Algorithmics I

Introduction and Revision

Dr. Gethin Norman

School of Computing Science
University of Glasgow

gethin.norman@glasgow.ac.uk

Schedule

Lecturer: Gethin Norman

- contact details on moodle
  - office hour Thursday 0900 from next week (message me on Teams)
- you can use Teams or padlet to ask questions

Lectures: Thursdays 1200-1300 (only second hour)

- split into pre-recorded lectures plus live session

Tutorial/Lab sessions: Thursdays 1300-1700

- tutorials will alternate between face to face and online
- information will be made available on moodle/teams

Assessment

- degree exam 75% (December 2021)
- assessed exercise 20% (15 October – 15 November 2021)
- quizzes 5% (after watching videos for the week)

Feedback from a student in 2018-19

"It needs to be made clear to students that if they don't attend lectures and tutorials they're VERY LIKELY to not do well because at first glance it seems like an easy subject so a lot of people kinda disregarded it"

Lectures

Will post all lectures as short 10-20 minute videos on moodle

- the videos will cover all the assessed material (videos are from last year)
- handouts of the lectures will also be available on moodle
- you can then study at your own pace
- post questions on Teams or padlet (if you want to be anonymous)
- I will answer these both offline and discuss at the live session each week
- please also try and take part in the discussions on Teams
- also an office hour Thursday 0900-1000
    - contact me on Teams and we can arrange a call

Live sessions will be used for recap, examples and Q&A

- start by quiz from previous week,
- then recap the material of the week
- then worked examples and Q&A
- will only use the second hour 1100-1200 (sessions will be recorded)

Labs (well really Tutorials)

Based on student feedback these sessions will run with you completing the tutorial sheets in small groups

- with support from myself and lab demonstrators
- these alternate between face to face in the lab and online

Previous year tutorials were run essential as a lecture

- with me going through the solutions as a lecture
- I have the videos from last year's zoom sessions so will also post these videos and slides, as well as solutions, afterwards
- so you get the best of both approaches

Any feedback or potential improvements please let me know

Labs (this time really labs)

The week before the assessed exercise is released week

- a lab with a warm-up exercise for the assessed exercise

For the two weeks before the assessed exercise deadline

- labs for help with the assessed exercise
- you can ask me and the lab assistants for help with you work

Coursework

Assessed exercise 20%

- based on text compression algorithms

Quizzes 5%

- 3 quiz questions per lecture
- best 15 marks for the quizzes will be aggregated to give a band
- affected questions will be voided (email me and do **not submit** good cause claims for quizzes)
- quizzes can only be accessed after watching lecture videos and are closed at the end of the week of corresponding lectures
    - deadline for the quiz each week is Friday at 23:59

Exam 75%

- in December online (and therefore open book)
- revision lecture in week 9
- week 10 free for revision with office hours to answer questions

Outline of course

Section 1: Sorting and Tries

Section 2: Strings and text algorithms

Section 3: Graphs and graph algorithms

Section 4: An introduction to NP completeness

Section 5: A (very) brief introduction to computability

Resources

Lecture videos and handouts on moodle

- algorithm animations and examples from lectures also on moodle

Course texts on moodle (recommended not required)

- M.T. Goodrich & R. Tamassia. **Algorithm Design: Foundations, Analysis, and Internet Examples.** Wiley, 2002
- D. Harel & Y. Feldman. **Algorithmics: the Spirit of Computing.** Addison Wesley, 2004 (also earlier 1992 edition by D. Harel)
- M. Sipser. **Introduction to the Theory of Computation.** Course Technology, 2006

Tutorial exercises (and solutions after tutorial class) on moodle

Moodle course webpage also has/will have links to

- lab and assessed exercise material, past papers and model answers

Background

From JP2

- Java and a Java-style pseudocode is used to describe algorithms
- some algorithms are partially or fully implemented in Java
- the assessed exercise is to be done in Java

