Algorithm Theory

Chapter 1
Divide and Conquer

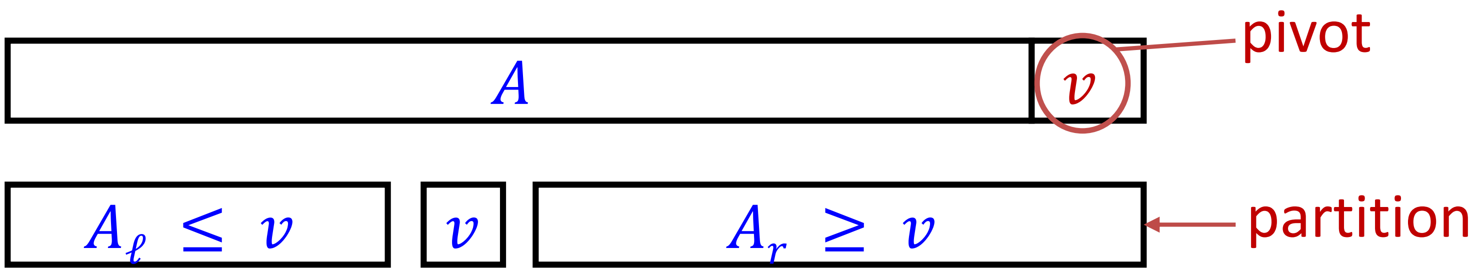
Part I:
Introduction & Running Time Analysis

Fabian Kuhn

Divide-And-Conquer Principle

- Important algorithm design principle
- Examples from basic alg. & data structures class:
  - Sorting: Mergesort, Quicksort
  - Binary search
- Further examples
  - Median
  - Compairing orders
  - Convex hull / Delaunay triangulation / Voronoi diagram
  - Closest pair of points
  - Line intersections
  - Polynomial multiplication / FFT
  - ...

Example 1: Quicksort



function Quick (A: array): array

{returns the sorted array A}

begin

  if $\text{size}(A) \leq 1$ then **return** A

  else { choose pivot element v in A;

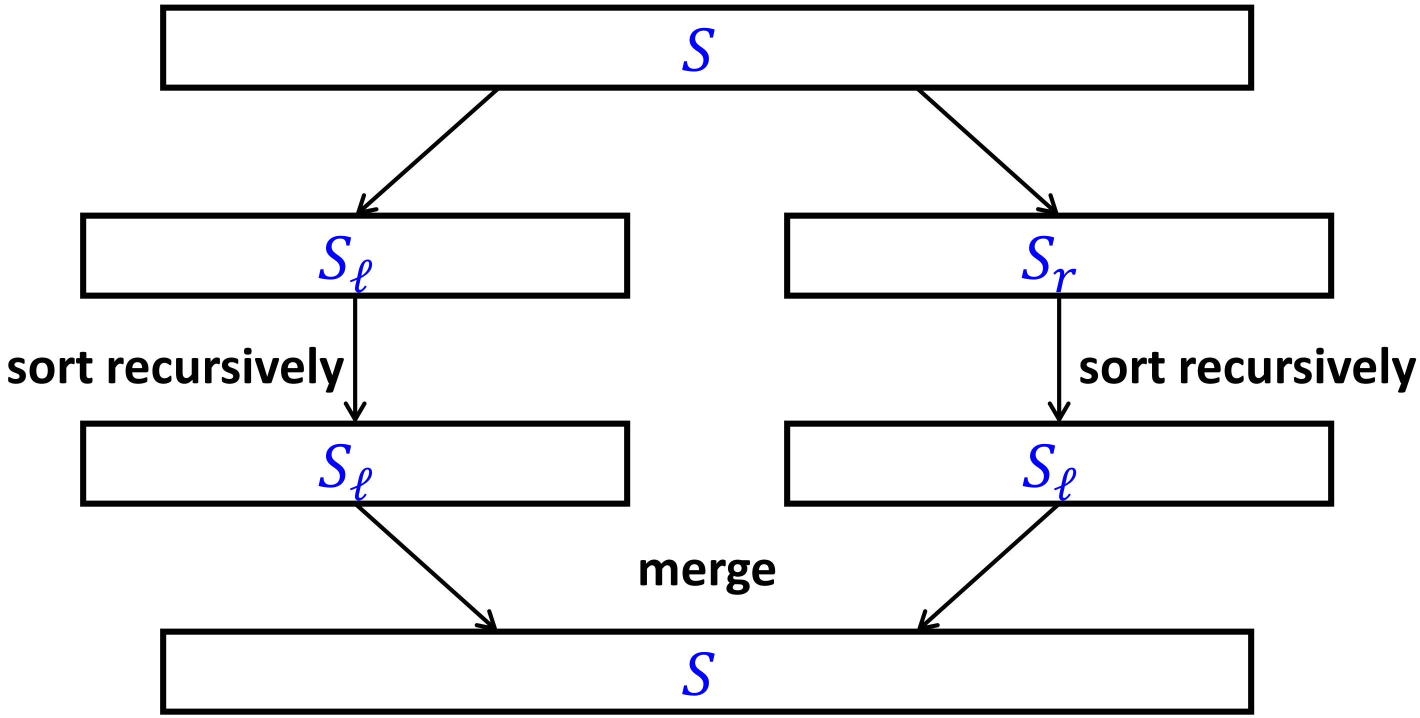
    partition A into A_ℓ with elements $\geq v$,

