10-607 Computational Foundations for Machine Learning

Computational Complexity

Instructor: Pat Virtue

Plan

Computational Complexity

- Counting operations
- Big-O
- Complexity classes
- Proving a Big-O relationship holds
- Proving a Big-O relationship does *not* hold

How many statements are executed?

```
int search(int x, int[] A, int n)
{
  for (int i = 0; i < n; i++)
  {
    if (A[i] == x) {
      return i;
    }
  }
  return -1;
}
```

If x is not in A…
how times are these statements executed?

1	`i = 0`
n+1	`i < n`
n	`if (A[i] == x)`
n	`i++`
1	`return -1`

How many operations are executed?

```
int search(int x, int[] A, int n)
{
  for (int i = 0; i < n; i++)
  {
    if (A[i] == x) {
      return i;
    }
  }
  return -1;
}
```

If x is not in A…
how times **operations** are executed?

1	`i = 0`
n+1	`i < n`
3n	`if (A[i] == x)`
2n	`i++` i = i + 1
1	`return -1`

Slide credit: CMU 15-122

How many **operations** are executed?

How many program operations are required to compute:

- L2 norm of vector
- Vector dot product
- Frobenius norm of matrix
- Matrix-vector multiplication
- Matrix-matrix multiplication

```
def norm(a):
    ss=0
    for i in range(len(a)):
        ss = ss + a[i]*a[i]
    norm = np.sqrt(ss)
    return norm
```

Operations:

- Arithmetic operations (e.g. + or **)
- Logical operations (e.g., and)
- Comparison operations (e.g., <=)
- Structure accessing operations (e.g. array indexing like A[i])
- Simple assignment such as copying a value into a variable
- Calls to library functions that don't depend on size of input (e.g., print)
- Control Statements (e.g. if X>5)

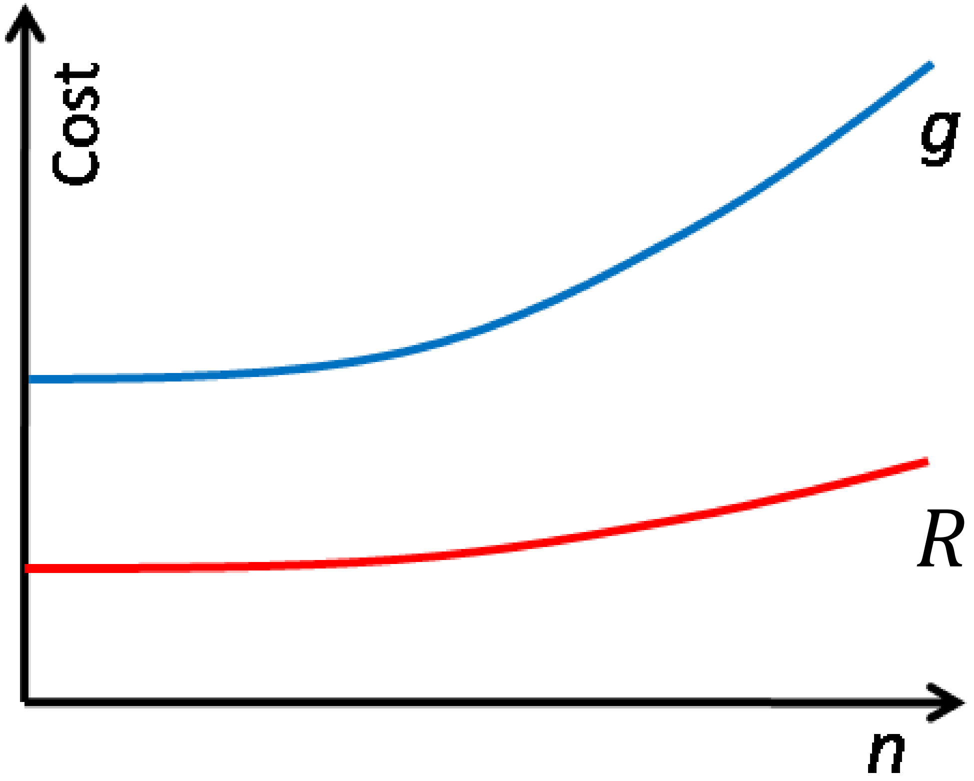
Be careful with function calls that scale with the size of the input

Exercise

Counting operations handout

Comparing functions of n

Which is better?



