

Data Structures for Network Problems

David Pisinger
pisinger@man.dtu.dk
DTU Management
Technical University of Denmark

Complexity of Algorithm

A good algorithm should

- be fast
- use little space

Larger instances use more time/space

- n is size of instance (how many bytes)
- measure running time and space as function of n

Independent of computer

- we count number of additions/multiplications/moves
- we do not care about constants

Data structures and complexity

Network problems, analyze how good they are

- Complexity, big-Oh notation, growth of functions
- Simple data structures (find, insert, delete, succ,min)
- Binary search trees (find, insert, delete, succ, min)
- Sorting
- Disjoint sets (does a, b belong to same set?)

Fast walk-through, focus on results, proofs omitted

Complexity of Algorithm

Algorithm 1		
n	time (s)	space (Mb)
10	1	1
20	2	2
30	4	4
40	8	8
50	16	16
60	32	32
70	64	64
80	128	128
90	256	256
100	512	512

$2^{n/10}$ $2^{n/10}$

Algorithm 2		
n	time (s)	space (Mb)
10	110	11
20	120	12
30	130	13
40	140	14
50	150	15
60	160	16
70	170	17
80	180	18
90	190	19
100	200	20

$100 + n$ $10 + n/10$

Algorithm 3		
n	time (s)	space (Mb)
10	23	10000
20	30	10000
30	34	10000
40	37	10000
50	39	10000
60	41	10000
70	42	10000
80	44	10000
90	45	10000
100	46	10000

$10 \log n$ 10000

Complexity of Algorithm

n	10	50	100	300	1000
$5n$	50	250	100	1500	5000
n^2	100	2500	10.000	90.000	10^7
n^3	1000	125.000	10^7	10^8	10^{10}
2^n	1024	10^{16}	10^{31}	10^{91}	10^{302}
$n!$	10^7	10^{65}	10^{161}	10^{623}	∞
n^n	10^{11}	10^{85}	10^{201}	10^{744}	∞

Atoms in the universe: 10^{78} – 10^{82}
Time since big-bang: 13.73 billion years = 10^{18} seconds

Simple data structures

Set of numbers $S = \{1, 2, 3, 5, 6, 7, 8, 9\}$
Operations:

- $S := S \cup \{t\}$ (insert t)
- $S := S \setminus \{t\}$ (delete t)
- find $t \in S$
- find successor t
- find predecessor t
- find $\min(S)$

Complexity of Algorithm

Asymptotic growth of functions
Big-Oh notation

$$f \text{ is } O(g) \Leftrightarrow \exists k, c : f(n) \leq kg(n) \text{ for } n > c$$

Ordering of complexity

$O(1)$	$O(\log n)$	$O(\log^2 n)$	$O(\log^3 n)$	$O(\sqrt{n})$
$O(n)$	$O(n \log n)$	$O(n \log^2 n)$	$O(n \log^3 n)$	$O(n\sqrt{n})$
$O(n^2)$	$O(n^2 \log n)$	$O(n^2 \log^2 n)$	$O(n^2 \log^3 n)$	$O(n^2 \sqrt{n})$
$O(n^3)$	$O(n^4)$	$O(2^n)$	$O(n!)$	$O(n^n)$

Notice

$$O(100n^3) = O(n^3)$$
$$O(2n^4 + 100n^2 + n \log n) = O(n^4)$$

Simple data structures

Table representation of n numbers. Find number 7. find 2

position	1	2	3	4	5	6	7	8
number	7	3	5	6	8	1	9	2

insert number 4

position	1	2	3	4	5	6	7	8	9
number	7	3	5	6	8	1	9	2	4

delete number 3

position	1	2	3	4	5	6	7	8
number	7	5	6	8	1	9	2	4

find	insert	delete	min
$O(n)$	$O(1)$	$O(n)$	$O(n)$

Simple data structures

Sorted table. Find 7

position	1	2	3	4	5	6	7	8
number	1	2	3	5	6	7	8	9

insert number 4

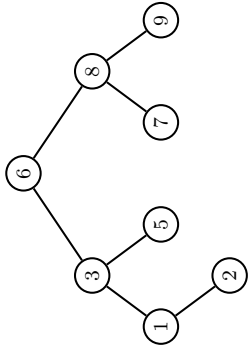
position	1	2	3	4	5	6	7	8	9
number	1	2	3	4	5	6	7	8	9

delete number 3

position	1	2	3	4	5	6	7	8
number	1	2	4	5	6	7	8	9

find	insert	delete	min
$O(\log n)$	$O(n)$	$O(n)$	$O(1)$

Binary tree



find 7
insert 4
delete 3

where h is height of tree

find	insert	delete	min
$O(h)$	$O(h)$	$O(h)$	$O(h)$

Simple data structures

Linked list

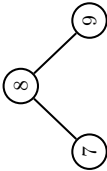
position	1	2	3	4	5	6	7	8
number	7	1	5	8	3	9	2	6
next	5	6	8	2	3	7	x	4



find 8
insert 4
delete 3

find	insert	delete	min
$O(n)$	$O(1)$	$O(1)$	$O(n)$

Binary tree

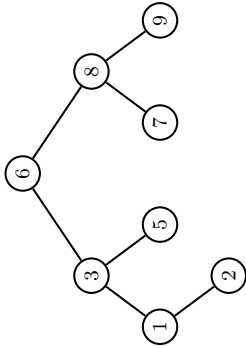


insert 6
insert 5
insert 4
insert 3
insert 2
insert 1
find 1

find	insert	delete	min
$O(n)$	$O(n)$	$O(n)$	$O(n)$

Balanced binary tree

Red-black trees



after each insert/delete make a clean-up in $O(1)$

find	insert	delete	min
$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

Quicksort

Choose a number t , partition in numbers $\leq t$ and $> t$

start:

73568192

pick 6, partition

35612789

pick 6 in first set, pick 8 in second set

35126789

pick 5 in first set

31256789

pick 1 in first set

12356789

Quicksort, bad example

How should we choose number t for partitioning?

start:

86312579

pick 1, partition

18632579

pick 2, partition

12863579

pick 3, partition

12386579

pick 5, partition

12358679

Time: $O(n^2)$

Sorting

Bubble-sort

Repeat: Find smallest number, swap it to current position

7	3	5	6	8	1	9	2
---	---	---	---	---	---	---	---

Algorithm

1 smallest	1	3	5	6	8	7	9	2
2 smallest	1	2	5	6	8	7	9	3
3 smallest	1	2	3	6	8	7	9	5
5 smallest	1	2	3	5	8	7	9	6
6 smallest	1	2	3	5	6	7	9	8
7 smallest	1	2	3	5	6	7	9	8
8 smallest	1	2	3	5	6	7	8	9

Time: $O(n^2)$

Quicksort

- If we choose t as **median of numbers**: Quicksort runs in $O(n \log n)$ time
- Median can be found in $O(n)$ time
- Randomly picking a number from the set is also good

Disjoint set datastructure

Merge: $A := A \cup C$

$$\begin{aligned} A &= \{1, 7, 9, 4, 5, 8, 0\} \\ B &= \{2\} \\ D &= \{3, 6\} \end{aligned}$$

number	0	1	2	3	4	5	6	7	8	9
set		A	A	B	D	A	D	A	A	A

findset	merge
$O(1)$	$O(n)$

Disjoint set datastructure

Set of sets (family of sets) $S_1, S_2, \dots, S_m$
each element belongs to one and only one set
Question: which set does t belong to?

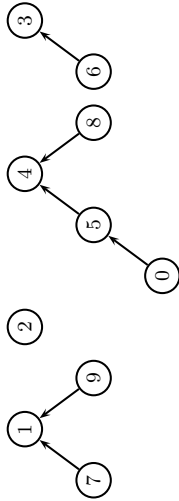
$$\begin{aligned} A &= \{1, 7, 9\} \\ B &= \{2\} \\ C &= \{4, 5, 8, 0\} \\ D &= \{3, 6\} \end{aligned}$$

Which set does 9 belong to?
Are 9, 8 in the same set?
Are 5, 8 in the same set?

number	0	1	2	3	4	5	6	7	8	9
set		C	A	B	D	C	C	D	A	C

Disjoint set datastructure

Make each element point to the first element in the set.
Return this number

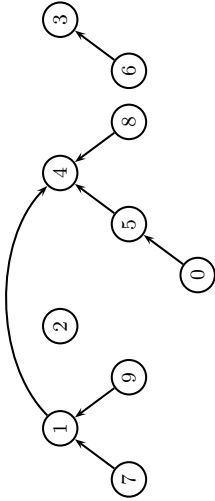


Which set does 0 belong to?
Are 5, 8 in the same set?
Merge: $A := A \cup C$
Are 9, 0 in the same set?

$$\begin{aligned} A &= \{1, 7, 9\} \\ B &= \{2\} \\ C &= \{4, 5, 8, 0\} \\ D &= \{3, 6\} \end{aligned}$$

Disjoint set datastructure

- Each time we merge, merge smallest set into biggest
- Each time we search, move pointers upward



Amortized time complexity (after many operations)

findset	merge
$O(1)$	$O(1)$

Exercises

Exercise 2

- Build a binary tree by inserting the following numbers in the specified order

6, 2, 4, 3, 8, 7, 1, 5, 9
- describe how you find successor(6), successor(5), sucessor(9)
- describe how you find predecessor(5), predecessor(7), predecessor(1)
- describe how delete(2) works

Exercises

Exercise 1 Write in big-Oh notation the asymptotic growth of the following functions

- $4n^3 + n^4 + n^3(\log n)^2$
- $100n + n \log n + \sqrt{n}$
- $5n^7 + 100n^6 + n^8$
- $n\sqrt{n} + n \log n + n \log(n^2)$

Exercises

Exercise 3 Choose the best data structure for a graph algorithm using

- n^2 insert, n^2 delete, n^3 minimum
- n insert, n find, n minimum
- n^2 insert, n^2 find, n minimum
- n insert, n^2 find

You may assume that at any time there is at most $O(n)$ numbers in the data structure.

Exercises

Exercise 4 Consider the following sorting algorithm: Given n numbers to be sorted, insert them in a binary balanced tree. Repeatedly use the minimum operation to extract the smallest number, and use the delete operation to remove the smallest number from the binary tree. What is the running time of this algorithm? What is the advantage/disadvantage of using Quicksort?