CS 170: Algorithms

CS 170: Algorithms

Hello and ...

CS 170: Algorithms

Hello and ...

S

CS 170: Algorithms

Hello and ...

S H

CS 170: Algorithms

Hello and ...

S H H

CS 170: Algorithms

Hello and ...

S H H H

CS 170: Algorithms

Hello and ...

S H H H H

CS 170: Algorithms

Hello and ...

S H H H H .

CS 170: Algorithms

Hello and ...

S H H H H . .

CS 170: Algorithms

Hello and ...

S H H H H . . .

CS 170: Algorithms

Hello and ...

S H H H H . . . .

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)
Page 3: Problem 3

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)
Page 3: Problem 3
Page 4: Problem 4

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)
Page 3: Problem 3
Page 4: Problem 4
Page 5: Problem 5

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)
Page 3: Problem 3
Page 4: Problem 4
Page 5: Problem 5
Page 6: Problem 6

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)
Page 3: Problem 3
Page 4: Problem 4
Page 5: Problem 5
Page 6: Problem 6
Page 7: Extra Page for Problem 2 (second page)

CS 170: Algorithms

Hello and ...

S H H H H . . . . .

Homework Submissions:

One page per problem. Label Problem.

Extra pages for any problem at end.

Example: 2 pages for problem 2.

Page 1: Problem 1
Page 2: Problem 2 (first page)
Page 3: Problem 3
Page 4: Problem 4
Page 5: Problem 5
Page 6: Problem 6
Page 7: Extra Page for Problem 2 (second page)

Last Time: hash functions.

IPs: 2^{32} ips into 250 buckets.
Any hash function $h : IPS \rightarrow \{0, \ldots, 249\}$ has (really) bad bucket.

Last Time: hash functions.

IPs: 2^{32} ips into 250 buckets.
Any hash function $h : IPS \rightarrow \{0, \ldots, 249\}$ has (really) bad bucket.
$(> 160,000,000)$

Last Time: hash functions.

IPs: 2^{32} ips into 250 buckets.
Any hash function $h : IPS \to \{0, \ldots, 249\}$ has (really) bad bucket.
$(> 160,000,000)$

Assumption Alert:

Last Time: hash functions.

IPs: 2^{32} ips into 250 buckets.
Any hash function $h : IPS \rightarrow \{0, \ldots, 249\}$ has (really) bad bucket.
($> 160,000,000$)

Assumption Alert:
The set of keys does not depend on the choice of hash function.

Last Time: hash functions.

IPs: 2^{32} ips into 250 buckets.
Any hash function $h : IPS \to \{0, \ldots, 249\}$ has (really) bad bucket.
($> 160,000,000$)

Assumption Alert:
The set of keys does not depend on the choice of hash function.

Choose randomly from a bunch of hash functions independently from the set of keys.

Last Time: hash functions.

IPs: 2^{32} ips into 250 buckets.
Any hash function $h: IPS \rightarrow \{0, \ldots, 249\}$ has (really) bad bucket.
$(> 160,000,000)$

Assumption Alert:
The set of keys does not depend on the choice of hash function.

Choose randomly from a bunch of hash functions independently from the set of keys.

"A bunch of hash functions" $\equiv$ A class of hash functions.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10)$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte",

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1, 0, 0, 0)$ is “first byte”, $a = (0, 0, 0, 1)$ is “last byte.”

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte", $a = (0,0,0,1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte", $a = (0,0,0,1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1, 0, 0, 0)$ is "first byte", $a = (0, 0, 0, 1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

What is this?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte", $a = (0,0,0,1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

What is this? A hash family.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte", $a = (0,0,0,1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

What is this? A hash family.
Select a random hash function?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte", $a = (0,0,0,1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

What is this? A hash family.
Select a random hash function? Choose a random $a = (a_1, a_2, a_3, a_4)$.
Is it good?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is “first byte”, $a = (0,0,0,1)$ is “last byte.”

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

What is this? A hash family.
Select a random hash function? Choose a random $a = (a_1, a_2, a_3, a_4)$.
Is it good? Probability of collision?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Note: $a = (1,0,0,0)$ is "first byte", $a = (0,0,0,1)$ is "last byte."

Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

What is this? A hash family.
Select a random hash function? Choose a random $a = (a_1, a_2, a_3, a_4)$.
Is it good? Probability of collision?
Random hash function: $\frac{1}{n}$.

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] =$$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

(B) 1

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

(B) 1

(C) $1/n$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

(B) 1

(C) $1/n$ (as if x and y were placed randomly.)

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_4 \pmod{N}.$$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog).

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_4 \pmod{N}.$$

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4 x_4 = Y + a_4 y_4 \pmod{N}.(*)$$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N. (*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N.$$

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N. (*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4 x_4 = Y + a_4 y_4 \pmod N. (*)$$

for $X = a_1 x_1 + a_2 x_2 + a_3 x_3$, and $Y = a_1 y_1 + a_2 y_2 + a_3 y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4 x_4 = Y + a_4 y_4 \pmod N . (*)$$

for $X = a_1 x_1 + a_2 x_2 + a_3 x_3$, and $Y = a_1 y_1 + a_2 y_2 + a_3 y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely 1 value of a_4 that works..

