INTRODUCTION TO ALGORITMS

RECITATION:

1 & 1/2 HAO DING — MONDAY, 3-4PM SB 001
h ding @ hawk.iit.edu

2 1/2 DEEKSHA PRABHAKAR — FRIDAY, 11-12
dprabhabakar @ hawk.iit.edu

SYLLABUS

1 1/2 semester Sorting

recurrence

- Asymptotic notation
- Methods for solving recurrence
  - → Decision tree
  - → Substitution method
  - → Master Theorem

SORTING ALGORITHMS → from min to max
- Insertion sort
- Merge sort
- Heap sort
- Quick sort
- Comparison sort

- Coder Statistics
- Binary search trees
- Dynamic programming
  - Rod cutting problem
  - Longest common subsequence problem → Related DNA
- Greedy Algorithm → goes to local/global min/max
  - Activity selection problem
  - Knapsack problem

• Huffman code — Assign shorter code for the most frequent letters.

SHORTEST PATH ALGORITHMS

• DIJKSTRA'S ALGORITHM — DIJKSTRA
• BELLMAN - FORD ALGORITHM

FLOW NETWORK

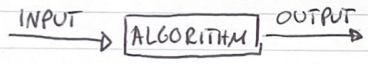
FORD - FULKERSON ALGORITHM

BIPARTITE GRAPHS

LINEAR PROGRAMMING / SIMPLEX METHOD

RECITATION → ENGINEERING CENTER

CLASS 3, 9/1



SORTING ALGORITHMS:

INPUT: Sequence of n numbers $(a_1, a_2, a_3, \dots)$
OUTPUT: Permutation of INPUT $(a_1', a_2', a_3' \dots a_n')$
$(a_1' < a_2' < a_3' \dots \leq a_n')$

1. INSERTION SORTING → The same as playing cards

element 2 → compare with element 1 and if needed, permutate
element 3 → compare with 2 — minor → compare with 1 — minor, ①
mayor ✓ mayor, ok, ②

INPUT TIME:
T(n)

each procedure has its own cost, and the times we think they will be executed.

best case $T(n) = an + b$

Worst case $T(n) = \frac{n(n-1)}{2} = an^2 + bn + c$

ORDER OF GROWTH OF THE RUNNING TIME

~~$\Theta(n^2)$~~ - INSERTION SORT → $\Theta(n^2)$

2. DIVIDE AND CONQUER APPROACH

I. Divide the problem into ~~FEW~~ sub-problems

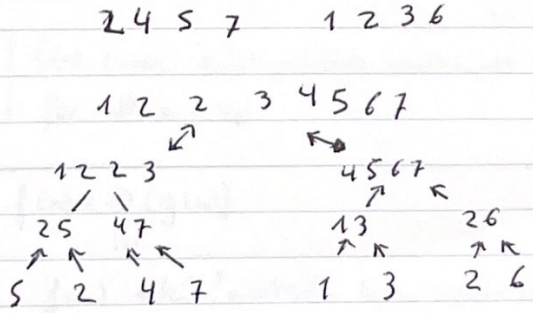
II. Solve the sub-problems by solving them recursively

III. Combine the solutions

MERGE SORT

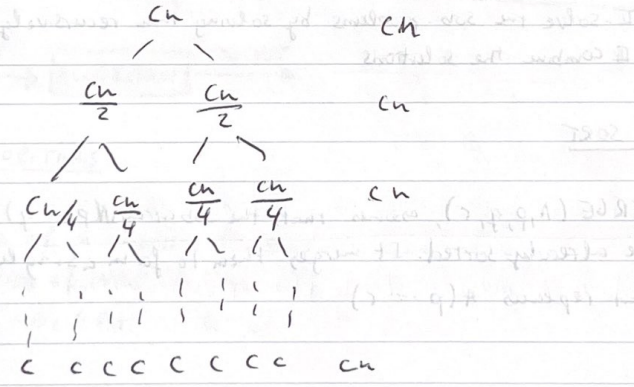
MERGE (A, p, q, r), assumes that the subarrays A(p ... q) and A(q+1 ... r) are already sorted. It merges them to form a single sorted subarray that replaces A(p ... r)

