)$ is $\Theta(n)$

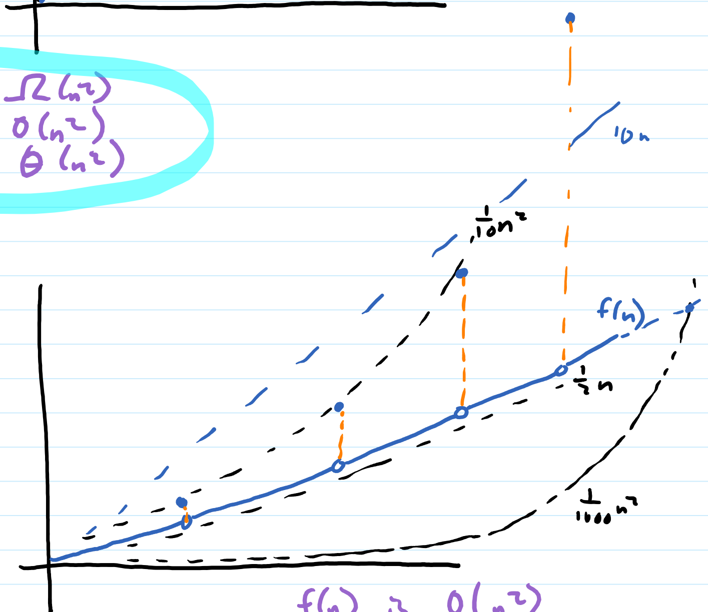


$\lim_{n\to\infty} \frac{f(n)}{n^2}$ does not exist

$f(n)$ is $\Theta(n^2)$

$\forall n \geq 0$, $f(n) \geq 1 \cdot n^2$ $\quad f(n)$ is $\Omega(n^2)$
$\forall n \geq 0$, $f(n) \leq 3 \cdot n^2$ $\quad f(n)$ is $O(n^2)$
so $f(n)$ is $\Theta(n^2)$





$f(n)$ is $O(n^2)$
$f(n)$ is $\Omega(n)$

$\forall n \geq 10$, $f(n) \geq 1 \cdot n^2$ $\quad f(n)$ is $\Omega(n^2)$
$\forall n \geq 10$, $f(n) \leq 3 \cdot n^2$ $\quad f(n)$ is $O(n^2)$
so $f(n)$ is $\Theta(n^2)$

Limit test:

if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ then $g(n)$ is $O(f(n))$

$= c$ for some $c > 0$ then $g(n)$ is $\Theta(f(n))$

$= 0$ then $f(n)$ is $O(g(n))$

Alg A runs in time $O(n)$ means worst case time for A is $O(n)$

Alg A runs in time $\Theta(n)$ means worst case time time for A is $\Theta(n)$
best case time time for A is $\Theta(n)$

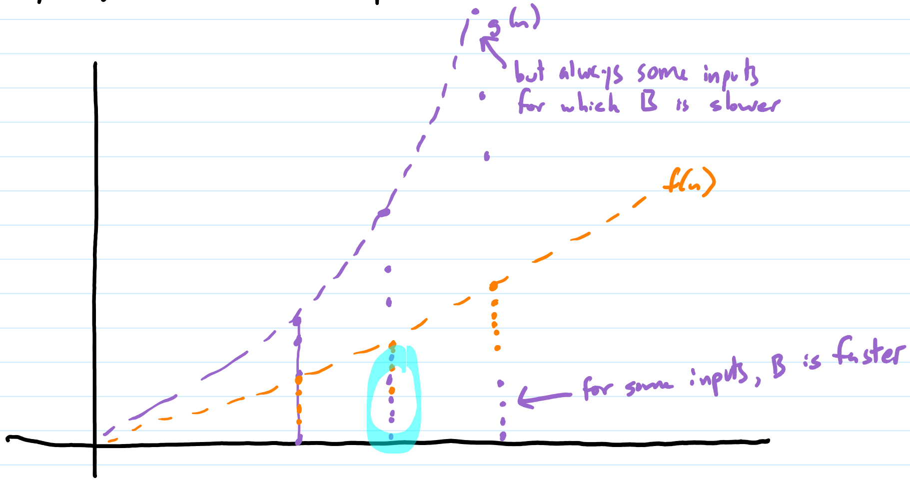
Remember: running time of alg. is a fxn of the input, not the size of the input
worst case running time is a fxn of size of input

Suppose Algs A, B both solve problem P and that worst case for A is $\Theta(f(n))$
and worst case for B is $\Theta(g(n))$

and $f(n)$ is $O(g(n))$ but not $\Theta(g(n))$

example: n^2 n^3

then for sufficiently large n, there are inputs of size n s.t. A is faster than B