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4 x_4 = Y + a_4 y_4 \pmod{N}.(*)$$

for $X = a_1 x_1 + a_2 x_2 + a_3 x_3$, and $Y = a_1 y_1 + a_2 y_2 + a_3 y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod{N}$.
There is precisely 1 value of a_4 that works..

1

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod{N}$.
There is precisely 1 value of a_4 that works..

1 out

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod{N}$.
There is precisely 1 value of a_4 that works..

1 out of

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N)$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely 1 value of a_4 that works..

1 out of n ...

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N)$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely 1 value of a_4 that works..

1 out of n ... $\frac{1}{n}$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N. (*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N)$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely 1 value of a_4 that works..

1 out of n ... $\frac{1}{n}$ is probability of collision!

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely 1 value of a_4 that works..

1 out of n ... $\frac{1}{n}$ is probability of collision! !

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and $Y = a_1y_1 + a_2y_2 + a_3y_3$.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, $a_4 = (Y - X)(x_4 - y_4)^{-1} \pmod N$.
There is precisely 1 value of a_4 that works..

1 out of n ... $\frac{1}{n}$ is probability of collision! ! !

Wrapping up.

Wrapping up.

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n$

(B) $1/n^2$

(C) 1

Wrapping up.

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n$

(B) $1/n^2$

(C) 1

A. We just argued this.

Wrapping up.

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n$

(B) $1/n^2$

(C) 1

A. We just argued this.

Just as if the keys were placed at random!

Universal hashing.

Design pattern.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0, \ldots, n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:
table size n, domain of size n^k,

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0, \ldots, n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1, \ldots, a_k)$, from $\{0, \ldots, n-1\}$.

$h_a(x_1, \ldots, x_k) = a_1 x_1 + \cdots a_k x_k \mod n.$

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$$h_a(x_1,\ldots,x_k) = a_1 x_1 + \cdots a_k x_k \mod n.$$

This is universal

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$$h_a(x_1,\ldots,x_k) = a_1x_1 + \cdots a_kx_k \mod n.$$

This is universal by the same argument as above

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$$h_a(x_1,\ldots,x_k) = a_1x_1 + \cdots a_kx_k \mod n.$$

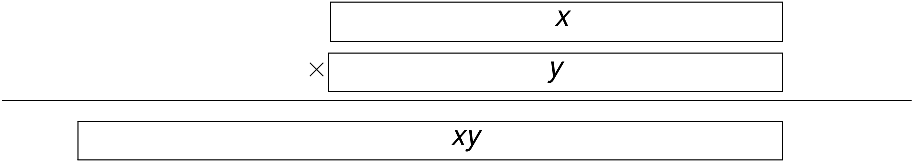
This is universal by the same argument as above if n is prime.

Chapter 2

Divide and conquer.

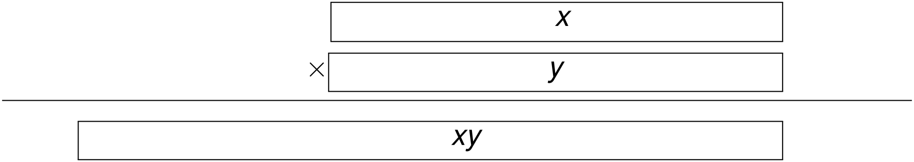
Definition of Multiplication.

n-bit numbers: x, y.



Definition of Multiplication.

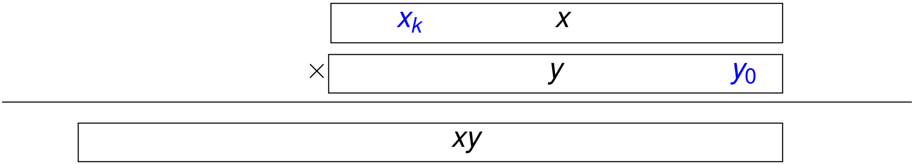
n-bit numbers: x, y.



kth “place” of xy:

Definition of Multiplication.

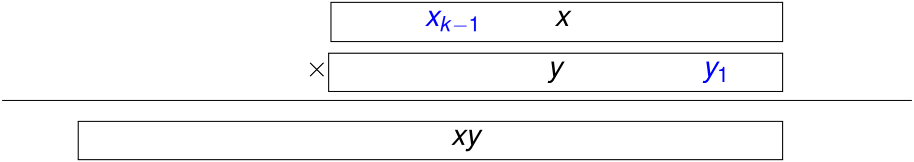
n-bit numbers: x, y.



kth "place" of xy: coefficient of 2^k

Definition of Multiplication.

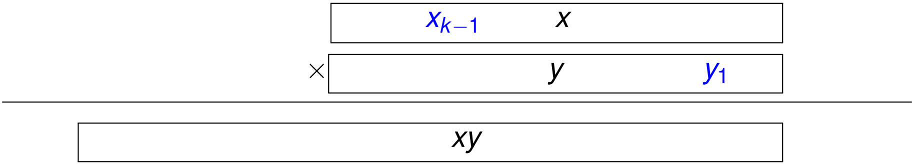
n-bit numbers: x, y.



kth "place" of xy: coefficient of 2^k

Definition of Multiplication.

n-bit numbers: x, y.