TEAM → 4 MEMBERS



$$T(n) \begin{cases} \Theta(1) \\ a\,T\left(\frac{n}{b}\right) + D(n) + C(n) \end{cases}$$

$$T(n) \begin{cases} \Theta(1) & \text{IF } n = 1 \\ 2T\left(\frac{n}{2}\right) + D(n) & n > 1 \end{cases}$$



Total: $cn \log_2 n + cn$

Asympto

How

tu

For a g

The set

$\Theta(g(n))$

Show th

$\frac{1}{2}$

$c_1 n^2$

$O(g(n))$

A given

$O($

Asymptotic efficiency of algorithms

How does the running time of the algorithm increase with the size of the input? $n \to \infty$

For a given function $g(n)$ we denote by $\Theta(g(n))$ → Tight bound (middle)

The set of functions:

$$\Theta(g(n)) = \{ f(n) : \text{There exists positive constraints } c_1, c_2 \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for All } n \ge n_0 \}$$

Show that

$$\frac{1}{2} n^2 - 3n = \Theta(n^2)$$

$$c_1 n^2 \le \frac{1}{2} n^2 - 3n \le c_2 n^2$$

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2 \quad \begin{cases} c_2 = \frac{1}{2} \\ c_1 \end{cases}$$

- $O(g(n))$ → Upper bound

A given function $g(n)$ we denote by $O(g(n))$ the set of functions:

$$O(g(n)) = \{ f(n) : \text{there exist positive constraints } C \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0$$

$f(n) = O(g(n))$

$f(n)$ está "acotada superiormente" por $cg(n)$

$\Omega(g(n))$ → Lower bound

For a given function $g(n)$ we denote by $\Omega(g(n))$

the set of constraints:

$$\Omega(g(n)) = \{ f(n) : \text{there exists positive constraints } c \text{ and } 0 \text{ such that } 0 \leq c\, g(n) \leq f(n) \text{ for all } n \geq n_0 \}$$

$$f(n) = \Omega(g(n))$$

EXAMPLE:

$f(n) = 2n \log n$ $\quad n \to \infty$

$g(n) = n^2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2n \log n}{n^2} = \lim_{n\to\infty} \frac{2 \log n}{n} \overset{\text{L'HOPITAL}}{=} \lim_{n\to\infty} \frac{2 \cdot c \cdot \frac{1}{x}}{1} = 0$$

($\frac{\partial \log n}{\partial n}$)

$$\Rightarrow 2n \log(n) = O(n^2)$$

- RECURRENCE

Recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

$n! = n(n-1)!$

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

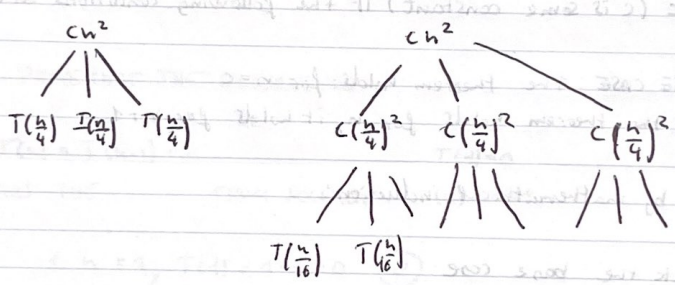
$T(n) = T(n-1) + c$

3 METHODS FOR SOLVING RECURRENCE
- RECURSION TREE
- SUBSTITUTION METHOD
- MASTER THEOREM.

A / RECURSION TREE

$T(n) = 3T\left(\frac{n}{4}\right) + cn^2$

$T(n) =$



II

EJEMPLO:

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

goes down slower

~~$k = \log \left(\frac{2}{3}\right)^k \cdot n = 1$~~

$k = \log_{\frac{3}{2}} n$, $T(n) \leq (\log_{\frac{3}{2}} n + 1)\, cn$

$T(n) \leq cn \log_{\frac{3}{2}} n + cn$

$T(n) \leq cn \frac{\log_2 n}{\log_2\left(\frac{3}{2}\right)} + cn$

$T(n) = O(n \log_2 n)$

B/ SUBSTITUTION METHOD

1. Guess the form of the solution (intuition), "Idea feliz"
2. Use the mathematical induction to show that the solution works.

Mathematical induction:

Mathematical induction states that a theorem is true for any value of $n \geq c$ (c is some constant) IF the following conditions are ~~true~~ TRUE

1. BASE CASE: The theorem holds for $n = c$
2. If the theorem holds for n, it holds for $n+1$

The proof by mathematical induction:

1. check the base case
2. State the induction hypothesis for $n-1$
3. Use the assumption from the induction hypothesis for $n-1$ to show that the result is true for n

Ex 1

$$\sum_{i=1}^{n} = S(n) = \frac{n(n+1)}{2}$$

A/

$n = 1$

$$S(1) = \frac{1(1+1)}{2} = 1 \quad (\checkmark)$$

B/

$$S(n-1) = \frac{(n-1)(n-1+1)}{2} = \frac{(n-1)n}{2}$$

$$S(n) = S(n-1) + n = \frac{(n-1)n+n}{2} = \frac{n(n+1)}{2}$$

EX 2. PROVE THAT THE SUM OF THE FIRST n POSITIVE ODD NUMBERS IS n^2

1. $1 = 1^2$
2. $\sum_{i=1}^{n-1} (2i-1) = (n-1)^2$
3. $\sum_{i=1}^{n} (2i-1) = \sum_{i=1}^{n-1} (2i-1) + 2n - 1 = (n-1)^2 + 2n - 1 = n^2 - 2n + 1 + 2n - 1 = \boxed{n^2}$

ya asumimos el teorema para $n-1$

demostramos que vale para n

EX. 3. PROVE THAT THE RECURRENCE

$T(n) = T(n-1) + 1$ $\quad T(1) = 0$

HAS THE FROM SOLUTION $T(n) = n - 1$

1. $n = 1$, $T(1) = 1 - 1 = 0$ (✓)
2. THE INDUCTION HYPOTHESIS: $T(n-1) = (n-1) - 1 = n - 2$
3. $T(n) = T(n-1) + 1 = n - 2 + 1 = n - 1$

C. SUBSTITUTION METHOD

We substitute the guessed solution for the function ____ applying the inductive hypothesis. Hence the name substitution method

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

assume $T(n) = O(n \log n)$ → GUESS

$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \leq c \left\lfloor \frac{n}{2} \right\rfloor \log\left(\frac{n}{2}\right)$

$2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \leq 2c \left\lfloor \frac{n}{2} \right\rfloor \log\left(\frac{n}{2}\right) + n$

$T(n) \leq 2c\left[\frac{n}{2}\right] \log \left[\frac{n}{2}\right] + n$

$T(n) \leq cn \log \frac{n}{2} + n$

$T(n) \leq cn(\log n - 1) + n$

$T(n) \leq cn \log n - cn + n$

$T(n) \leq cn \log n - n(c-1)$

$T(n) \leq c \log n$ ¿when? → when $c \geq 1$

$T(n) = 2T([\frac{n}{2}]) + n$

$T(n) = O(n \log n)$

$T(n) \leq cn \log n$

$T(1) \leq c \cdot 0$

given n_0

$T(n_0) \leq cn_0 \log n_0$

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + 1$

$\underline{T(n) = O(n)}$

$T(n) = 3T\left[\frac{n}{4}\right] + \Theta(n^2)$

$\rightarrow T(n) = O(n^2)$

$T(n) \leq d n^2$

$T\left(\frac{n}{4}\right) \leq d \left(\frac{n}{4}\right)^2$

~~$3T\left(\frac{n}{4}\right) \leq 3d\left(\frac{n}{4}\right)^2$~~ $\quad 3T\left(\frac{n}{4}\right) + cn^2 \leq 3d\left(\frac{n}{4}\right)^2 + cn^2$