Asymptotic Running Time

Selection sort is $O(n^3)$

$T(n)$ is $O(f(n))$ means there are constants $n_0 \ge 0$, $c > 0$ s.t. for all $n \ge n_0$, $T(n) \le c \cdot f(n)$

informal - for large enough n, $T(n) \le f(n)$ (ignoring multiplicative constants)

selection sort does $\frac{n(n-1)}{2}$ comparisons

$\forall n \ge 10$, $\frac{n(n-1)}{2} \le 1 \cdot n^3$

$\Omega(f(n))$ there are constants $n_0 \ge 0$, $c > 0$ s.t. for all $n \ge n_0$, $T(n) \ge c \cdot f(n)$

$\Theta(f(n))$ there are constants $n_0 \ge 0$, $c_1, c_2 > 0$ s.t. for all $n \ge n_0$, $c_1 \cdot f(n) \le T(n) \le c_2 \cdot f(n)$

transitive:

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

f is $O(h)$ and g is $O(h)$ then $f+g$ is $O(h)$

f is $O(g)$ then $f+g$ is $\Theta(g)$

If f is a polynomials of degree d (with positive coefficients on n^d term) then f is $\Theta(n^d)$

$n^4 + 1000n^3 - 764n^2 + 1961$ is $\Theta(n^4)$

For every $\varepsilon, d \ge 0$ n^d is $O(n^{d+\varepsilon})$ n^3 is $O(n^{3.01})$

For every $b > 1$, $d > 0$, $\log_b n$ is $O(n^d)$ $\log_{100} n$ is $O(n^2)$ and $O(\sqrt[100]{n})$

For every $d > 0$, $r > 1$, n^d is $O(r^n)$ n^{100} is $O(1.001^n)$

For $\varepsilon \ge 0$, $r \ge 0$ r^n is $O(r+\varepsilon)^n$ 2^n is $O(2.01^n)$

```
for i=1 to n
    do_something()
```

suppose do_something takes $O(n)$ time

then total time for loop is $O(n^2)$

could the total time also be $O(n)$? SURE!

total time $= \sum_{i=1}^{n}$ work done by that iteration

if work done $\le c \cdot n$ then

$\sum$ work done $\le \sum c \cdot n$

$= \underline{c \cdot n^2}$ $O(n^2)$

if work done usually 1 but occasionally n then work done $\le c \cdot n$

but can have $\sum$ work done $\approx \underline{c \cdot n}$ $O(n)$

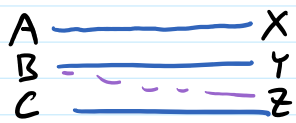
in these cases, go back to

Stable Matching

Problem (informal): Given n machinists and n welders, find a good way to match them.
(good: whatever this means)

Machinist	Preferences
A	X, Y, Z
B	X, Z, Y
C	Z, X, Y

Welder	Preferences
X	A, B, C
Y	A, C, B
Z	A, B, C



(B, Z) B would rather switch Y → Z (who they are currently matched with)
Z would rather switch C → B
instability

Matching: set S of ordered pairs M×W s.t. for each m∈M there is at most 1 w s.t. (m, w) ∈ S and for each w∈W there is at most 1 m s.t. (m, w) ∈ S

Perfect matching: matching s.t. for each m there is exactly 1 w s.t. (m, w) ∈ S and for each w there is exactly 1 m s.t. (m, w) ∈ S

Instability with respect to a perfect matching S is a pair (m, w') ∉ S s.t. m prefers w' to whichever w s.t. (m, w) ∈ S
w' prefers m to whichever m' s.t. (m', w') ∈ S
(instability = unmatched pair with both in pair preferring the other in the pair to who they are currently matched with)

Stable Matching: perfect matching with no instabilities

Gayle-Shapely

```
FreeM <- M                  -> unmatched machinists
FreeW <- W                  -> unmatched welders
Invitations <- {}           -> which machinists have asked which welders
Tentative <- {}             -> partial matching

While there is an m in FreeM s.t. there is a w s.t. (m,w) not in Invitations
    choose such an m
    let w be m's highest ranked s.t. (m,w) not in Invitations    so each m sends invitations in order of their preferences

    add (m,w) to Invitations    so we can determine later

    if w in FreeW then
        remove w from FreeW      } accept invitation
        remove m from FreeM      }
        add (m,w) to Tentative   }
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
            remove m from FreeM           } break current tentative match
            add m' to FreeM               }
            remove (m', w) from Tentative }
            add(m, w) to Tentative        }
return Tentative
```

Machinist	Preferences
A	X, Y, Z
B	X, Z, Y
C	Z, X, Y

Welder	Preferences
X	A, B, C
Y	A, C, B
Z	A, B, C

Invitations	FreeM	FreeW	Tentative
	A, B, C	X, Y, Z	
(A, X)	B, C	Y, Z	(A, X)
(B, X)			
(C, Z)	B	Y	~~(C, Z)~~
(B, Z)	C		(B, Z)
(C, X)			
(C, Y)			(C, Y)

A, X got 1st choice, so (A, –) or (–, X) not an instability

is (B, Y) an instability? no – B prefers Z

(C, Z) an instability? no – Z prefers B

no instabilities, so stable!

Does this always terminate?

Does this always return a stable matching?

What is the running time?