

Algorithms and datastructures I

Lecture 10: universal hashing and divide & conquer

Jan Hubička

Department of Applied Mathematics
Charles University
Prague

April 21 2020

Hash functions

Hash function is a function h from universe $\mathcal{U}$ to set $\mathcal{P} = \{0, 1, \dots, p-1\}$ (of **hashes**).

Hash table with separate chaining for set $S \subseteq U$ with hash function $h: \mathcal{U} \rightarrow \mathcal{P}$.

Hash table is an array H of linked lists indexed by $\mathcal{P}$. List $H[i]$ contains all elements e of set S such that $h(e) = i$.

Assumptions

1. $h(x)$ can be computed in $O(1)$.
2. $h(x)$ “behaves randomly”.

Hash functions

Hash function is a function h from universe $\mathcal{U}$ to set $\mathcal{P} = \{0, 1, \dots, p-1\}$ (of **hashes**).

Hash table with separate chaining for set $S \subseteq U$ with hash function $h: \mathcal{U} \rightarrow \mathcal{P}$.

Hash table is an array H of linked lists indexed by $\mathcal{P}$. List $H[i]$ contains all elements e of set S such that $h(e) = i$.

Assumptions

1. $h(x)$ can be computed in $O(1)$.
2. $h(x)$ “behaves randomly”.

Observation

Every entry of the hash table will contain approximately $\frac{|S|}{p}$ elements.

Corollary

Operations **FIND**, **INSERT** and **DELETE** will run in $O(|S|)$ however expected (average) runtime is only $O(\frac{|S|}{p})$.

Corollary

Putting $p \sim |S|$ we get **FIND**, **INSERT** and **DELETE** is running on average approximately in $O(1)$.

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is c -universal for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is **c -universal** for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Lemma

Let $\mathcal{S}$ be c -universal system of hash functions $\mathcal{U} \rightarrow \{0, 1, \dots, p\}$. Let $x_1, x_2, \dots, x_n, y$ be pairwise different elements of $\mathcal{U}$. Then

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] \leq \frac{cn}{p}.$$

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is **c -universal** for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Lemma

Let $\mathcal{S}$ be c -universal system of hash functions $\mathcal{U} \rightarrow \{0, 1, \dots, p\}$. Let $x_1, x_2, \dots, x_n, y$ be pairwise different elements of $\mathcal{U}$. Then

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] \leq \frac{cn}{p}.$$

The lemma shows expected runtime of **INSERT**, **FIND** and **DELETE** with separate chaining.

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is **c -universal** for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Lemma

Let $\mathcal{S}$ be c -universal system of hash functions $\mathcal{U} \rightarrow \{0, 1, \dots, p\}$. Let $x_1, x_2, \dots, x_n, y$ be pairwise different elements of $\mathcal{U}$. Then

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] \leq \frac{cn}{p}.$$

Proof.

We define indicators $l_1, l_2, \dots, l_n$:

$$l_i = \begin{cases} 0 & \text{if } h(x_i) \neq h(y) \\ 1 & \text{if } h(x_i) = h(y). \end{cases}$$

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is **c -universal** for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Lemma

Let $\mathcal{S}$ be c -universal system of hash functions $\mathcal{U} \rightarrow \{0, 1, \dots, p\}$. Let $x_1, x_2, \dots, x_n, y$ be pairwise different elements of $\mathcal{U}$. Then

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] \leq \frac{cn}{p}.$$

Proof.

We define indicators $l_1, l_2, \dots, l_n$:

$$\mathbb{E}[l_i] = \Pr[l_i = 1] \leq \frac{c}{p}. \text{ (by universality)}$$

$$l_i = \begin{cases} 0 & \text{if } h(x_i) \neq h(y) \\ 1 & \text{if } h(x_i) = h(y). \end{cases}$$

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is **c -universal** for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Lemma

Let $\mathcal{S}$ be c -universal system of hash functions $\mathcal{U} \rightarrow \{0, 1, \dots, p\}$. Let $x_1, x_2, \dots, x_n, y$ be pairwise different elements of $\mathcal{U}$. Then

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] \leq \frac{cn}{p}.$$

Proof.