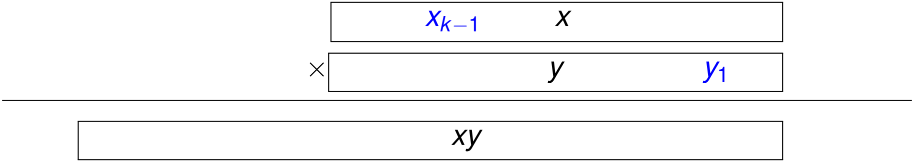


kth "place" of xy: coefficient of 2^k

$$a_k = \sum_{i \leq k} x_i y_{k-i}.$$

Definition of Multiplication.

n-bit numbers: x, y.



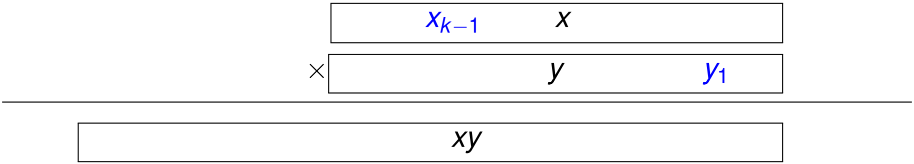
kth "place" of xy: coefficient of 2^k

$$a_k = \sum_{i \leq k} x_i y_{k-i}.$$

$x * y = \sum_{k=0}^{2n} 2^k a_k$.

Definition of Multiplication.

n-bit numbers: x, y.



kth "place" of xy: coefficient of 2^k

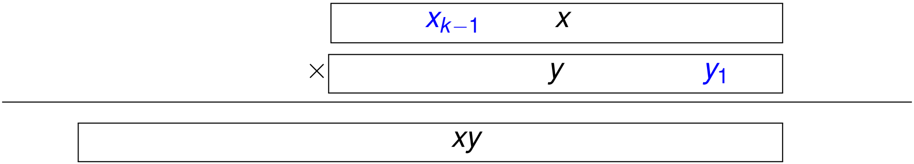
$$a_k = \sum_{i \leq k} x_i y_{k-i}.$$

$x * y = \sum_{k=0}^{2n} 2^k a_k$.

Number of "basic operations":

Definition of Multiplication.

n-bit numbers: x, y.



kth "place" of xy: coefficient of 2^k

$$a_k = \sum_{i \le k} x_i y_{k-i}.$$

$x * y = \sum_{k=0}^{2n} 2^k a_k$.

Number of "basic operations":

$$\sum_{k \le 2n} \min(k, 2n-k) = \Theta(n^2).$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \begin{array}{|c|c|}\hline \qquad x_L \qquad & \qquad x_R \qquad \\ \hline \end{array}$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$
\begin{array}{rcl}
x & = & \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2} x_L + x_R \\
y & = & \boxed{\quad y_L \quad}\boxed{\quad y_R \quad}
\end{array}
$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R$$

$$y = \boxed{\quad y_L \quad}\boxed{\quad y_R \quad} = 2^{n/2}y_L + y_R$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R$$

$$y = \boxed{\quad y_L \quad}\boxed{\quad y_R \quad} = 2^{n/2}y_L + y_R$$

Multiplying out

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$
\begin{aligned}
x &= \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R \\
y &= \boxed{\quad y_L \quad}\boxed{\quad y_R \quad} = 2^{n/2}y_L + y_R
\end{aligned}
$$

Multiplying out

$$x \times y$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R$$

$$y = \boxed{\quad y_L \quad}\boxed{\quad y_R \quad} = 2^{n/2}y_L + y_R$$

Multiplying out

$$x \times y = (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R)$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R$$

$$y = \boxed{\quad y_L \quad}\boxed{\quad y_R \quad} = 2^{n/2}y_L + y_R$$

Multiplying out

$$\begin{aligned} x \times y &= (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R) \\ &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$
\begin{array}{rcl}
x & = & \boxed{\quad x_L \quad}\boxed{\quad x_R \quad} = 2^{n/2}x_L + x_R \\
y & = & \boxed{\quad y_L \quad}\boxed{\quad y_R \quad} = 2^{n/2}y_L + y_R
\end{array}
$$

Multiplying out

$$
\begin{aligned}
x \times y &= (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R) \\
&= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R
\end{aligned}
$$

Four $n/2$-bit multiplications: $x_L y_L$, $x_L y_R$, $x_R y_L$, $x_R y_R$.

Recursive Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = \boxed{\; x_L \;}\boxed{\; x_R \;} = 2^{n/2}x_L + x_R$$
$$y = \boxed{\; y_L \;}\boxed{\; y_R \;} = 2^{n/2}y_L + y_R$$

Multiplying out

$$\begin{aligned} x \times y &= (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R) \\ &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

Four $n/2$-bit multiplications: $x_L y_L$, $x_L y_R$, $x_R y_L$, $x_R y_R$.
Recurrence:

$$T(n) = 4T(\frac{n}{2}) + O(n)$$

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1.

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.
One for each pair of digits!

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.
One for each pair of digits!

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.
One for each pair of digits!

Really?

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.
One for each pair of digits!

Really? Unfolded recusion in my head?!?!

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.
One for each pair of digits!

Really? Unfolded recusion in my head?!?!
How did I really obtain bound?

Recurrence for recursive algorithm.

Recurrence:

$$T(n) = 4T(\frac{n}{2}) + \Theta(n)$$

$T(n)$ is

(A) $\Theta(n)$.

1. $\Theta(n^2)$.
2. $\Theta(n^3)$.

1. Doh... I mean (B)..no (1).
Why?

Idea: $\Theta(n^2)$ base cases.
One for each pair of digits!