From AF2

- fundamental concepts of graphs
- terminology and notation of sets, relations and functions
- basic mathematical reasoning and proof

Background

From ADS2

- concept of an abstract data type (ADT)
- common data structures (e.g. lists, arrays and binary trees)
- basics of algorithm analysis (e.g. time complexity and big-O notation)
- common searching algorithms (e.g. linear search and binary search)
- common sorting algorithms (e.g. insertion sort and merge sort)

Section 0 – Revision

Algorithm Analysis

- big O notation
- the log function
- polynomial and exponential time algorithms

Abstract data structures

- stacks
- queues
- priority queues

Revision – Algorithm analysis

Time complexity as function of input size
- either worst-case or average case

Worst-case analysis is the most commonly specified
- it gives a guarantee of an algorithm's performance
- asymptotic behaviour is the key factor
  - this indicates what will happen as the input size grows
- generally expressed using the 'Big O' notation

Space complexity can be significant for some algorithms

Revision – Polynomial time algorithms

Polynomial-time: $O(n^c)$ for some constant c

Execution time of algorithms with various complexity functions

- times are in seconds (unless otherwise stated)
- the assumption is 10^9 operations per second

complexity function	input size			
	1,000	10,000	100,000	1,000,000
n	10^{-6}	10^{-5}	10^{-4}	10^{-3}
$n \log_2 n$	10^{-5}	1.3×10^{-4}	1.6×10^{-3}	0.02
n^2	0.001	0.1	10	16 mins
n^3	1	16 mins	11 days	32 yrs

Revision – Exponential time algorithms

Exponential-time: no better than $O(c^n)$ for some constant $c>1$

Execution time of algorithms with various complexity functions

- times are in seconds (unless otherwise stated)
- the assumption is 10^9 operations per second

complexity function	input size			
	10	**20**	**30**	**40**
2^n	10^{-6}	0.001	1.1	18 mins
n!	0.004	77 years	★	★

★ – means longer than the age of the earth

$n! > (n/2)^{n/2}$

(we will use this later)

Polynomial-time: potentially useful in practice

Exponential-time: potentially useless in practice

Revision – The stack abstract data type

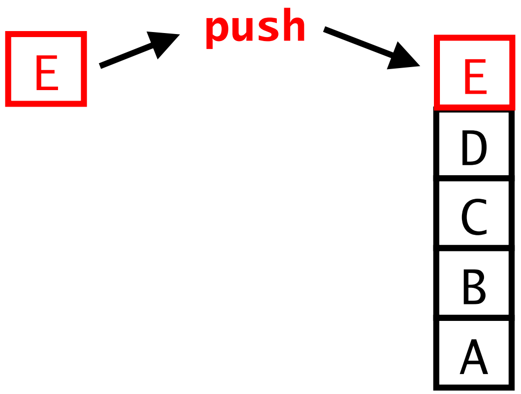
Basic operations are

- `create` (create an empty stack)
- `isEmpty` (check if stack is empty)
- `push` (insert a new item on the top of the stack)
- `pop` (delete and return the item on the top of the stack)

Revision – The stack abstract data type

Basic operations are

- `create` (create an empty stack)
- `isEmpty` (check if stack is empty)
- **push** (insert a new item on the top of the stack)
- **pop** (delete and return the item on the top of the stack)

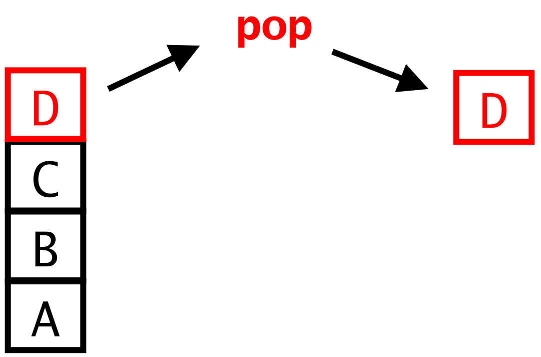


new item inserted on the top of the stack

Revision – The stack abstract data type

Basic operations are

- `create` (create an empty stack)
- `isEmpty` (check if stack is empty)
- **push** (insert a new item on the top of the stack)
- **pop** (delete and return the item on the top of the stack)



items removed from the top of the stack

Revision – The stack abstract data type

Basic operations are

- `create` (create an empty stack)
- `isEmpty` (check if stack is empty)
- **push** (insert a new item on the top of the stack)
- **pop** (delete and return the item on the top of the stack)