We define indicators $l_1, l_2, \dots, l_n$:

$$l_i = \begin{cases} 0 & \text{if } h(x_i) \neq h(y) \\ 1 & \text{if } h(x_i) = h(y). \end{cases}$$

$$\mathbb{E}[l_i] = \Pr[l_i = 1] \leq \frac{c}{p}. \text{ (by universality)}$$

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] = \sum_{1 \leq i \leq n} \mathbb{E}[l_i].$$

Universal hashing

Definition (c -universal system of hash functions)

System $\mathcal{S}$ of hash functions from universe $\mathcal{U}$ to $\{0, 1, \dots, p-1\}$ is **c -universal** for given $c \geq 1$ if for every $x, y \in \mathcal{U}$, $x \neq y$

$$\Pr_{h \in \mathcal{S}}[h(x) = h(y)] \leq \frac{c}{p}.$$

Lemma

Let $\mathcal{S}$ be c -universal system of hash functions $\mathcal{U} \rightarrow \{0, 1, \dots, p\}$. Let $x_1, x_2, \dots, x_n, y$ be pairwise different elements of $\mathcal{U}$. Then

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] \leq \frac{cn}{p}.$$

Proof.

We define indicators $l_1, l_2, \dots, l_n$:

$$l_i = \begin{cases} 0 & \text{if } h(x_i) \neq h(y) \\ 1 & \text{if } h(x_i) = h(y). \end{cases}$$

$$\mathbb{E}[l_i] = \Pr[l_i = 1] \leq \frac{c}{p}. \text{ (by universality)}$$

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] = \sum_{1 \leq i \leq n} \mathbb{E}[l_i].$$

$$\mathbb{E}_{h \in \mathcal{S}}[\#i: h(x_i) = h(y)] = \sum_{1 \leq i \leq n} \Pr[l_i = 1] \leq \frac{cn}{p}.$$



1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq 0\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. \text{ (} a_i x_i \text{ is the scalar product).}$$

1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq 0\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. \text{ (} a_i x_i \text{ is the scalar product).}$$

Theorem

$\mathcal{S}$ is 1-universal.

1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq \vec{0}\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. \text{ (} a_i x_i \text{ is the scalar product).}$$

Theorem

$\mathcal{S}$ is 1-universal.

Proof.

Set $\vec{x} \neq \vec{y} \in \mathbb{Z}_p^d$. WLOG $x_d \neq y_d$. What is $\Pr_{\vec{a} \in \mathbb{Z}_p^d}[\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}]$?

1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq \vec{0}\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. \text{ (} a_i x_i \text{ is the scalar product).}$$

Theorem

$\mathcal{S}$ is 1-universal.

Proof.

Set $\vec{x} \neq \vec{y} \in \mathbb{Z}_p^d$. WLOG $x_d \neq y_d$. What is $\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}]$?

Put $\vec{z} = \vec{x} - \vec{y}$.

($\vec{a}\vec{x} \equiv \vec{a}\vec{y}$ means $\vec{a}\vec{z} = \vec{0} \pmod{p}$)

$$\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{z} \equiv \vec{0}] =$$

1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq \vec{0}\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. (a_i x_i \text{ is the scalar product}).$$

Theorem

$\mathcal{S}$ is 1-universal.

Proof.

Set $\vec{x} \neq \vec{y} \in \mathbb{Z}_p^d$. WLOG $x_d \neq y_d$. What is $\Pr_{\vec{a} \in \mathbb{Z}_p^d}[\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}]$?

Put $\vec{z} = \vec{x} - \vec{y}$.

($\vec{a}\vec{x} \equiv \vec{a}\vec{y}$ means $\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}$)

$$\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{x} \equiv \vec{a}\vec{y}] = \Pr_{\vec{a} \in \mathbb{Z}_p^d} \left[\sum_{i=1}^d a_i z_i \equiv 0 \right] =$$

1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq \vec{0}\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. \text{ (} a_i x_i \text{ is the scalar product).}$$

Theorem

$\mathcal{S}$ is 1-universal.

Proof.

Set $\vec{x} \neq \vec{y} \in \mathbb{Z}_p^d$. WLOG $x_d \neq y_d$. What is $\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}]$?

Put $\vec{z} = \vec{x} - \vec{y}$.