R is better than g

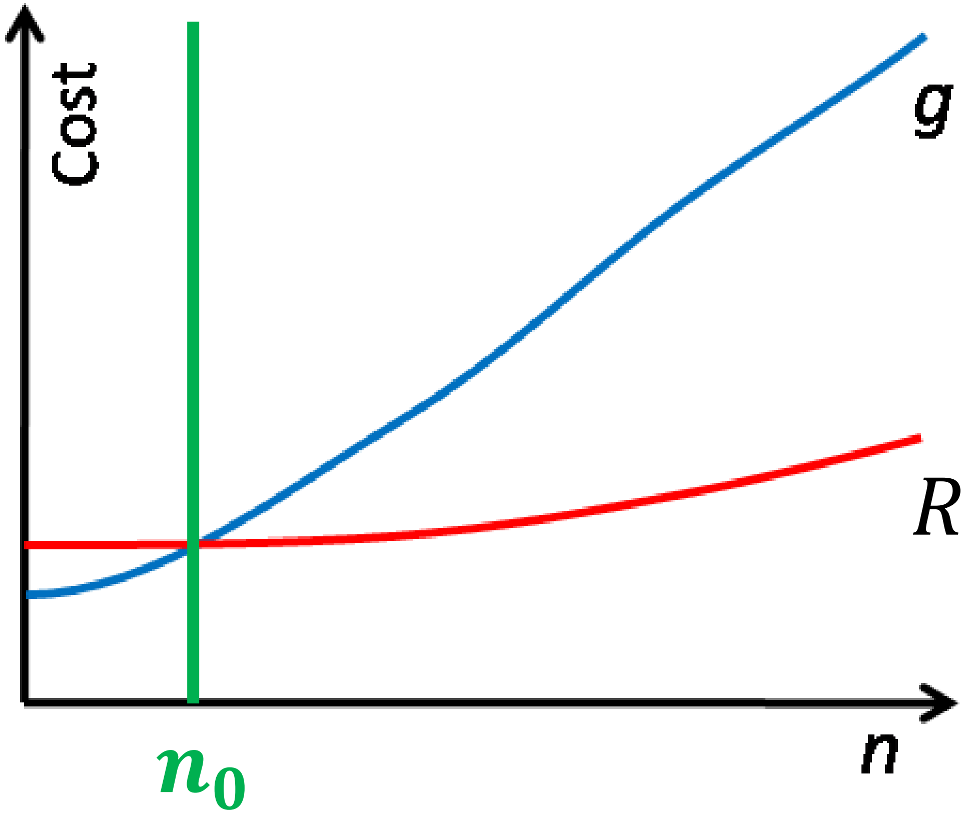
if

$$R(n) \leq g(n)$$

for all n

Comparing functions of n

Which is better?



R is better than g

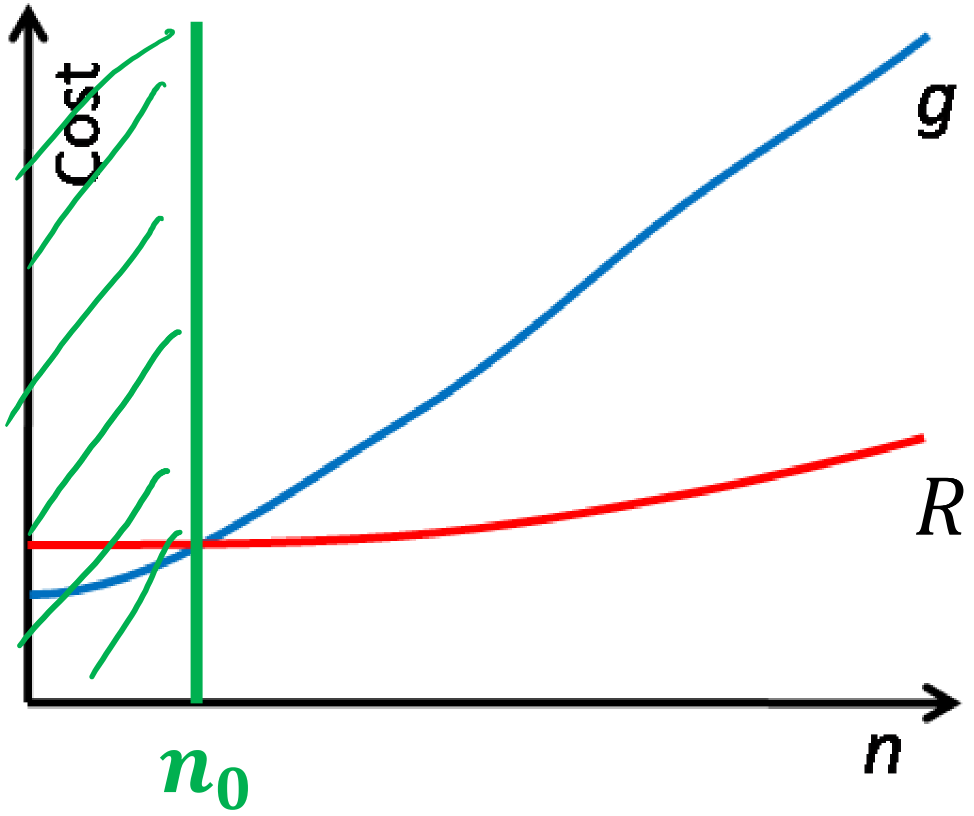
if

$$R(n) \leq g(n)$$

for all n

Comparing functions of n

Which is better?