$T(n) \leq \frac{3d}{16} n^2 + cn^2$

$T(n) \leq \left(\frac{3d}{16} + c\right) n^2$

$\frac{3d}{16} + c \leq d \Rightarrow c \leq \frac{13}{16} d \Rightarrow \frac{16}{13} \leq d$

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

$T(n) = O(n \log n)$

$T(n) \leq dn \log n$

$\in A$

$3x + 7 = 3 \quad ? \; x = 3?$

$x = 3$ $\quad 3 = 3$

THE MASTER THEOREM

The recurrence $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, $a \geq 1$ $b > 1$ can be solved as follows

I) IF $af\left(\frac{n}{b}\right) = c$ for some constant $c < 1$

Then, $T(n) = \Theta(f(n))$

II) IF $af\left(\frac{n}{b}\right) = cf(n)$ for some constant $c > 1$, Then

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

III) IF $af\left(\frac{n}{b}\right) = f(n)$ "$c = 1$", Then $T(n) = \Theta(f(n)\log_b n)$

Proof:

$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \dots + a^L f\left(\frac{n}{b^L}\right)$$

$$af\left(\frac{n}{b}\right) = c \cdot f(n)$$

$$af\left(\frac{n}{b^2}\right) = cf\left(\frac{n}{b}\right)$$

$$a^2 f\left(\frac{n}{b^2}\right) = acf\left(\frac{n}{b}\right) = ac\, f(n) = c^2 f(n)$$

$$a\, f\left(\frac{n}{b}\right)$$

$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad a \geq 1,\ b > 1$

$a\, f\left(\frac{n}{b}\right) = c\, f(n)$

1. $c < 1 \Rightarrow T(n) = \Theta(f(n))$
2. $c > 1 \Rightarrow T(n) = \Theta\left(n^{\log_b(a)}\right)$
3. $c = 1 \Rightarrow T(n) = \Theta\left(f(n) \log_b(n)\right)$

EX 1 $T(n) = T\left(\frac{3n}{4}\right) + n$ solve it by applying the master theorem

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a \cdot f\left(\frac{n}{b}\right) = 1 \cdot \frac{3n}{4} = \frac{3n}{4} \rightarrow c = \frac{3}{4} < 1$

$a = 1$

$b = 4/3$

$f(n) = n$

$T(n) = \Theta(n)$

EX 2: $T(n) = 3T\left(\frac{n}{2}\right) + n \longrightarrow a = 3,\ b = 2,\ f(n) = n$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a\, f\left(\frac{n}{b}\right) = c\, f(n)$

$3 \cdot \frac{n}{2} = c \cdot n \Rightarrow c = \frac{3}{2} \Rightarrow c > 1$

$T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 3}\right)$

$T(n) = 2T\left(\frac{n}{2}\right) + n$ ⟹ (MERGE-SORT)

$a = 2$

$b = 2$ $\qquad a f\left(\frac{n}{b}\right) = c f(n)$

$f(n) = n$ $\qquad 2\frac{n}{2} = n$

$n = cn \Rightarrow c = 1$

$T(n) = n \log_2(n)$

$T(n) = \Theta(f(n) \log_b(n))$

EITHER OF THEM NORMALLY, 1/3

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

$a = 4$

$b = 2$

$f(n) = n \log(n)$

$a f\left(\frac{n}{b}\right) = 4\frac{n}{2} \log\frac{n}{2} = 2n \log\frac{n}{2} = 2n(\log n - 1) = 2n \log n - 2n$