($\vec{a}\vec{x} \equiv \vec{a}\vec{y}$ means $\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}$)

$$\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{x} \equiv \vec{a}\vec{y}] = \Pr_{\vec{a} \in \mathbb{Z}_p^d} \left[\sum_{i=1}^d a_i z_i \equiv 0 \right] = \Pr_{\vec{a} \in \mathbb{Z}_p^d} \left[\sum_{i=1}^{d-1} a_i z_i + a_d z_d \equiv 0 \right].$$

1-universal system

System of functions $\mathcal{S}: \mathbb{Z}_p^d \rightarrow \{0, 1, \dots, p-1\}$

Let p be a prime number, $\mathcal{P} = \mathbb{Z}_p$ (ring modulo p), $\mathcal{U} = \mathbb{Z}_p^d$ (vectors of length d in $\mathbb{Z}_p$).

$$\mathcal{S} = \{h_{\vec{a}}: \vec{a} \in \mathbb{Z}_p^d, \vec{a} \neq \vec{0}\} \text{ where } h_{\vec{a}}(x) = \vec{a}\vec{x} = \sum_{i=1}^d a_i x_i \pmod{p}. \text{ (} a_i x_i \text{ is the scalar product).}$$

Theorem

$\mathcal{S}$ is 1-universal.

Proof.

Set $\vec{x} \neq \vec{y} \in \mathbb{Z}_p^d$. WLOG $x_d \neq y_d$. What is $\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}]$?

Put $\vec{z} = \vec{x} - \vec{y}$.

($\vec{a}\vec{x} \equiv \vec{a}\vec{y}$ means $\vec{a}\vec{x} = \vec{a}\vec{y} \pmod{p}$)

$$\Pr_{\vec{a} \in \mathbb{Z}_p^d} [\vec{a}\vec{x} \equiv \vec{a}\vec{y}] = \Pr_{\vec{a} \in \mathbb{Z}_p^d} \left[\sum_{i=1}^d a_i z_i \equiv 0 \right] = \Pr_{\vec{a} \in \mathbb{Z}_p^d} \left[\sum_{i=1}^{d-1} a_i z_i + a_d z_d \equiv 0 \right].$$

$\sum_{i=1}^{d-1} a_i z_i + a_d z_d \equiv 0$ happens only if $\sum_{i=1}^{d-1} a_i z_i \equiv -a_d z_d$. This has probability $\frac{1}{p}$. □

Divide & Conquer

“Divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide-and-conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.”



John von Neumann

Divide & Conquer

“Divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide-and-conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.”



John von Neumann

MergeSort ($x_1, \dots, x_n$), John von Neumann, 1945

1. $(y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}) \leftarrow \text{MergeSort}(x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor})$
2. $(z_1, \dots, z_{\lceil \frac{n}{2} \rceil}) \leftarrow \text{MergeSort}(x_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, x_n)$
3. Return Merge $((y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}), (z_1, \dots, z_{\lceil \frac{n}{2} \rceil}))$.

Divide & Conquer

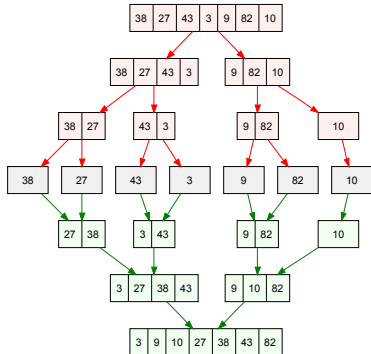
“Divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide-and-conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.”



John von Neumann

MergeSort $(x_1, \dots, x_n)$, John von Neumann, 1945

1. $(y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}) \leftarrow \text{MergeSort}(x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor})$
2. $(z_1, \dots, z_{\lceil \frac{n}{2} \rceil}) \leftarrow \text{MergeSort}(x_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, x_n)$
3. Return Merge $((y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}), (z_1, \dots, z_{\lceil \frac{n}{2} \rceil}))$.



Divide & Conquer

“Divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide-and-conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.”