R is better than g

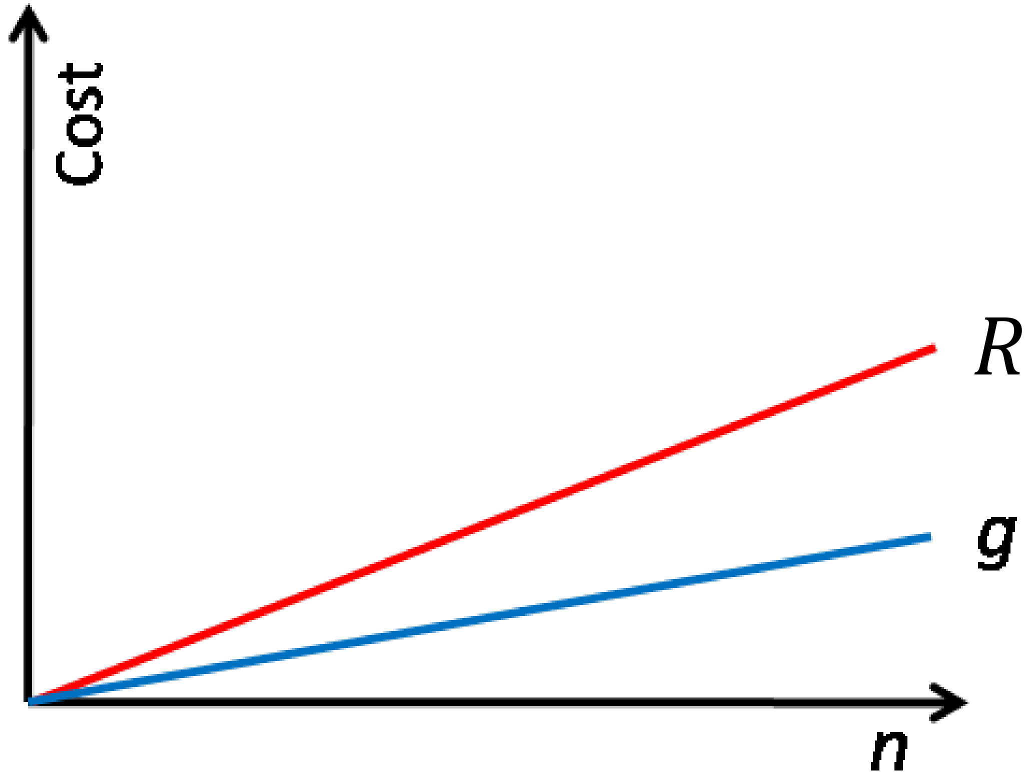
if **there exists n_0, such that**

$$R(n) \leq g(n)$$

for all $n \geq n_0$

Comparing functions of n

Which is better?



R is better than g

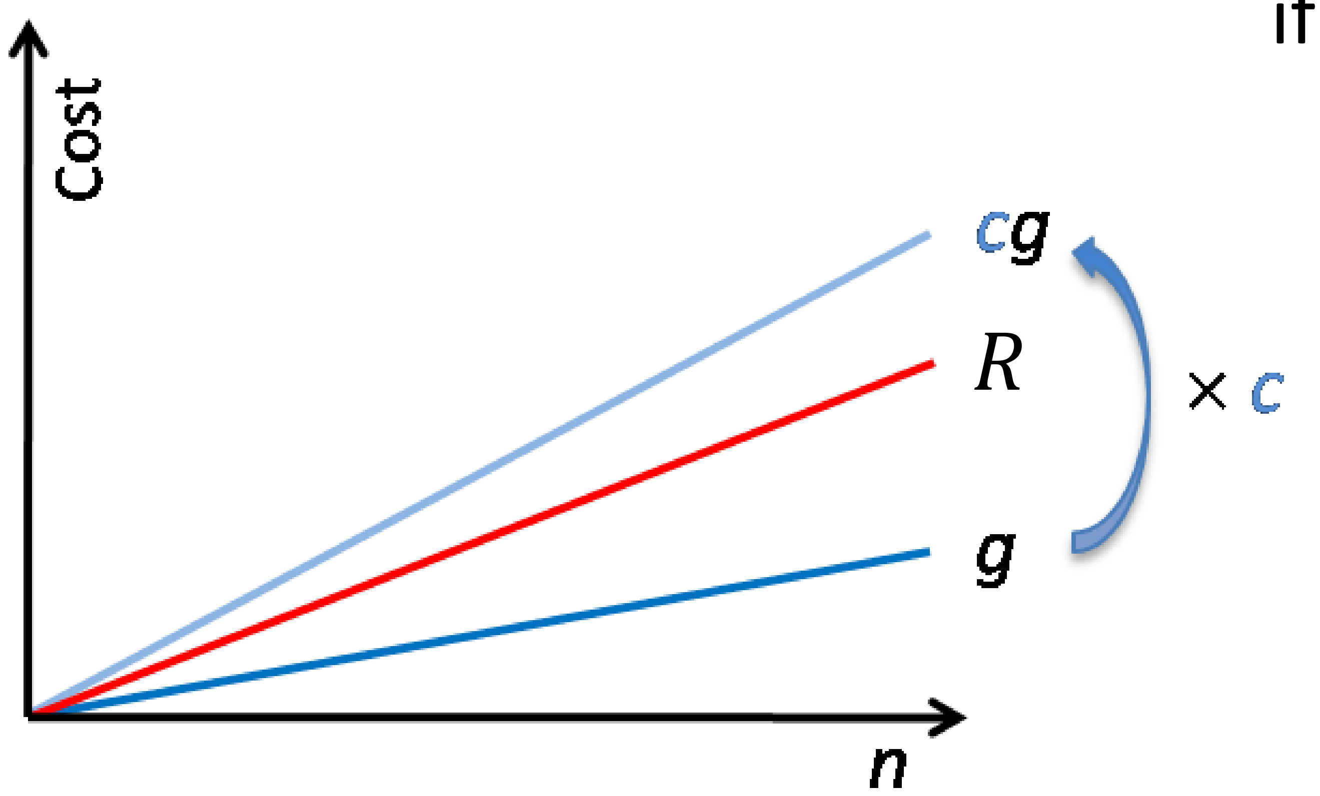
if there exists n_0, such that

$$R(n) \leq g(n)$$

for all $n \geq n_0$

Comparing functions of n

Which is better?