Order of removal of elements: last in first out

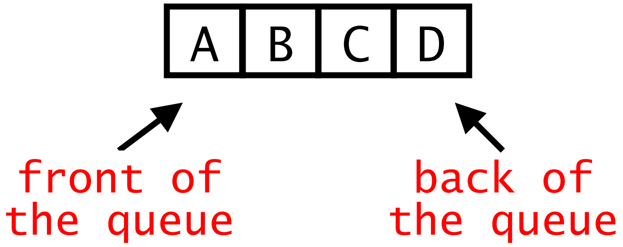
Representations of a stack

- as an array
  - the bottom of the stack is “anchored” to one end of the array
  - all operations are `O(1)`
- as a linked list
  - again all operations are `O(1)`

Revision – The queue abstract data type

Basic operations are

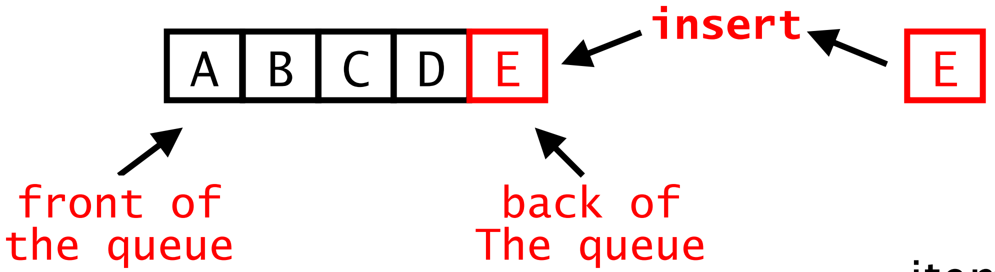
- `create` (create an empty queue)
- `isEmpty` (check if queue is empty)
- `insert` (insert a new item at the back of the queue)
- `delete` (delete and return the item at the front of the queue)



Revision – The queue abstract data type

Basic operations are

- **`create`** (create an empty queue)
- **`isEmpty`** (check if queue is empty)
- **`insert`** (insert a new item at the back of the queue)
- **`delete`** (delete and return the item at the front of the queue)

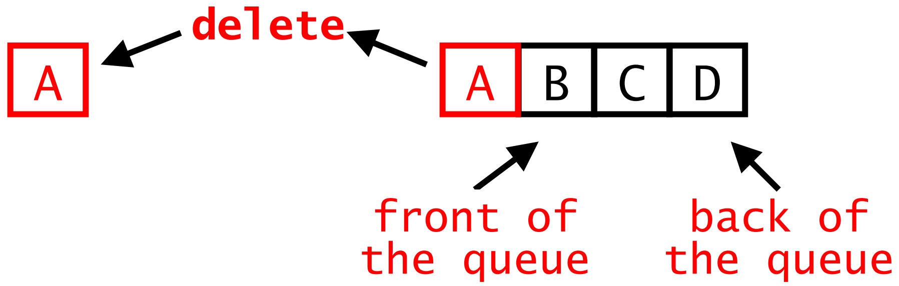


items added to the back of the queue

Revision – The queue abstract data type

Basic operations are

- `create` (create an empty queue)
- `isEmpty` (check if queue is empty)
- `insert` (insert a new item at the back of the queue)
- **`delete`** (delete and return the item at the front of the queue)



items removed from the front of the queue