John von Neumann

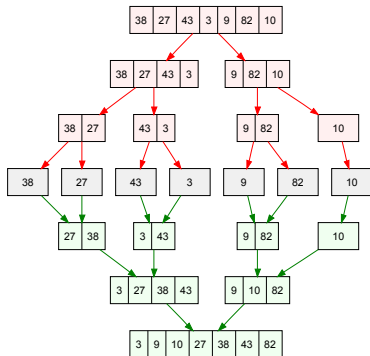
MergeSort $(x_1, \dots, x_n)$, John von Neumann, 1945

1. $(y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}) \leftarrow \text{MergeSort}(x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor})$
2. $(z_1, \dots, z_{\lceil \frac{n}{2} \rceil}) \leftarrow \text{MergeSort}(x_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, x_n)$
3. Return Merge $((y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}), (z_1, \dots, z_{\lceil \frac{n}{2} \rceil}))$.

Time complexity (for $n = 2^k$)

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

$$T(1) = 1$$



Solving the recurrence: method 1 (by substitution)

Time complexity (for $n = 2^k$)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(1) = 1$$

$$T(n) = 2 \left(2T\left(\frac{n}{4}\right) + \frac{cn}{2} \right) + cn$$

Solving the recurrence: method 1 (by substitution)

Time complexity (for $n = 2^k$)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2 \left(2T\left(\frac{n}{4}\right) + \frac{cn}{2} \right) + cn \\ &= 4T\left(\frac{n}{4}\right) + 2cn \end{aligned}$$

Solving the recurrence: method 1 (by substitution)

Time complexity (for $n = 2^k$)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2 \left(2T\left(\frac{n}{4}\right) + \frac{cn}{2} \right) + cn \\ &= 4T\left(\frac{n}{4}\right) + 2cn \\ &= 8T\left(\frac{n}{4}\right) + 3cn \end{aligned}$$

Solving the recurrence: method 1 (by substitution)

Time complexity (for $n = 2^k$)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2 \left(2T\left(\frac{n}{4}\right) + \frac{cn}{2} \right) + cn \\ &= 4T\left(\frac{n}{4}\right) + 2cn \\ &= 8T\left(\frac{n}{4}\right) + 3cn \\ &\dots \\ &= 2^i T\left(\frac{n}{2^i}\right) + icn \end{aligned}$$

Solving the recurrence: method 1 (by substitution)

Time complexity (for $n = 2^k$)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2 \left(2T\left(\frac{n}{4}\right) + \frac{cn}{2} \right) + cn \\ &= 4T\left(\frac{n}{4}\right) + 2cn \\ &= 8T\left(\frac{n}{4}\right) + 3cn \\ &\dots \\ &= 2^i T\left(\frac{n}{2^i}\right) + icn \\ &\dots \\ &= 2^k T(1) + kcn \end{aligned}$$

Solving the recurrence: method 1 (by substitution)

Time complexity (for $n = 2^k$)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2 \left(2T\left(\frac{n}{4}\right) + \frac{cn}{2} \right) + cn \\ &= 4T\left(\frac{n}{4}\right) + 2cn \\ &= 8T\left(\frac{n}{4}\right) + 3cn \\ &\dots \\ &= 2^i T\left(\frac{n}{2^i}\right) + icn \\ &\dots \\ &= 2^k T(1) + kcn \\ &= n + cn \log n = \Theta(n \log n) \end{aligned}$$

Solving the recurrence: method 1 (by substitution)

MergeSort ($x_1, \dots, x_n$), John von Neumann, 1945

1. $(y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}) \leftarrow \text{MergeSort}(x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor})$
2. $(z_1, \dots, z_{\lceil \frac{n}{2} \rceil}) \leftarrow \text{MergeSort}(x_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, x_n)$
3. Return Merge $((y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}), (z_1, \dots, z_{\lceil \frac{n}{2} \rceil}))$.

Memory complexity (for $n = 2^k$)

$$R(n) = R\left(\frac{n}{2}\right) + \Theta(n)$$

$$R(1) = 1$$

Solving the recurrence: method 1 (by substitution)

MergeSort ($x_1, \dots, x_n$), John von Neumann, 1945

1. $(y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}) \leftarrow \text{MergeSort}(x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor})$
2. $(z_1, \dots, z_{\lceil \frac{n}{2} \rceil}) \leftarrow \text{MergeSort}(x_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, x_n)$
3. Return Merge $((y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor}), (z_1, \dots, z_{\lceil \frac{n}{2} \rceil}))$.