Really? Unfolded recusion in my head?!?!
How did I really obtain bound? In a moment.

Demo

Yes?

Demo

Yes? No?

Demo

Yes? No?

$O(n^2)$
$n \to 2n$

Demo

Yes? No?

$O(n^2)$
$n \rightarrow 2n$
Runtime: $T = cn^2 \rightarrow T' = c(2n)^2 = 4(cn^2) = 4T$

Demo

Yes? No?

$O(n^2)$
$n \rightarrow 2n$
Runtime: $T = cn^2 \rightarrow T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$
Runtime: $T \to 3T$.

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$
Runtime: $T \to 3T$.
Asymptotics: $T = cn^w \to c((2n)^w) = T' = 3T = 3(cn^w)$.

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$
Runtime: $T \to 3T$.
Asymptotics: $T = cn^w \to c((2n)^w) = T' = 3T = 3(cn^w)$.
.... $\to 2^w = 3$.

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$
Runtime: $T \to 3T$.
Asymptotics: $T = cn^w \to c((2n)^w) = T' = 3T = 3(cn^w)$.
.... $\to 2^w = 3$. or $w = \log_2 3 \approx 1.58$.

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$
Runtime: $T \to 3T$.
Asymptotics: $T = cn^w \to c((2n)^w) = T' = 3T = 3(cn^w)$.
.... $\to 2^w = 3$. or $w = \log_2 3 \approx 1.58$.
Python multiply: $O(n^{\log_2 3})$

Demo

Yes? No?

$O(n^2)$
$n \to 2n$
Runtime: $T = cn^2 \to T' = c(2n)^2 = 4(cn^2) = 4T$

...Python multiply?

$n \to 2n$
Runtime: $T \to 3T$.
Asymptotics: $T = cn^w \to c((2n)^w) = T' = 3T = 3(cn^w)$.
.... $\to 2^w = 3$. or $w = \log_2 3 \approx 1.58$.
Python multiply: $O(n^{\log_2 3})$
Much better than grade school.

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i})$$

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Gauss's trick.

$$(a + b\mathbf{i})(c + d\mathbf{i}) = (ac - bd) + (ad + bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a + b)(c + d) = ac + ad + bc + bd.$$

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one!

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

$$(ac-bd) = P_2 - P_3.$$

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

$$(ac-bd) = P_2 - P_3.$$

$$(ad+bc) = P_1 - P_2 - P_3.$$

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

$$(ac-bd) = P_2 - P_3.$$

$$(ad+bc) = P_1 - P_2 - P_3.$$

Only three multiplications.

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

$$(ac-bd) = P_2 - P_3.$$

$$(ad+bc) = P_1 - P_2 - P_3.$$

Only three multiplications. An extra addition though!

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

$$(ac-bd) = P_2 - P_3.$$

$$(ad+bc) = P_1 - P_2 - P_3.$$

Only three multiplications. An extra addition though!
Which is harder? multiplication or addition.

Gauss's trick.

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(ad+bc)\mathbf{i}.$$

Four multiplications: ac, bd, ad, bd.

Drop the i:

$$P_1 = (a+b)(c+d) = ac+ad+bc+bd.$$

Four multiplications from one! ..but all added up.

Two more multiplications: $P_2 = ac$, $P_3 = bd$.

$$(ac-bd) = P_2 - P_3.$$

$$(ad+bc) = P_1 - P_2 - P_3.$$

Only three multiplications. An extra addition though!
Which is harder? multiplication or addition.
Multiplication!

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R$$

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2} x_L + x_R \quad ; \quad y = 2^{n/2} y_L + y_R$$

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$

$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$

$$x \times y \;=\; 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$

$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Used four $\frac{n}{2}$-bit multiplications: $x_L y_L$, $x_L y_R$, $x_R y_L$, $x_R y_R$.

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$
$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Used four $\frac{n}{2}$-bit multiplications: $x_L y_L$, $x_L y_R$, $x_R y_L$, $x_R y_R$.

Can you compute three terms with 3 multiplications?

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$
$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Used four $\frac{n}{2}$-bit multiplications: $x_L y_L$, $x_L y_R$, $x_R y_L$, $x_R y_R$.

Can you compute three terms with 3 multiplications?

(A) Yes.

(B) No

Faster Algorithm for Multiplication.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$
$$x \times y \;=\; 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Used four $\frac{n}{2}$-bit multiplications: $x_L y_L$, $x_L y_R$, $x_R y_L$, $x_R y_R$.

Can you compute three terms with 3 multiplications?

(A) Yes.

(B) No

(A) Yes.

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2} x_L + x_R \quad ; \quad y = 2^{n/2} y_L + y_R$$

$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$

$$x \times y \;=\; 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute

$$P_1 = (x_L + x_R)(y_L + y_R)$$

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$

$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute

$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2} x_L + x_R \quad ; \quad y = 2^{n/2} y_L + y_R$$

$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute

$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Two more: $P_2 = x_L y_L$, $P_3 = x_R y_R$.

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2} x_L + x_R \quad ; \quad y = 2^{n/2} y_L + y_R$$
$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute

$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Two more: $P_2 = x_L y_L$, $P_3 = x_R y_R$. $(x_L y_R + x_R y_L) = P_1 - P_2 - P_3$

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$
$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute
$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Two more: $P_2 = x_L y_L$, $P_3 = x_R y_R$. $(x_L y_R + x_R y_L) = P_1 - P_2 - P_3$
3 multiplications!