R is better than g

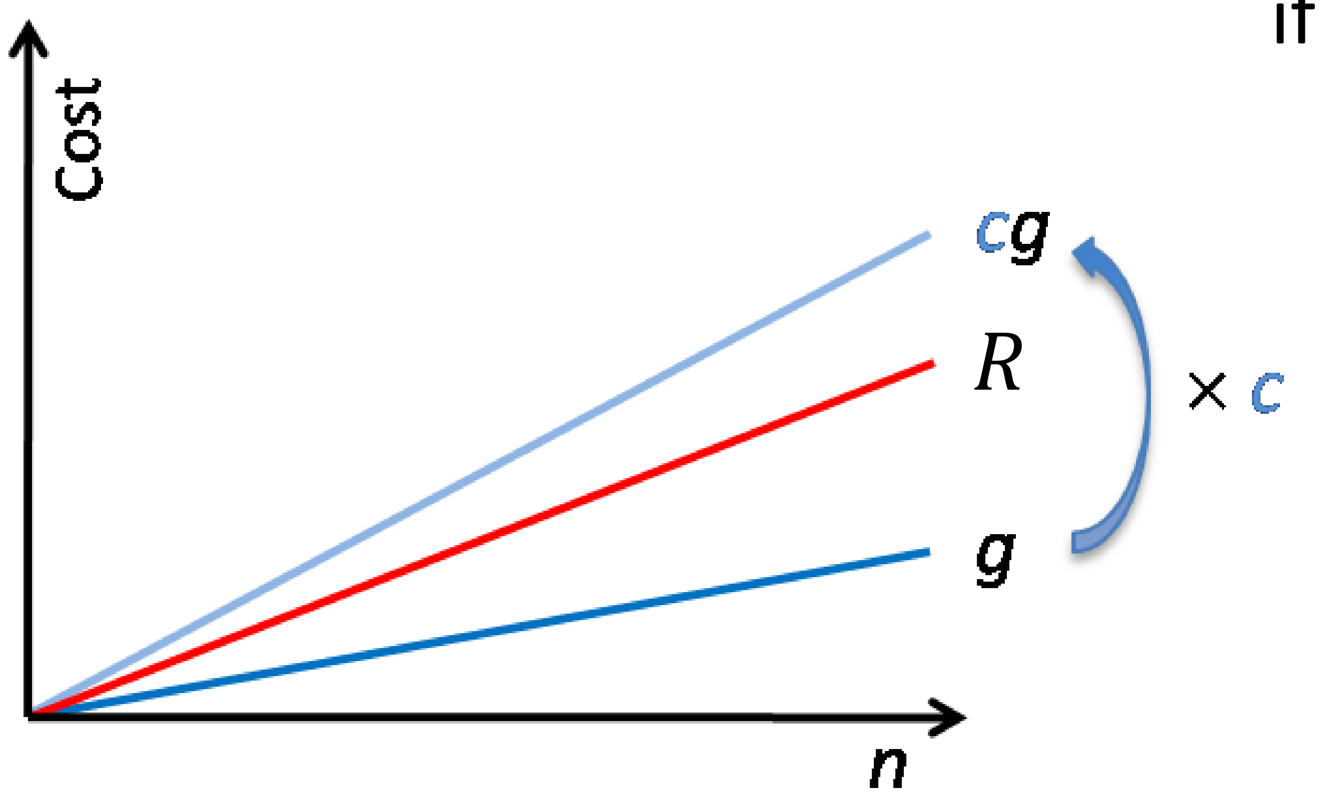
if there exists n_0, such that

$$R(n) \leq g(n)$$

for all $n \geq n_0$

Comparing functions of n

Which is better?



R is better than g

if there exists n_0 **and c**, s.t.

$$R(n) \leq \boldsymbol{c}g(n)$$

for all $n \geq n_0$

A set of better functions

R is better than g

if there exists n_0 and c, s.t.

$$R(n) \leq cg(n)$$

for all $n \geq n_0$

A set of better functions

The set of all functions R where

there exists n_0 and c, s.t.

$$R(n) \leq cg(n)$$

for all $n \geq n_0$

Definition of Big O of g

A set of better functions

Definition of Big O of g

The set of all functions R where

there exists n_0 and c, s.t.