Memory complexity (for $n = 2^k$)

$$R(n) = R\left(\frac{n}{2}\right) + \Theta(n)$$

$$R(1) = 1$$

$$R(n) = R\left(\frac{n}{2}\right) + \Theta(n)$$

$$R(n) = \Theta(n) + \Theta\left(\frac{n}{2}\right) + \dots + \Theta\left(\frac{n}{2^k}\right) = \Theta(n)$$

# of subprob	size of subprob.	time per subprob	time per level

Multiplication (Karatsuba 1960)

$$X = \boxed{A} \boxed{B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C} \boxed{D} = C \cdot 10^{\frac{n}{2}} + D$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A} \boxed{B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C} \boxed{D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C} \boxed{D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 4T\left(\frac{n}{2}\right) + cn$$

Tree of recursion:

# of subprob	size of subprob.	time per subprob	time per level



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$



Anatolii Alexeievitch Karatsuba

# of subprob	size of subprob.	time per subprob	time per level

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) = \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$\begin{aligned} X \cdot Y &= AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD \\ &= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD \end{aligned}$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) = \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1}$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) = \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right)$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) = \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right) = \Theta\left(n \cdot \left(\frac{3^{\log n}}{2^{\log n}}\right)\right)$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$\begin{aligned} X \cdot Y &= AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD \\ &= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD \end{aligned}$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right) = \Theta\left(n \cdot \left(\frac{3^{\log n}}{2^{\log n}}\right)\right) = 3^{\log n} \\ &= \Theta\left(\left(2^{\log 3}\right)^{\log n}\right) \end{aligned}$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$\begin{aligned} X \cdot Y &= AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD \\ &= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD \end{aligned}$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right) = \Theta\left(n \cdot \left(\frac{3^{\log n}}{2^{\log n}}\right)\right) = 3^{\log n} \\ &= \Theta\left(\left(2^{\log 3}\right)^{\log n}\right) = \Theta\left(2^{\log 3 \log n}\right) \end{aligned}$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$\begin{aligned} X \cdot Y &= AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD \\ &= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD \end{aligned}$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right) = \Theta\left(n \cdot \left(\frac{3^{\log n}}{2^{\log n}}\right)\right) = 3^{\log n} \\ &= \Theta\left(\left(2^{\log 3}\right)^{\log n}\right) = \Theta\left(2^{\log 3 \log n}\right) = \Theta\left(\left(2^{\log n}\right)^{\log 3}\right) \end{aligned}$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right) = \Theta\left(n \cdot \left(\frac{3^{\log n}}{2^{\log n}}\right)\right) = 3^{\log n} \\ &= \Theta\left(\left(2^{\log 3}\right)^{\log n}\right) = \Theta\left(2^{\log 3 \log n}\right) = \Theta\left(\left(2^{\log n}\right)^{\log 3}\right) = \Theta\left(n^{\log 3}\right) \end{aligned}$$



Anatolii Alexeievitch Karatsuba

Multiplication (Karatsuba 1960)

$$X = \boxed{A \mid B} = A \cdot 10^{\frac{n}{2}} + B$$

$$Y = \boxed{C \mid D} = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + (AD + BC) \cdot 10^{\frac{n}{2}} + BD$$

$$= AC \cdot 10^n + ((A + B)(C + D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i cn = cn \cdot \frac{\left(\frac{3}{2}\right)^{\log n + 1} - 1}{\left(\frac{3}{2}\right) - 1} = \Theta\left(n \cdot \left(\frac{3}{2}\right)^{\log n}\right) = \Theta\left(n \cdot \left(\frac{3^{\log n}}{2^{\log n}}\right)\right) = 3^{\log n} \\ &= \Theta\left(\left(2^{\log 3}\right)^{\log n}\right) = \Theta\left(2^{\log 3 \log n}\right) = \Theta\left(\left(2^{\log n}\right)^{\log 3}\right) = \Theta\left(n^{\log 3}\right) = \Theta\left(n^{1.59\dots}\right). \end{aligned}$$



Anatolii Alexeievitch Karatsuba

Recall
○

Universal hashing
○○

Divide & Conquer
○

Merge sort
○○○

Multiplication
○○