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$
$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute
$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Two more: $P_2 = x_L y_L$, $P_3 = x_R y_R$. $(x_L y_R + x_R y_L) = P_1 - P_2 - P_3$
3 multiplications!

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$
$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute
$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Two more: $P_2 = x_L y_L$, $P_3 = x_R y_R$. $(x_L y_R + x_R y_L) = P_1 - P_2 - P_3$
3 multiplications!

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Technically: $\frac{n}{2} + 1$ bit multiplication.

Three multiplications and faster algorithm.

Two n-bit numbers: x, y.

$$x = 2^{n/2}x_L + x_R \quad ; \quad y = 2^{n/2}y_L + y_R$$

$$x \times y = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Need 3 terms: $x_L y_L$, $x_L y_R + x_R y_L$, $x_R y_R$.

Compute

$$P_1 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_L y_R + x_R y_L + x_R y_R.$$

Two more: $P_2 = x_L y_L$, $P_3 = x_R y_R$. $(x_L y_R + x_R y_L) = P_1 - P_2 - P_3$
3 multiplications!

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Technically: $\frac{n}{2} + 1$ bit multiplication. Don't worry.

Analysis of runtime.

Recurrence for “fast algorithm”.

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Analysis of runtime.

Recurrence for “fast algorithm”.

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$

(B) $\Theta(n^2)$

Analysis of runtime.

Recurrence for “fast algorithm”.

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$

(B) $\Theta(n^2)$

(C) $\Theta(n^{\log_2 3})$

Analysis of runtime.

Recurrence for “fast algorithm”.

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$

(B) $\Theta(n^2)$

(C) $\Theta(n^{\log_2 3})$

(C)

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$

(B) $\Theta(n^2)$

(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Analysis of runtime.

Recurrence for “fast algorithm”.

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58...})$$

Analysis of runtime.

Recurrence for “fast algorithm”.

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58\ldots})!$$

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58\ldots})!!$$

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58...})!!!$$

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58...})!!!!$$

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58...})!!!!!!$$

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58...})!!!!!!$$

Analysis of runtime.

Recurrence for "fast algorithm".

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

Runtime is

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^{\log_2 3})$

(C) Idea: number of base cases is $n^{\log_2 3}$.
More in a moment.

So multiplication algorithm with ..

$$T(n) = 3T(\frac{n}{2}) + \Theta(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58...})!!!!!!!!$$

We stopped here in lecture due to questions; which are good to have.

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc}$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb}$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log:

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithm Quiz: $a^{\log_b n} = n^{\log_b a}$?

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithm Quiz: $a^{\log_b n} = n^{\log_b a}$?

Yes!

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithm Quiz: $a^{\log_b n} = n^{\log_b a}$?

Yes!

$$a^{\log_b n}$$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithm Quiz: $a^{\log_b n} = n^{\log_b a}$?

Yes!

$$a^{\log_b n} = (b^{\log_b a})^{\log_b n}$$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithm Quiz: $a^{\log_b n} = n^{\log_b a}$?

Yes!

$$a^{\log_b n} = (b^{\log_b a})^{\log_b n} = (b^{\log_b n})^{\log_b a}$$

Logarithms reminder.

Exponents Quiz: $(a^b)^c = (a^c)^b$?

Yes? No?

Yes. $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b$.

Definition of log: $a = b^{\log_b a}$

Logarithm Quiz: $a^{\log_b n} = n^{\log_b a}$?

Yes!

$$a^{\log_b n} = (b^{\log_b a})^{\log_b n} = (b^{\log_b n})^{\log_b a} = n^{\log_b a}$$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})\quad T(\frac{n}{2})\quad T(\frac{n}{2})\quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\cdots \quad \cdots \quad \cdots$				
$T(\frac{n}{4})\cdots\quad T(\frac{n}{4})\quad T(\frac{n}{4})\quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \; \swarrow \quad \searrow \; \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \; T(\frac{n}{4}) \quad T(\frac{n}{4}) \; \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$T(\frac{n}{4})\cdots$ $T(\frac{n}{4})$ $T(\frac{n}{4})$ $\cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \; \swarrow \quad \searrow \; \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \quad T(\frac{n}{4}) \qquad T(\frac{n}{4}) \quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$4^{\log n}$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \swarrow \quad \searrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \quad T(\frac{n}{4}) \qquad T(\frac{n}{4}) \quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n}$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})\ \ T(\frac{n}{2})\ \ T(\frac{n}{2})\ \ T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$T(\frac{n}{4})\cdots\ T(\frac{n}{4})\ \ \ T(\frac{n}{4})\ \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems.

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \swarrow \quad \searrow \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \quad T(\frac{n}{4}) \qquad T(\frac{n}{4}) \quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \searrow \swarrow \searrow \swarrow \searrow \swarrow \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1.