$$R(n) \leq cg(n)$$

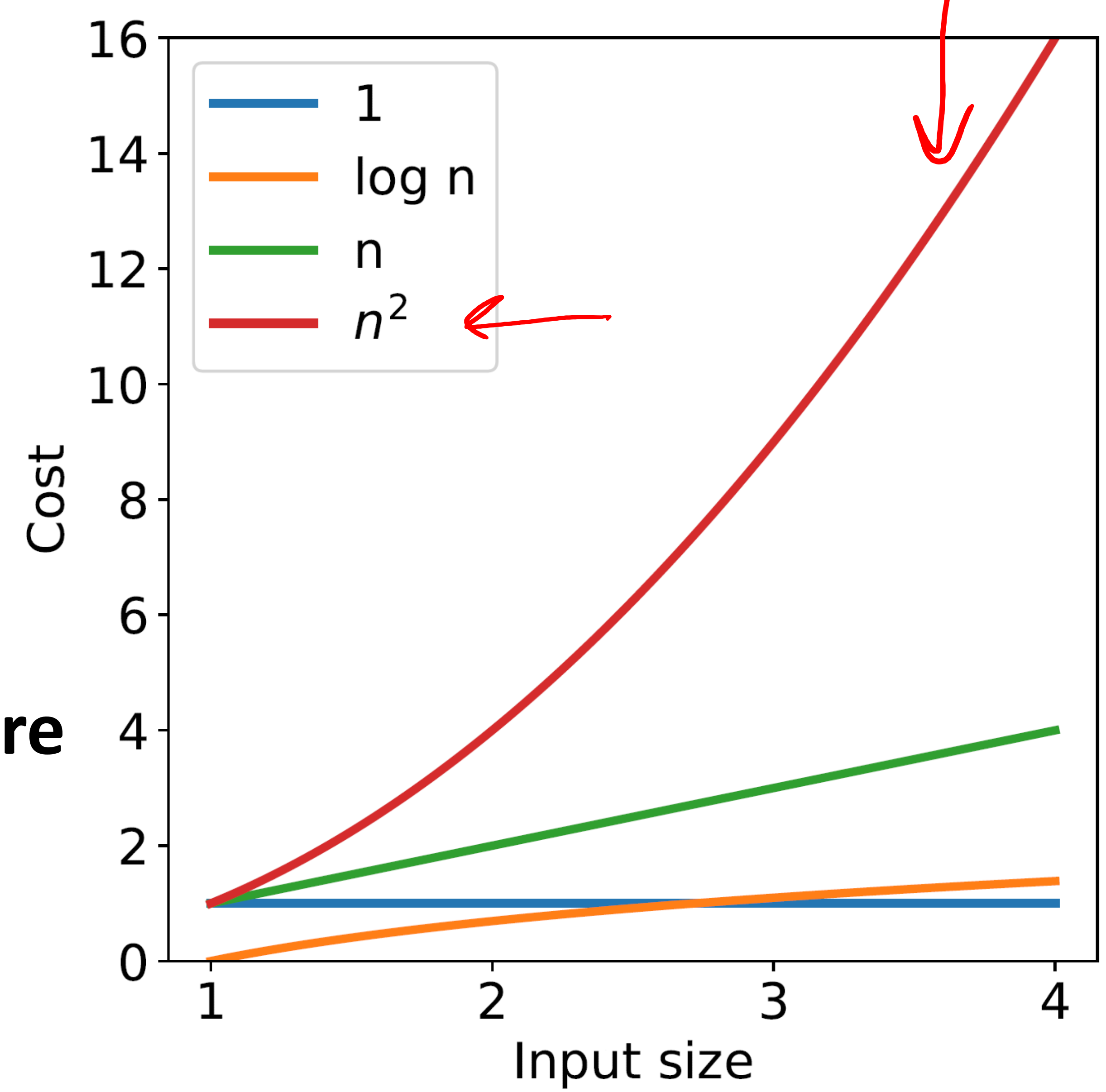
for all $n \geq n_0$

e.g. The set of all functions R where

there exists n_0 and c, s.t.

$$R(n) \leq c\mathbf{n^2}$$

for all $n \geq n_0$



A set of better functions

Definition of Big O of g

The set of all functions R where

there exists n_0 and c, s.t.

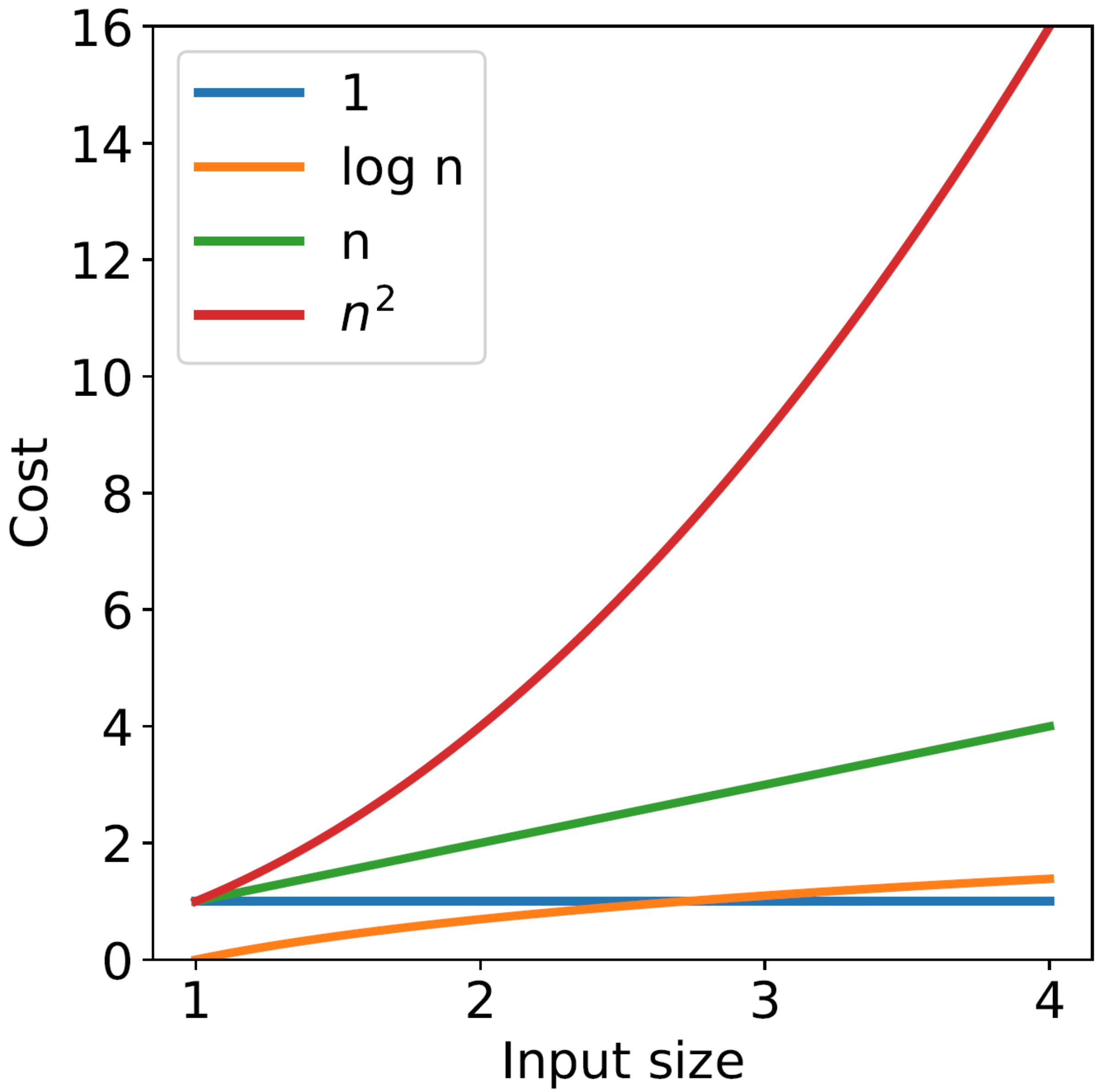
$$R(n) \leq cg(n)$$

for all $n \geq n_0$

e.g. $O(n^2)$ is a set that includes:

$R_1(n) = 1$ $\quad$ $R_3(n) = n$

$R_2(n) = \log n$ $\quad$ $R_4(n) = n^2$



Slide credit: CMU 15-122

A set of better functions

Definition of Big O of g

The set of all functions R where

there exists n_0 and c, s.t.

$$R(n) \leq cg(n)$$

for all $n \geq n_0$

e.g. $O(n^2)$ is a set that includes:

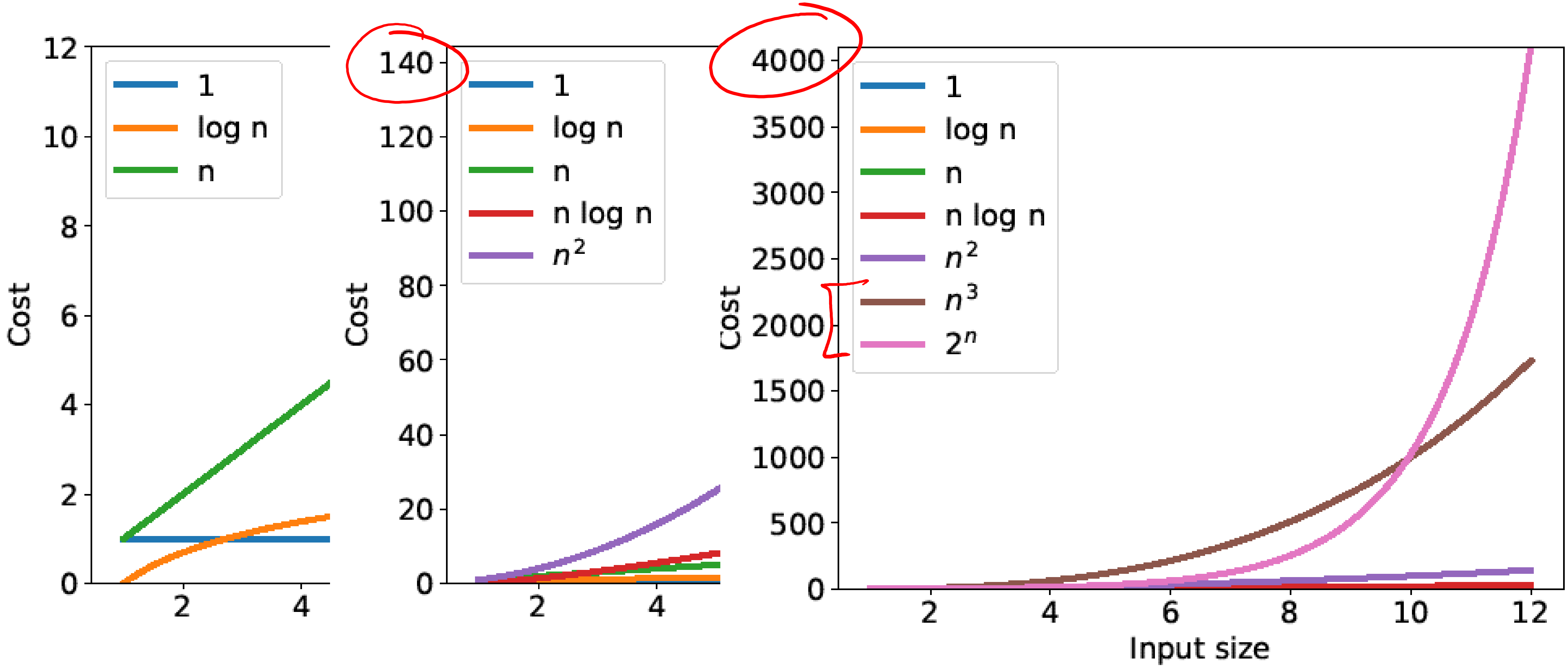
$R_1(n) = 1$ $\quad$ $R_3(n) = n$

$R_2(n) = \log n$ $\quad$ $R_4(n) = n^2$

$R_1 \in O(n^2)$
$R_2 \in O(n^2)$
$R_3 \in O(n^2)$
$R_4 \in O(n^2)$

Slide credit: CMU 15-122

Complexity Classes



Complexity Classes

Complexity class	**Conventional name**
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n \log n)$	*"n log n"*
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(2^n)$	Exponential

$$O(1) \subseteq O(\log n) \subseteq O(n) \subseteq O(n \log n) \subseteq O\left(n^2\right) \subseteq O\left(n^3\right) \subseteq O(2^n)$$

Slide credit: CMU 15-122

Complexity Classes

Can determine whether or not a problem can be solved at all!

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

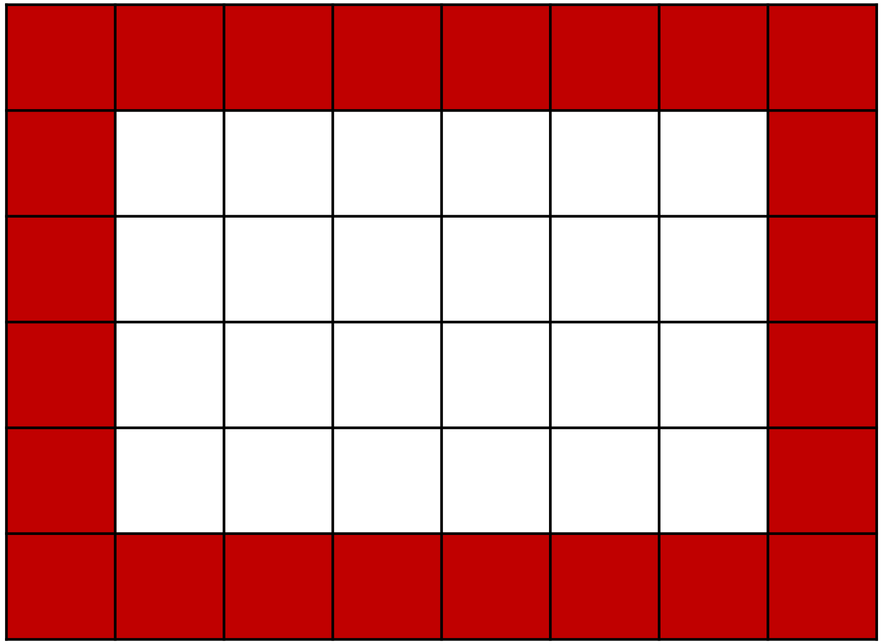
Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