    and A_r with elements $\geq v$

    return | Quick(A_ℓ) | v | Quick(A_r) |

end;

Example 2: Mergesort



Divide and Conquer: Highlevel Principle

Divide-and-conquer method for solving a problem instance of size n:

1. Divide

$n \leq c$: Solve the problem directly.

$n > c$: Divide the problem into k subproblems of sizes $n_1, \dots, n_k < n$ $(k \geq 2)$.

2. Conquer

Solve the k subproblems in the same way (typically by using recursion).

3. Combine

Combine the partial solutions to generate a solution for the original instance.

QS	MS
choose pivot & partition	divide in middle
recursion	recursion
–	merge sorted halves

Analysis

Recurrence relation:

- $\boldsymbol{T(n)}$: max. number of steps necessary for solving an instance of size n
- $\boldsymbol{T(n) = \begin{cases} c & \textbf{if } n \leq n_0 \\ T(n_1) + \cdots + T(n_k) & \textbf{if } n > n_0 \\ + \textbf{cost for divide and combine} & \end{cases}}$

Special case: $\boldsymbol{k = 2, n_1 = n_2 = {}^{n}/_{2}}$

- cost for divide and combine: $\mathrm{DC}(n)$
- $T(1) = c$
- $T(n) = 2 \cdot T(n/2) + \mathrm{DC}(n)$

Mergesort: $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n)$

Analysis Example: Mergesort

Recurrence relation:

$$T(n) \leq 2 \cdot T(n/2) + cn, \qquad T(1) \leq c$$

Guess the solution by repeated substitution:

$$\begin{aligned}
\boldsymbol{T(n)} &\leq 2 \cdot T\left(\frac{n}{2}\right) + c \cdot n \\
&\leq 2 \cdot \left(2T\left(\frac{n}{4}\right) + c \cdot \frac{n}{2}\right) + c \cdot n \qquad = 4 \cdot T\left(\frac{n}{4}\right) + 2c \cdot n \\
&\leq 4 \cdot \left(2T\left(\frac{n}{8}\right) + c \cdot \frac{n}{4}\right) + 2c \cdot n \qquad = 8 \cdot T\left(\frac{n}{8}\right) + 3c \cdot n \\
&\;\vdots \\
&\leq 2^k \cdot T\left(\frac{n}{2^k}\right) + k \cdot cn \\
&\;\vdots \\
&\leq n \cdot T(1) + (\log_2 n) \cdot cn \quad \boldsymbol{\leq cn \cdot (1 + \log_2 n)}
\end{aligned}$$

Analysis Example: Mergesort

Recurrence relation:

$$T(n) \le 2 \cdot T(n/2) + cn, \qquad T(1) \le c$$

Verify by induction:

Guess: $\boldsymbol{T(n) \le cn \cdot (1 + \log_2 n)}$

– For simplicity, assume that n is a power of 2

Induction Base: $T(1) \le c \cdot 1 \cdot (1 + \log_2 1) = c$

Induction Step:

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

Plug in induction hypothesis for $T(n/2)$.