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \; \swarrow \quad \searrow \; \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \; T(\frac{n}{4}) \quad T(\frac{n}{4}) \; \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})\quad T(\frac{n}{2})\quad T(\frac{n}{2})\quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$T(\frac{n}{4})\cdots\ T(\frac{n}{4})\quad\cdots\quad T(\frac{n}{4})\ \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\cdots$ $\cdots$ $\cdots$				
$T(\frac{n}{4})\cdots$ $T(\frac{n}{4})$ $T(\frac{n}{4})$ $\cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.
Total Work:

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \;\swarrow \quad \searrow\; \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \qquad \cdots \qquad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \quad T(\frac{n}{4}) \qquad T(\frac{n}{4}) \quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.

Total Work: cn

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\cdots$ $\cdots$ $\cdots$				
$T(\frac{n}{4})\cdots$ $T(\frac{n}{4})$ $T(\frac{n}{4})$ $\cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.
Total Work: $cn + 2cn$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \swarrow \searrow \searrow$				
$T(\frac{n}{2})\ T(\frac{n}{2})\ T(\frac{n}{2})\ T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \searrow \swarrow \searrow \swarrow \searrow \swarrow \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.

Total Work: $cn + 2cn + 4cn + \cdots$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \; \swarrow \quad \searrow \; \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \quad T(\frac{n}{4}) \qquad T(\frac{n}{4}) \quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.
Total Work: $cn + 2cn + 4cn + \cdots + cn^2$

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$ $T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$T(\frac{n}{4})\cdots$ $T(\frac{n}{4})$ $\cdots$ $T(\frac{n}{4})$ $\cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.
Total Work: $cn + 2cn + 4cn + \cdots + cn^2 = O(n^2)$.

Solving recurrences.

$$T(n) = 4T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \swarrow \quad \searrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	4	$\frac{n}{2}$	$c(\frac{n}{2})$	$2cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots \quad T(\frac{n}{4}) \qquad T(\frac{n}{4}) \quad \cdots T(\frac{n}{4})$	4^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$4cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	4^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$2^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.
$4^{\log n} = 2^{2\log n} = n^2$ base case problems. size 1. Work/Prob: c Work: cn^2.
Total Work: $cn + 2cn + 4cn + \cdots + cn^2 = O(n^2)$. Geometric series.

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow\cdots\searrow \quad \cdots \quad \swarrow\cdots\searrow$				
$T(\frac{n}{4})\cdots T(\frac{n}{4}) \quad T(\frac{n}{4})\cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \downarrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow\cdots\searrow \quad \cdots \quad \swarrow\cdots\searrow$				
$T(\frac{n}{4})\cdots T(\frac{n}{4}) \quad T(\frac{n}{4})\cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \; \downarrow \; \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n}$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

$$\begin{array}{ccccc}
\text{Recursion Tree} & \text{\# probs} & \text{sz} & \text{tm./prob} & \text{tm./lvl} \\
T(n) & 1 & n & cn & cn \\
\swarrow \quad \downarrow \quad \searrow & & & & \\
T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) & 3 & \frac{n}{2} & c(\frac{n}{2}) & (\frac{3}{2})cn \\
\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow & & & & \\
T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4}) & 3^2 & \frac{n}{4} & c(\frac{n}{4}) & (\frac{3}{2})^2 cn \\
\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow & \vdots & \vdots & \vdots & \vdots \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & 3^i & \frac{n}{2^i} & c(\frac{n}{2^i}) & (\frac{3}{2})^i cn
\end{array}$$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems.

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \downarrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1.

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c.

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

$$\begin{array}{ccccc}
\text{Recursion Tree} & \text{\# probs} & \text{sz} & \text{tm./prob} & \text{tm./lvl} \\
T(n) & 1 & n & cn & cn \\
\swarrow \ \downarrow \ \searrow & & & & \\
T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2}) & 3 & \frac{n}{2} & c(\frac{n}{2}) & (\frac{3}{2})cn \\
\swarrow\cdots\searrow \quad \cdots \quad \swarrow\cdots\searrow & & & & \\
T(\frac{n}{4})\cdots T(\frac{n}{4}) \quad T(\frac{n}{4})\cdots T(\frac{n}{4}) & 3^2 & \frac{n}{4} & c(\frac{n}{4}) & (\frac{3}{2})^2 cn \\
\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow & \vdots & \vdots & \vdots & \vdots \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & 3^i & \frac{n}{2^i} & c(\frac{n}{2^i}) & (\frac{3}{2})^i cn
\end{array}$$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \downarrow \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work:

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \downarrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work: cn

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work: $cn + (\frac{3}{2})cn$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow\cdots\searrow \quad \cdots \quad \swarrow\cdots\searrow$				
$T(\frac{n}{4})\cdots T(\frac{n}{4}) \quad T(\frac{n}{4})\cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work: $cn + (\frac{3}{2})cn + \cdots$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \downarrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work: $cn + (\frac{3}{2})cn + \cdots + cn^{\log_2 3}$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \ \downarrow \ \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work: $cn + (\frac{3}{2})cn + \cdots + cn^{\log_2 3} = O(n^{\log_2 3})$

Fast multiplication.

$$T(n) = 3T(\frac{n}{2}) + cn; \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn	cn
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{2}) \quad T(\frac{n}{2}) \quad T(\frac{n}{2})$	3	$\frac{n}{2}$	$c(\frac{n}{2})$	$(\frac{3}{2})cn$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{4}) \cdots T(\frac{n}{4}) \quad T(\frac{n}{4}) \cdots T(\frac{n}{4})$	3^2	$\frac{n}{4}$	$c(\frac{n}{4})$	$(\frac{3}{2})^2 cn$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	3^i	$\frac{n}{2^i}$	$c(\frac{n}{2^i})$	$(\frac{3}{2})^i cn$

$\frac{n}{2^i} = 1$ when $i = \log_2 n \implies$ Depth: $d = \log_2 n$.