Slide credit: Algorithm Design, Tardos, Kleinberg (table), UC Berkeley CS 61B, Hug (slide)

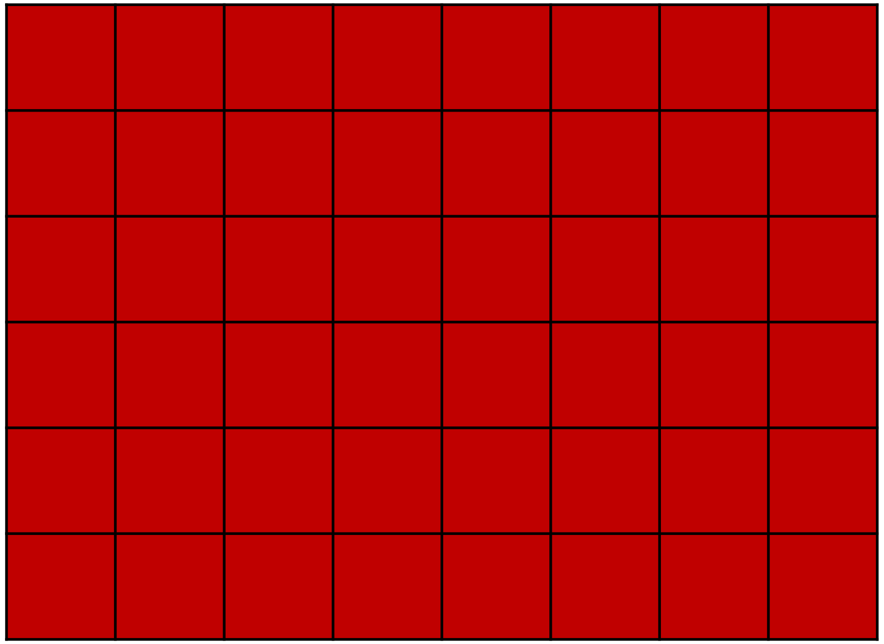
Input

Nothing special about variable n, we could have input variables w and h

$O(w+h)$



$O(wh)$



Plan

Computational Complexity

- ✓ Counting operations
- ✓ Big-O
- ✓ Complexity classes
- ▪ Proving a Big-O relationship holds
- ▪ Proving a Big-O relationship does *not* hold

Definition of Big O

Definition of $R(n) \in O\big(f(n)\big)$

Let $R(n)$ be a function, be the running time of some program as a function of the input size n. We assume that:

1. n is an integer ≥ 0
2. $R(n) \geq 0$ for all n

- $f(n)$ is a function defined on n. We say that "$R(n)$ is in $O\big(f(n)\big)$" if there exists a constant $c > 0$ and an integer n_0 such that, for all integers $n \geq n_0$, we have $R(n) \leq cf(n)$.

Witnesses

n_0 and c are called witnesses that $R(n)$ is in $O\big(f(n)\big)$. Finding such witnesses is a form of proof of $R(n)$ being in $O\big(f(n)\big)$.

Proving a Big-O Relationship Holds

Template to prove $R(n) \in O(f(n))$

1. State the witnesses n_0 and c as specific constants, e.g., $n_0 = 32$ and $c = 5$.
2. By appropriate algebraic manipulation, show that if $n \geq n_0$ then $R(n) \leq cf(n)$.

Proving a Big-O Relationship Holds

Example: Prove $(n+1)^2 \in O(n^2)$

Suppose $R(0) = 1$, $R(1) = 4$, $R(2) = 9$, and in general $R(n) = (n+1)^2$.

$(n+1)^2 = n^2 + 2n + 1$

$\leq n^2 + 2\underline{n^2} + 1$

$\leq n^2 + 2n^2 + n^2$

$\leq 4n^2$

$\leq cn^2$

$n_0 = 1$

$n \leq n^2$

$1 \leq n^2$

$c = 4$

$n_0 = 1$

Proving a Big-O Relationship Holds

Example: Prove $(n+1)^2 \in O(n^2)$

Suppose $R(0) = 1$, $R(1) = 4$, $R(2) = 9$, and in general $R(n) = (n+1)^2$.

We can say that $R(n) \in O(n^2)$, by choosing witnesses $n_0 = 1$ and $c = 4$:

- Expand $(n+1)^2 = n^2 + 2n + 1$
- if $n \geq 1$, we know that $n \leq n^2$ and $1 \leq n^2$
- Thus $n^2 + 2^n + 1 \leq n^2 + 2n^2 + n^2 = 4n^2$.

Proving a Big-O Relationship Holds

Example: Prove $(n+1)^2 \in O(n^2)$

Suppose $R(0) = 1$, $R(1) = 4$, $R(2) = 9$, and in general $R(n) = (n+1)^2$.

We can say that $R(n) \in O(n^2)$, by choosing witnesses $n_0 = 1$ and $c = 4$:

- Expand $(n+1)^2 = n^2 + 2n + 1$
- if $n \geq 1$, we know that $n \leq n^2$ and $1 \leq n^2$
- Thus $n^2 + 2^n + 1 \leq n^2 + 2n^2 + n^2 = 4n^2$.

Choosing witnesses

We could have also picked $n_0 = 3$ and $c = 2$.

However, we *can't* pick $n_0 = 0$ with any c (why?). But that doesn't matter, because we only need to find one pair of witnesses n_0 and c.

Proving a Big-O Relationship Holds

Example: Prove $(n+1)^2 \in O(n^2)$

But $(n+1)^2$ is bigger than n^2!!!

It may seem odd that $(n+1)^2 \in O(n^2)$ even though $(n+1)^2 > n^2$.

But being in $O(f(n))$ does not mean "less than" $f(n)$.