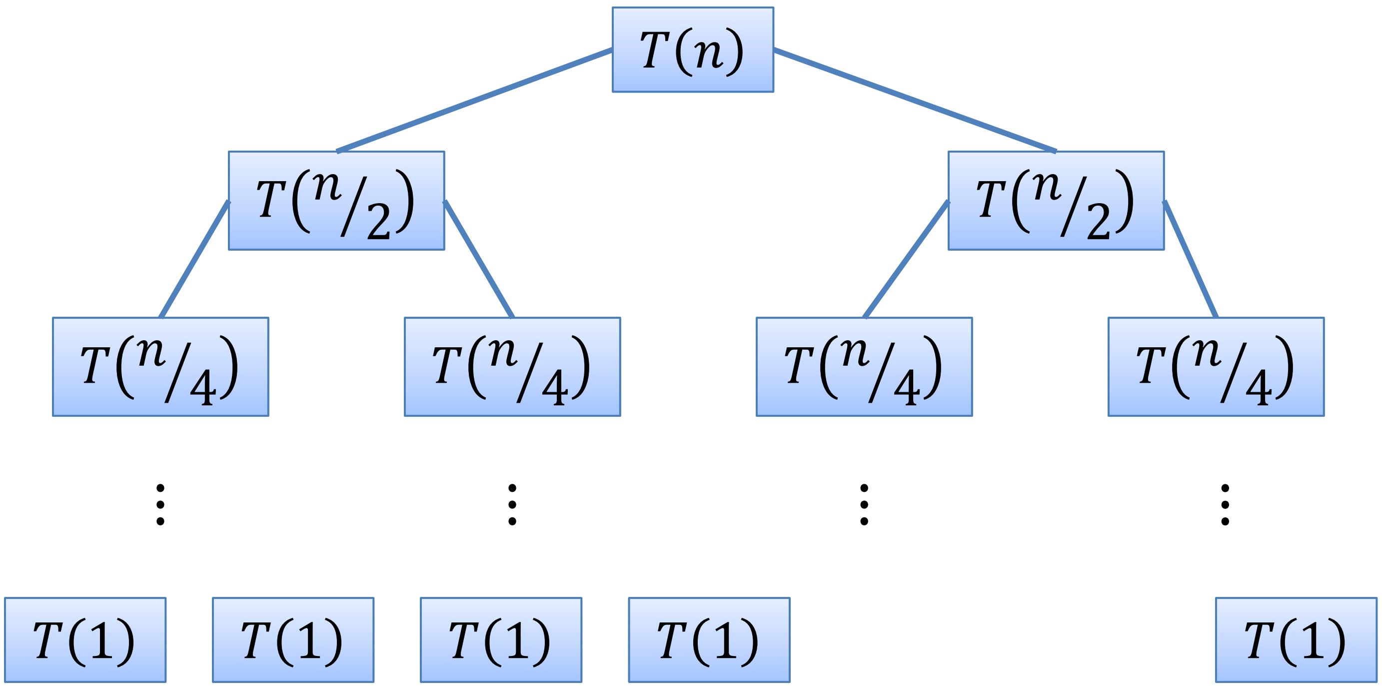
$$T(n) \le \underbrace{2 \cdot \left(c \cdot \frac{n}{2} \cdot \left(1 + \underbrace{\log_2 \frac{n}{2}}_{= \log_2 n}\right)\right)}_{= c \cdot n \cdot \log_2 n} + cn \quad = cn \cdot (1 + \log_2 n)$$

Analysis Example: Mergesort

Recurrence relation:

$$T(n) \leq 2 \cdot T(n/2) + cn, \qquad T(1) \leq c$$

Guess the solution by drawing the recursion tree :