$3^{\log_2 n} = n^{\log_2 3}$ base case problems. size 1. Work/Prob: c. Work: $cn^{\log_2 3}$.

Total Work: $cn + (\frac{3}{2})cn + \cdots + cn^{\log_2 3} = O(n^{\log_2 3})$ Geometric series.

Divide and Conquer: In general.

$$T(n) = aT(\frac{n}{b}) + O(n^d); \qquad T(1) = c$$

Divide and Conquer: In general.

$$T(n) = aT(\frac{n}{b}) + O(n^d); \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn^d	cn^d
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{b}) \quad T(\frac{n}{b}) \quad T(\frac{n}{b})$	a	$\frac{n}{b}$	$c(\frac{n}{b})^d$	$(\frac{a}{b^d})cn^d$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2}) \quad T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2})$	a^2	$\frac{n}{b^2}$	$c(\frac{n}{b^2})^d$	$(\frac{a}{b^d})^2 cn^d$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	a^i	$\frac{n}{b^i}$	$c(\frac{n}{b^i})^d$	$(\frac{a}{b^d})^i cn^d$

Divide and Conquer: In general.

$$T(n) = aT(\frac{n}{b}) + O(n^d); \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn^d	cn^d
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{b}) \quad T(\frac{n}{b}) \quad T(\frac{n}{b})$	a	$\frac{n}{b}$	$c(\frac{n}{b})^d$	$(\frac{a}{b^d})cn^d$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2}) \quad T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2})$	a^2	$\frac{n}{b^2}$	$c(\frac{n}{b^2})^d$	$(\frac{a}{b^d})^2 cn^d$
$\swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow \ \swarrow \ \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	a^i	$\frac{n}{b^i}$	$c(\frac{n}{b^i})^d$	$(\frac{a}{b^d})^i cn^d$

$\frac{n}{b^i} = 1$ when $i = \log_b n$

Divide and Conquer: In general.

$$T(n) = aT(\frac{n}{b}) + O(n^d); \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn^d	cn^d
$\swarrow \quad \downarrow \quad \searrow$				
$T(\frac{n}{b}) \quad T(\frac{n}{b}) \quad T(\frac{n}{b})$	a	$\frac{n}{b}$	$c(\frac{n}{b})^d$	$(\frac{a}{b^d})cn^d$
$\swarrow \cdots \searrow \quad \cdots \quad \swarrow \cdots \searrow$				
$T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2}) \quad T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2})$	a^2	$\frac{n}{b^2}$	$c(\frac{n}{b^2})^d$	$(\frac{a}{b^d})^2 cn^d$
$\swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow \; \swarrow \; \searrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$	a^i	$\frac{n}{b^i}$	$c(\frac{n}{b^i})^d$	$(\frac{a}{b^d})^i cn^d$

$\frac{n}{b^i} = 1$ when $i = \log_b n \implies$ Depth: $k = \log_b n$.

Divide and Conquer: In general.

$$T(n) = aT(\frac{n}{b}) + O(n^d); \qquad T(1) = c$$

Recursion Tree	# probs	sz	tm./prob	tm./lvl
$T(n)$	1	n	cn^d	cn^d
$T(\frac{n}{b}) \quad T(\frac{n}{b}) \quad T(\frac{n}{b})$	a	$\frac{n}{b}$	$c(\frac{n}{b})^d$	$(\frac{a}{b^d})cn^d$
$T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2}) \quad T(\frac{n}{b^2}) \cdots T(\frac{n}{b^2})$	a^2	$\frac{n}{b^2}$	$c(\frac{n}{b^2})^d$	$(\frac{a}{b^d})^2 cn^d$
	a^i	$\frac{n}{b^i}$	$c(\frac{n}{b^i})^d$	$(\frac{a}{b^d})^i cn^d$

$\frac{n}{b^i} = 1$ when $i = \log_b n \implies$ Depth: $k = \log_b n$.

Level i work: $(\frac{a}{b^d})^i n^d$.

Master's Theorem

Depth: $\log_b n$.

Master's Theorem

Depth: $\log_b n$.
Level i work:

Master's Theorem

Depth: $\log_b n$.
Level i work:

$$(\frac{a}{b^d})^i n^d.$$

Master's Theorem

Depth: $\log_b n$.
Level i work:

$$(\frac{a}{b^d})^i n^d.$$

Total:

$$n^d \sum_{i=0}^{\log_b n} (\frac{a}{b^d})^i$$

Master's Theorem

Depth: $\log_b n$.