$a f\left(\frac{n}{b}\right) = c f(n)$

↳ $c > 1$

$\Theta \circ \Theta$

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

$T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$

$-0{,}1\, n \log n < -2n$

$\forall \log n > \epsilon + 20\%$

$\log n \geq 20$

MASTER THEOREM → (CONT), OFFICIAL THEOREM

$a \geq 1$, $b > 1$, $f(n)$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

EITHER OF THEM NORMALLY 1/3

I. if $f(n) = O\left(n^{\log_b(a) - \varepsilon}\right)$ FOR SOME CONSTANT $\varepsilon > 0$, THEN $T(n) = \Theta\left(n^{\log_b a}\right)$

II if $f(n) = \Theta\left(n^{\log_b a}\right)$ THEN $T(n) = \Theta\left(n^{\log_b a} \log_b n\right)$

III if $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ for some constant $\varepsilon > 0$ and if $a f\left(\frac{n}{b}\right) \leq c f(n)$ for some constant $c < 1$ for all sufficiently large n, Then $T(n) = \Theta(f(n))$

EXAMPLE

$T(n) = 9T\left(\frac{n}{3}\right) + n$ $\begin{cases} a = 9 \\ b = 3 \\ f(n) = n \end{cases}$

$n^{\log_3 9} = n^2 \Rightarrow T(n) = \Theta(n^2)$

"nine" ↑ I. $\forall\ \varepsilon < 1$

EX. 2

$T(n) = T\left(\frac{2n}{3}\right) + 1$ $\quad a = 1$ $b = \frac{3}{2}$ $f(n) = 1$

$n^{\log_{3/2}(1)} = 1$

↓ (II)

$T(n) = \Theta(\log(n))$

Ex 3:

$T(n) = 3T(\frac{n}{4}) + n\log(n)$ $\quad a = 3 \quad b = 4, f(n) = n\log(n)$

III a) $n^{\log_b a} = n^{\log_4 3} \approx n^{0.793}$ for example with $\epsilon = 0.2$

III b) $3\frac{n}{4}\log\frac{n}{4} = \frac{3n}{4}\left(\log n - 2\right) = \frac{3n}{4}\log(n) - \frac{3n}{2}$

$\frac{3n}{4}\log(n) - \frac{3n}{2} \leq \frac{3}{4} n \log n$

$T(n) = \Theta(n\log(n))$

- SORTING ALGORITHM

MAX-HEAP

The MAX-HEAP PROPERTY is that for every node i other than the root $A[Parent(i)] \geq A(i)$

Height of a node in a heap is the number of ~~ages (generations)~~ edges on the longest simple downward path from the node to a leaf (of the leaf height = 0)

Ex_4 $T(n) = T(\frac{2n}{3}) + \Theta(1)$

MASTER THEOREM (TEXTBOOK VERSION)

case 2: $f(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^{\log_b a}\log n)$

MASTER THEOREM (LECTURE VERSION)

$af(\frac{n}{b}) = cf(n)$

$f(\frac{n}{b})$

$k = ck$

$c = 1$

$T(n) = \Theta(f(n)\log_b n)$

$T(n) = \Theta(\log n)$

THEOREM:

AN n ELEMENT HEAP HAS ~~HEIGHT~~ HEIGHT $\lceil \log n \rceil$ AND AT MOST $\left\lceil \frac{n}{2^{h+1}} \right\rceil$ NODES

(ceiling)

OF ANY HEIGHT h

For example ($n=10$, $h=1$) $\frac{10}{2^{1+1}} = \left\lceil \frac{10}{4} \right\rceil = 3$

$$\sum_{h=0}^{\log n} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = \sum \left[\frac{n}{2^{h+1}} + O(1) \right] O(h)$$

$$O\left(n \sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h} \right)$$

$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \sum h \left(\frac{1}{2}\right)^h$$

$$\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2}$$

ASSIGNMENT → Resolver la ~~e~~función recursiva

- QUICKSORT

- ASSIGNMENT;