In fact, $(n+1)^2$ is also in big-O of any fraction of n^2, for example:

$(n+1)^2 \in O(n^2/100)$ with witnesses $n_0 = 1$ and $c = 400$

Proving a Big-O Relationship Holds

Quick tips and important points

Constant factors don't matter

For any positive constant d and any function that is $O(f(n))$ is also $O(df(n))$. (Choose $n_0 = 0$ and $c = 1/d$.)

Low-order terms don't matter

Consider a polynomial $R(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_2 n^2 + a_1 n + a_0$ where the leading coefficient, a_k, is positive. We can throw away all terms except the term with the highest exponent, k, and we can ignore a_k (a constant), replacing it by 1. $R(n) \in O(n^k)$. (To prove, choose $n_0 = 1$, and $c = \sum_{i \in \{1,\ldots,k\} \mid a^i > 0} a^i$.)

Poll 1

Which of the following functions are in $O(n^2)$?

Select all that apply.

1. 1
2. n
3. $n \log(n)$ $\leq n \cdot n$
4. n^2
5. $4n^2$
6. $4n^2 + \cancel{n \log(n)}$
7. $4n^2 + \cancel{n \log(n)} + \cancel{n}$
8. n^3
9. $n^3 + n^2$

Proving a Big-O Relationship Does Not Hold

Template to disprove $R(n) \in O\big(f(n)\big)$

1. Assume that witnesses n_0 and c exist
2. Derive a contradiction

Proving a Big-O Relationship Does Not Hold

Example: Prove that n^2 is not in $O(n)$

R(n) f(n)

- Assume $n^2 \in O(n)$ cf(n)
- Then there exist n_0 and c such that $n^2 \leq cn$ for all $n \geq n_0$.

Def $n \leq c$

$n_a > c$ $\longleftrightarrow$ $n_a \leq c$ Contradiction

$n_a > n_0$

Proving a Big-O Relationship Does Not Hold

Example: Prove that n^2 is not in $O(n)$

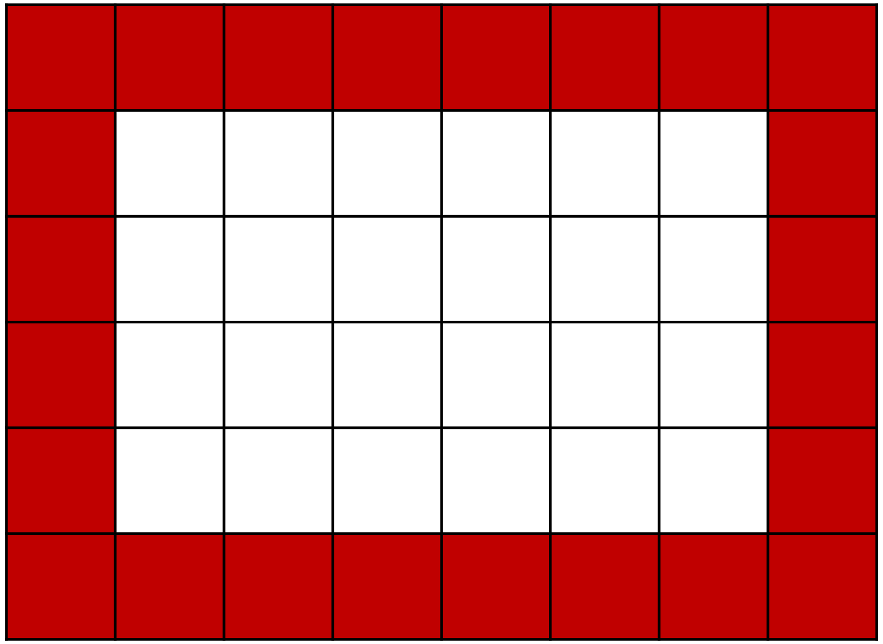
- Assume $n^2 \in O(n)$
- Then there exist n_0 and c such that $n^2 \leq cn$ for all $n \geq n_0$.
- Let n_a be a value $n_a > \max(n_0, c) + 1$
- Then $(\mathrm{n_a})^2 \leq cn_a$
- Dividing both sides by n_a, we have $n_a \leq c$. Contradiction!
- Therefore $n^2 \notin O(n)$

Input

Nothing special about variable n, we could have input variables w and h.

However, be careful to pay attention to which inputs we care about analyzing.

$O(w + h)$



Input

Nothing special about variable n, we could have input variables w and h.

However, be careful to pay attention to which inputs we care about analyzing.

Nearest neighbor example

N: datapoints

$\vec{x}^{(i)} \in \mathbb{R}^M$

$\vec{x}_{new} \in \mathbb{R}^M$

find i s.t. $\min_i \|\vec{x}_{new} - \vec{x}^{(i)}\|_2$

$O(MN)$ $O(N)$ $O(M)$

Exercise

What is the computation complexity of matrix multiplication of two $N \times N$ matrices? Give the tightest complexity in its simplest form.

$A \in \mathbb{R}^{N \times N}$, $B \in \mathbb{R}^{N \times N}$, $C \in \mathbb{R}^{N \times N}$

$C = AB$

$O(n^3)$

$O(n^3 + n^3 - n^2)$

$\forall i \in \{1..N\}, \forall j \in \{1..N\}$

$$C_{i,j} = \sum_{k=1}^{N} A_{ik} B_{kj}$$

```
C = zeros(N, N)
for i in 1...N
    for j in 1...N
        for k in 1...N
            C[i,j] += A[i,k]*B[k,j]
```

$O(n^4)$

$O(2^n)$

$O(n^n)$

Exercise

Prove that n^3 is in $O(2^n)$

Prove that $n^2 + 100$ is in $O(n^4)$

Prove that $\frac{1}{4}n^2 + n\log(n) + n$ is in $O(n^2)$