Level i work:

$$(\frac{a}{b^d})^i n^d.$$

Total:

$$n^d \sum_{i=0}^{\log_b n} (\frac{a}{b^d})^i$$

Geometric series:

Master's Theorem

Depth: $\log_b n$.
Level i work:

$$(\frac{a}{b^d})^i n^d.$$

Total:

$$n^d \sum_{i=0}^{\log_b n} (\frac{a}{b^d})^i$$

Geometric series: If $\frac{a}{b^d} < 1$ ($d > \log_b a$), first term dominates

$$O(n^d),$$

Master's Theorem

Depth: $\log_b n$.
Level i work:

$$(\frac{a}{b^d})^i n^d.$$

Total:

$$n^d \sum_{i=0}^{\log_b n} (\frac{a}{b^d})^i$$

Geometric series: If $\frac{a}{b^d} < 1$ ($d > \log_b a$), first term dominates

$$O(n^d),$$

if $\frac{a}{b^d} > 1$ ($d < \log_b a$), last term dominates.

$$O(n^{\log_b a}),$$

Master's Theorem

Depth: $\log_b n$.
Level i work:

$$(\frac{a}{b^d})^i n^d.$$

Total:

$$n^d \sum_{i=0}^{\log_b n} (\frac{a}{b^d})^i$$

Geometric series: If $\frac{a}{b^d} < 1$ ($d > \log_b a$), first term dominates

$$O(n^d),$$

if $\frac{a}{b^d} > 1$ ($d < \log_b a$), last term dominates.

$$O(n^{\log_b a}),$$

and if $\frac{a}{b^d} = 1$ ($d = \log_b a$), then all terms are the same

$$O(n^d \log_b n).$$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$d > \log_b a \qquad T(n) = O(n^d)$
$d < \log_b a \qquad T(n) = O(n^{\log_b a})$
$d = \log_b a \qquad T(n) = O(n^d \log_b n)$.

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$$\begin{array}{ll} d > \log_b a & T(n) = O(n^d) \\ d < \log_b a & T(n) = O(n^{\log_b a}) \\ d = \log_b a & T(n) = O(n^d \log_b n). \end{array}$$

$T(n) = 4T(\frac{n}{2}) + O(n)$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$d > \log_b a \quad T(n) = O(n^d)$
$d < \log_b a \quad T(n) = O(n^{\log_b a})$
$d = \log_b a \quad T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have
$d > \log_b a \quad T(n) = O(n^d)$
$d < \log_b a \quad T(n) = O(n^{\log_b a})$
$d = \log_b a \quad T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have
$d > \log_b a \quad T(n) = O(n^d)$
$d < \log_b a \quad T(n) = O(n^{\log_b a})$
$d = \log_b a \quad T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$d > \log_b a \qquad T(n) = O(n^d)$
$d < \log_b a \qquad T(n) = O(n^{\log_b a})$
$d = \log_b a \qquad T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$d > \log_b a \quad T(n) = O(n^d)$
$d < \log_b a \quad T(n) = O(n^{\log_b a})$
$d = \log_b a \quad T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$$\begin{array}{ll} d > \log_b a & T(n) = O(n^d) \\ d < \log_b a & T(n) = O(n^{\log_b a}) \\ d = \log_b a & T(n) = O(n^d \log_b n). \end{array}$$

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.
$1 > \log_2 1 = 0$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$d > \log_b a \quad T(n) = O(n^d)$
$d < \log_b a \quad T(n) = O(n^{\log_b a})$
$d = \log_b a \quad T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.
$1 > \log_2 1 = 0 \implies T(n) = O(n)$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$d > \log_b a$	$T(n) = O(n^d)$
$d < \log_b a$	$T(n) = O(n^{\log_b a})$
$d = \log_b a$	$T(n) = O(n^d \log_b n)$.

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.
$1 > \log_2 1 = 0 \implies T(n) = O(n)$

$T(n) = 2T(\frac{n}{2}) + O(n)$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$$
\begin{array}{ll}
d > \log_b a & T(n) = O(n^d) \\
d < \log_b a & T(n) = O(n^{\log_b a}) \\
d = \log_b a & T(n) = O(n^d \log_b n).
\end{array}
$$

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.
$1 > \log_2 1 = 0 \implies T(n) = O(n)$

$T(n) = 2T(\frac{n}{2}) + O(n)$ $a = 2$, $b = 2$, and $d = 1$.

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$$\begin{array}{ll} d > \log_b a & T(n) = O(n^d) \\ d < \log_b a & T(n) = O(n^{\log_b a}) \\ d = \log_b a & T(n) = O(n^d \log_b n). \end{array}$$

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.
$1 > \log_2 1 = 0 \implies T(n) = O(n)$

$T(n) = 2T(\frac{n}{2}) + O(n)$ $a = 2$, $b = 2$, and $d = 1$.
$1 = \log_2 2$

Master's Theorem: examples.

For a recurrence $T(n) = aT(n/b) + O(n^d)$
We have

$$\begin{array}{ll} d > \log_b a & T(n) = O(n^d) \\ d < \log_b a & T(n) = O(n^{\log_b a}) \\ d = \log_b a & T(n) = O(n^d \log_b n). \end{array}$$

$T(n) = 4T(\frac{n}{2}) + O(n)$ $a = 4$, $b = 2$, and $d = 1$.
$d = 1 < 2 = \log_2 4 = \log_b a \implies T(n) = O(n^{\log_b a}) = O(n^2)$.

$T(n) = T(\frac{n}{2}) + O(n)$ $a = 1$, $b = 2$, and $d = 1$.
$1 > \log_2 1 = 0 \implies T(n) = O(n)$

$T(n) = 2T(\frac{n}{2}) + O(n)$ $a = 2$, $b = 2$, and $d = 1$.
$1 = \log_2 2 \implies T(n) = O(n \log n)$

See you ..

..on Thursday.