$T(1) = 4$

$$T(n+1) = T(n)\left(\frac{n^2+4n}{n^2+4n+3}\right)^{(n+4)n}\frac{n+4}{n+1} = T(n-1)\left(\frac{(n+4)^n}{(n+3)^n}\cdot\frac{n^n\cdot(n+4)}{(n+1)^{n+1}}\right)\left(\frac{(n+4)^{n+1}\,n^n}{(n+3)^n(n+1)^{n+1}}\right)$$

$$= T(n-1)\,\frac{(n-1+4)^{n-1}(n-1)^{n-1}}{(n-1+3)^{n-1}(n-1+1)^{n-1}}\,\frac{(n-1+4)}{(n-1+1)}\,\frac{(n+4)^n(n)^n}{(n+3)^n(n+1)^n}\,\frac{(n+4)}{(n+1)}$$

$$= T(n-1)\,\frac{(n+3)^{n-1}(n-1)^{n-1}}{(n+2)^{n-1}(n)^{n-1}}\,\frac{(n+3)}{n}\,\frac{(n+4)^n\,n^n}{(n+3)^n(n+1)^n}\,\frac{(n+4)}{(n+1)}$$

$$= T(n-1)\,\frac{(n+4)^n(n-1)^{n-1}}{(n+1)^n(n+2)^{n-1}}\,\frac{(n+4)}{n+1}$$

$$T(n+1) = \frac{(n+4)^{n+1}\cdot(n+3)^n\cdot(n+2)^{n-1}\cdot(\dots)}{(n+3)^n\cdot(n+2)^{n-1}\cdot(n+1)^{n-1}\cdot(\dots)}\cdot\frac{n^n\,(n-1)^{n-1}\,(n-2)^{n-2}\cdot(\dots)}{(n+1)^n\,(n)^{n-1}\,(n-1)^{n-1}\cdot(\dots)}$$

$$= 4\cdot\frac{4\cdot 4}{2}\cdot\left(\frac{n+4}{n+1}\right)^{n+1}$$

19,54

$$\lim_{n\to\infty}\left(\frac{n+4}{n+1}\right)^{n+1}$$

$$\lim_{n\to\infty} \frac{(n+4)^{n+1}}{(n+1)^{n+1}} \overset{\text{L'HOPITAL}}{=} \frac{1\cdot(n+4)^{n+1}\cdot\ln(n+4)}{1\cdot(n+1)^{n+1}\ln(n+1)}$$

1^∞: $\lim_{x\to\infty}\left(\frac{(n+4)}{(n+1)}\right)^{n+1} = e^{\lim_{x\to\infty}\left(\frac{n+4}{n+1}-1\right)\cdot n+1} = e^{\lim_{x\to\infty}\frac{n+4-n-1}{n+1}(n+1)} = e^{\lim_{n\to\infty} 3}$

$\left(\frac{n+3}{n}-1\right)(n$

$T(100) = \left(\frac{103}{100}\right)^{99} = 18{,}65$

$T(n) - c = \left(\frac{n+3}{n}\right)^{n-1} - \frac{cn^{n-1}}{n^{n-1}} = \frac{(n+3)^{n-1} - cn^{n-1}}{n^{n-1}} = \frac{((1-c)n+3)^{n-1}}{n^{n-1}}$

if $c > 4$, $T(n) - c < 0$ $\forall n \geq 1$ (upper bound)

if $c < 1$, $T(n) - c > 0$ $\forall n \geq 1$ (lower bound)

$n \geq 10$, $c_u > \frac{13}{10}$, $T(n) - c < 0$

$c_L < \frac{13}{10}$

$T(n) - T(n-1) = T(n+1) - T(n) = \left(\frac{n+4}{n+1}\right)^{n+1} - \left(\frac{n+3}{n}\right)^n$