$\boldsymbol{cn}$

$\boldsymbol{2 \cdot c \cdot \frac{n}{2} = cn}$

$\boldsymbol{4 \cdot c \cdot \frac{n}{4} = cn}$

$\boldsymbol{n \cdot T(1) \leq cn}$

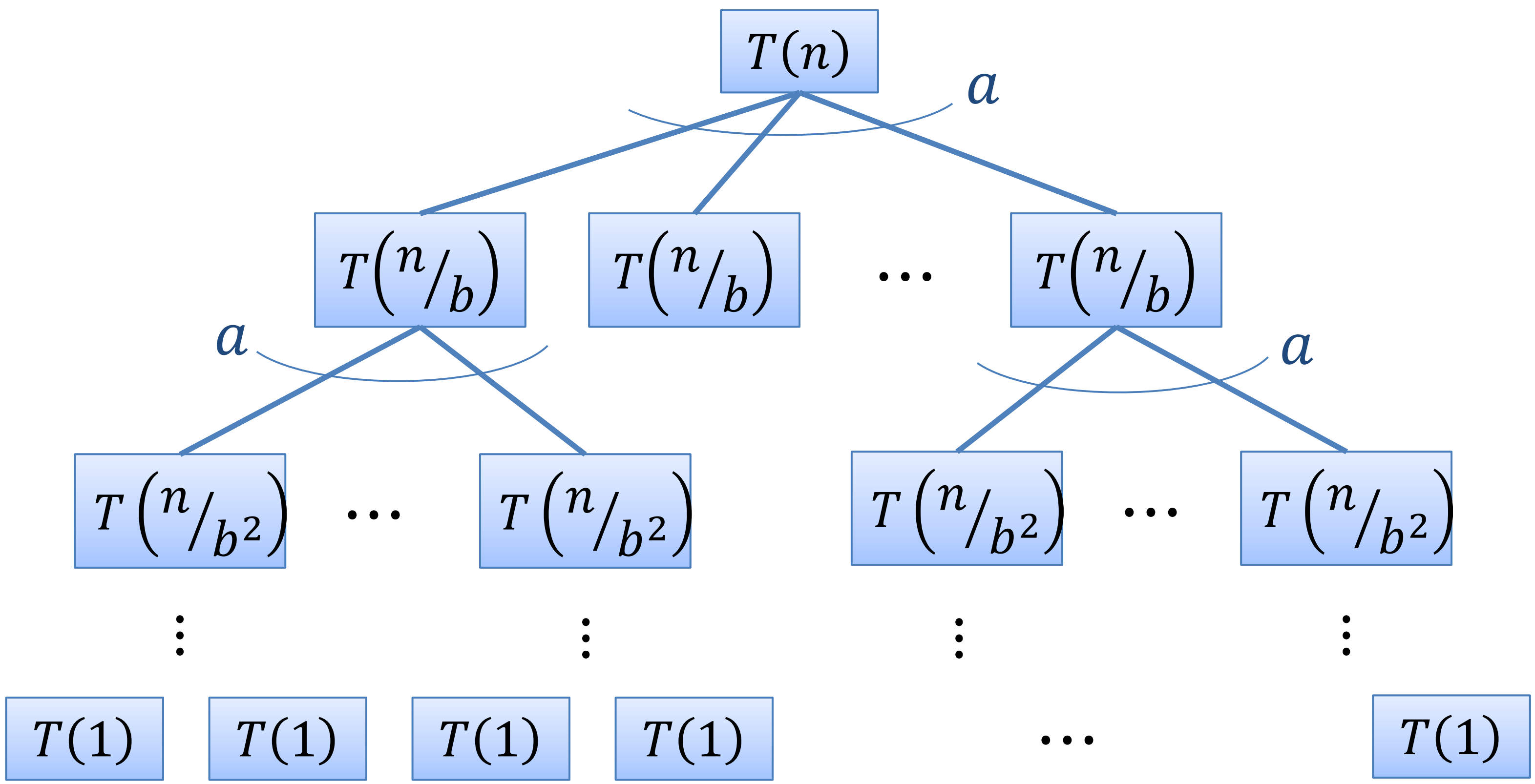
Total time: $(1 + \log_2 n) \cdot cn$

More General Recurrence Relations

Recurrence relation

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^c), \qquad T(n) = O(1) \text{ for } n \le n_0$$

Obtain Intuition by Looking at Recursion:



More General Recurrence Relations

Recurrence relation

$$\boldsymbol{T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^c)}, \qquad \boldsymbol{T(n) = O(1)} \text{ for } \boldsymbol{n \leq n_0}$$

Obtain Intuition by Looking at Recursion:

Rec. Level	Subpr. Size	#Subproblems	Time
1	n	1	$1 \cdot n^c$
2	n/b	a	$a \cdot (n/b)^c = \frac{a}{b^c} \cdot n^c$
3	n/b^2	a^2	$a^2 \cdot \left(n/b^2\right)^c = \left(\frac{a}{b^c}\right)^2 \cdot n^c$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\log_b n$	1	$a^{\log_b n}$	$a^{\log_b n} \cdot 1 = n^{\log_b a}$

More General Recurrence Relations

Recurrence relation

$$\boldsymbol{T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^c), \qquad T(n) = O(1) \text{ for } n \le n_0}$$

Obtain Intuition by Looking at Recursion Tree:

Observations:

- Time grows/shrinks by factor (a/b^c) per level
- If ${}^{a}/_{b^c} < 1$ $(c > \log_b a)$, first level dominates:
  $$T(n) = O(n^c)$$
- If ${}^{a}/_{b^c} > 1$ $(c < \log_b a)$, last level dominates:
  $$T(n) = O\left(n^{\log_b a}\right)$$
- If ${}^{a}/_{b^c} = 1$ $(c = \log_b a)$, all levels are the same:
  $$T(n) = O(n^c \cdot \log n)$$

$1 \cdot n^c$

$({}^{n}/_{b})^c = \frac{a}{b^c} \cdot n^c$

$)^c = \left(\frac{a}{b^c}\right)^2 \cdot n^c$

$\vdots$

$n \cdot 1 = n^{\log_b a}$

Recurrence Relations: Master Theorem

Recurrence relation

$$\boldsymbol{T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)}, \qquad \boldsymbol{T(n) = O(1)} \text{ for } \boldsymbol{n \leq n_0}$$

Cases

- $f(n) = O(n^c), \; c < \log_b a$

$$\boldsymbol{T(n) = \Theta\left(n^{\log_b a}\right)}$$

- $f(n) = \Omega(n^c), \; c > \log_b a$

$$\boldsymbol{T(n) = \Theta\big(f(n)\big)}$$

- $f(n) = \Theta\left(n^c \cdot \log^k n\right), \; c = \log_b a$

$$\boldsymbol{T(n) = \Theta\left(n^c \cdot \log^{k+1} n\right)}$$