$= n(n+4)$

$$\left(\frac{n+3}{n}\right)^n = \left(1+\frac{3}{n}\right)^n$$

$$\left(1+\frac{3}{n}\right)^n < \left(1+\frac{3}{n+1}\right)^{n+1}$$

~~T(n)~~

CLASS ALGORITHMS

30%

- Cheatsheet is allowed
- Multiple-choice questions and fill in the blanks
- Bring calculator

Quicksort: WORST CASE

$T(n) = T(n-1) + T(0) + \Theta(n)$

↑ worst case

$T(n) = T(n-1) + \Theta(n)$

$T(n-1) = T(n-2) + \Theta(n-1)$

$T(n) = \Theta(n^2)$

BEST CASE

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

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

$f(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n \log n)$

CASE SPLIT 9:1

$\Theta(n) + \Theta(n-1) = \Theta(n)$

Random index — worst case

$T(n) = \max\left[T(q) + T(n-q-1)\right] + \Theta(n)$

$0 \le q \le n-1$

$T(n) \le cn^2$

$T(n) \le \max\left[cq^2 + c(n-q-1)^2\right] + \Theta(n)$

$0 \le q \le n-1$

$c \max\left[q^2 + (n-q-1)^2\right] + \Theta(n)$

↓ we look for the maximum

$b - 4ac = 4(1-n)^2 - 4 \cdot 2(n-1)^2 = -4(n-1)^2 \qquad b^2 - 4ac < 0$

$q^2 + (n-q-1)^2 \le (n-1)^2$

$T(n) \le cn^2 - c(2n-1) + \Theta(n)$

$T(n) \le cn^2 - c\left[c(2n-1) - \Theta(n)\right]$

$T(n) \le cn^2$

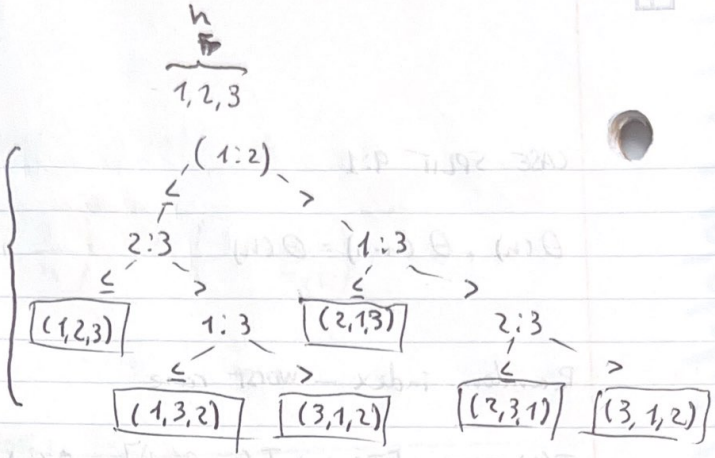
COMPARISON SORT

$n! \leq 2^h$

$h \geq \log_2(n!)$

$T(n) = \Omega(n)$

$T(n) = \Omega(n \log n)$



HEAPSORT AND MERGESORT $O(n \log n)$ ⇒ HEAP SORT AND MERGE SORT ARE ASYMPTOTICLY OPTIMAL COMPARISON SORTS

LINEAR SORTING: COUNTING SORT

N

$K - h$ INPUT ELEMENTS ARE IN THE RANGE FROM 0 TO K

$\theta(n)$ $\begin{cases} \theta(K+n), \text{ normally } K \wedge n \\ K = O(n) \end{cases}$

Property: it is stable Counting sort is stable

"numbers with the same value appear in the output array in the same order as they do in the input array"

"whichever number appears first in the input array, appears first in the input array"

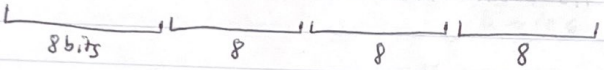
- RADIX SORT: ORDER THE NUMBERS FROM THE LEAST SIGNIFICANT DIGIT FIRST

RADIX-SORT $\Theta(d(n+k))$

b-BIT numbers

Digits of ~~nbits~~ $b = 32$

$r = 8$



100 87 63 11

- BUCKET SORT

PDF

PROBABILITY DENSITY FUNCTION

A

1 .78
2 .17
3 .39
4 .26
5
6
7
8
9
10

AVERAGE CASE → $\Theta(n)$

LOOKING FOR MAXIMUM → $n-1$ comparisons

MINIMUM → $n-1$

MINIMUM + MAXIMUM → $3\left(\frac{n}{2}\right)$

OBJECTIVE: PRINT OUT ALL THE KEYS IN A BINARY SEARCH TREE IN SORTED ORDER

IN ORDER TREE WALK PRINTS THE KEY OF THE ROOT OF A SUBTREE BETWEEN PRINTING THE VALUES IN ITS LEFT SUBTREE AND PRINTING THOSE IN ITS RIGHT SUBTREE

$T(n) \leq (c+d)\,k + c + (c+d)(n-k-1) + c + d$

$T(n) \leq (c+d)\,n + c - (c+d) + c + d$

$T(n) = \begin{matrix} \Omega(n) \\ O(n) \end{matrix} \Big\} \; \Theta(n) = T(n)$

SUCCESSOR IN THE SORTED ORDER DETERMINED BY AN IN ORDER TREE WALK

The successor of a node x is the node with the smallest key greater than x key

IF The right subtree of a node x is nonempty then the successor of x is the left most node in x's right subtree

IF The right subtree of a node x is empty and x has a successor, then y is the lowest ancestor of x whose left child is also an ancestor of x.

GNU PLOT → for plotting, very professional

97
95
87
279 / 3 = 93

Exam: calculator + cheetsheet, 1 sheet double-sided

70 ~~questions~~ minutes
33 questions
Don't go back
} laptop

multiple-choice
T/F
Fill in the blank
} no partial credit

most difficult is logarithms

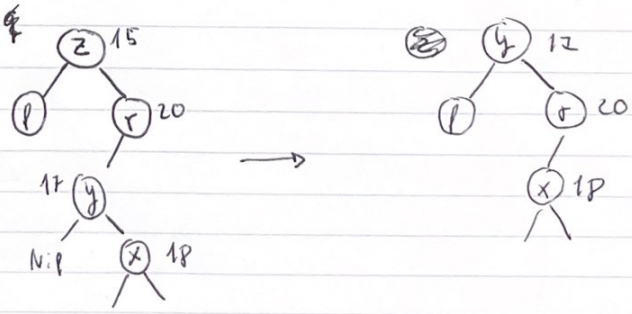
A ~ 80%
B ~ 60%

Exam value 30%
HW 40%

DELETING A NODE z FROM A BINARY SEARCH TREE

1. z Has no children → simply remove it
2. z Has children, but only 1
   - 2.a. z Has no left child, replace z by its right child
   - 2b. z Has no right child, replace z by its left child

3. z Has left and right childs, we find z's succesor y
   - → If y is its right child, then we replace z by y leaving y's right child alone
   - → Otherwise y lies within z's right subtree but it is not z's right child
     We replace y by its own right child and then we replace z with y

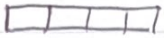


binary search tree is in

ADA1
30% 1st Ex → -10%
30% Final
40% HW → -26%
-36% → 64%

OPTIMIZATION PROCEDURES:

~~1. Dynamic~~ I. DYNAMIC PROGRAMMING



$$r_n = \max_{1 \le i \le n} (p_i + r_{n-i})$$

CUT ROD WITH PARAMETER ~~#~~ n

$$T(n) = 1 + \sum_{j=0}^{n-1} T(j)$$