Algorithms
Past Papers by topic

Analysis of algorithms:

- **2016P1Q8 (a)** sqrt-recurrence relation,
- **2014P1Q9** amortised analysis, aggregate analysis, potential method, multi-pop stack analysis, vector (i.e. array with append)
- **1994P10Q7 (a),(b)** prove that $f(n) = A n^k = O(2^n)$, do all sorting methods take $O(n^5)$ time,
- **1993P3Q8** Determine which algorithm is which based on the running times (informal)

Searching and Sorting:

- **2016P1Q8 (d)** Quicksort last entry vs random pivot (how to trigger quadratic time complexity)
- **2014P1Q7** radix sort (description and correctness), given $k$, find $x$ and $y$ such that $k = x + y.$
- **2014P1Q8 (a)-(c)** Quicksort $k$-th element vs sorting to find $k$-th element, find the $k$ smallest elements, draw 4 distinct BSTs for $\{1, 2, 3, 4\}$.
- **2012P1Q5** describe Quicksort, assuming all partitions are balanced derive recurrence relation, time complexity for when all items are sorted, random pivot
- **2011P1Q6 (a)** Quicksort vs Priority queue for finding $k$-th element
- **2010P1Q5** mergesort example, insertion sort recurrence relation, linked-list mergesort show $O(1)$ space, should convert to linked lists?
- **2008P11Q7** Quicksort implementation, prove termination, worst-case behaviour and how to avoid it, problems of choosing random pivot, find median
- **2007P10Q10** Quicksort, find $k$-th largest elements, minimum-comparisons max, minimum-comparisons second-max
- **2007P1Q4 (a)-(d)** heapsort running time, insertion sort worst-case, shellsort, sorting algorithms lower bound
- **2007P1Q11** Quicksort example, space and time complexity, how to avoid the worst-case behaviour.
- **2006P1Q4** Quicksort, time and space complexity, Quicksort with quadratic search in the end.
- **2006P1Q11** binary search time complexity, comparison-based sorting lower bound, analysis of ternary search, practical comparison between binary and ternary search.
- **2002P4Q4** Shellsort (out of syllabus), radix sort
- **2001P3Q5** Quicksort, counting sort, find median when range of elements is small.
- **2000P3Q5** Quicksort, (worst-case linear-time) select, practical estimate for number of operations.
- **1997P4Q5** Quicksort, risk of randomisation
- **1996P4Q5** Prove sorting lower bound, sorting can take less time, radix sort
- **1995P3Q5 (b)** Sorting an almost sorted sequence
- **1994P4Q7** Three-way quicksort, solve recurrence relation, practical estimation of running time.
- **1994P10Q7 (c)-(e)** true/false questions for sorting algorithms
- **1994P11Q6 (a)** describe an algorithm for finding the median
• 1993P10Q7 lower bound for sorting, comparison between quadratic and \( n \log n \) sort for small values of \( n \), (not clear what is meant by binary insertion).

Priority queues:
• 2016P1Q9 which of the given representations is a valid Fibonacci heap, worst-case (and worst-case amortised) costs for insert, extract-min and decrease-key, analysis of Fibonacci heap variant
• 2008P1Q4 update to the priority queue, heap array implementation, find parent, find children, insert, extract-min
• 2006P1Q12 (b) define the heap property, heapsort relation to quadratic time sorting
• 2011P1Q5 describe decreaseKey(), describe modify, \( d \)-ary heap representation in an array, \( d \)-heapify analyse its complexity
• 2009P1Q6 state properties of min-heap, describe pointer to array-based representation conversion, "sorted array \( \Rightarrow \) represents min-heap", heapify example, extract-min examples, running time of heapsort,
• 2006P6Q1 define heaps, heapify, example, extract min, describe insert
• 2003P3Q3 heapify, heapsort worst-case performance, how many comparisons if sorted and if reverse sorted.
• 1999P4Q5 describe heapsort, prove worst-case time complexity, ternary heaps stored at \( 3i - 1, 3i \) and \( 3i + 1 \).
• 1998P3Q6 (same as 1995P4Q5)
• 1997P3Q5 Find the \( M \) largest values of \( X_i - Y_j \).
• 1995P4Q5 Define priority queues, define heaps, explain how to store a heap in an array, access parent, access offspring, describe insert, delete, heapify, argue that (plain) heapsort is not stable.
• 1994P10Q8 compare three implementations of priority queues: (a) unsorted array, (b) sorted array, (c) binary heap

Divide and Conquer:
• 2000P6Q1 Closest pair of points in Euclidean and Manhattan distance.
• 1994P3Q7 (a) Describe the divide and conquer technique and give examples of problems that can be solved using this.

Binary trees and BSTs:
• 2015P1Q7 which of the following trees is a RBT, largest number of nodes in a RBT, smallest number of nodes in a RBT
• 2014P1Q8 (d) height-balanced BSTs.
• 2012P1Q6 algorithm to check if a sequence is a valid search comparison sequence, check if binary tree is BST, check if binary tree is min-heap, output BST values in sorted order in linear time, why not possible for binary min-heap?
• 2009P1Q5 five variants of RBTs, 2-3-4 trees to RBTs, min/max nodes in RBT, define BST rotation, convert any BST to any other BST using rotations
• 2008P1Q11 define BST rotation, move \( i \)-th largest node to the root (select), recursive delete using select, delete without rotations
• 2007P1Q12 insertions and deletions in 2-3-4 tree, isomorphism between RBT and 2-3-4 trees, diagrams for left/right rotations, use BST rotations to move a node to the root.
• 2006P3Q2 define BST, find predecessor, delete, property of node in a BST
• 1995P3Q5 (c)
• 1994P3Q7 (c) Give examples of balanced data structures and usage in problems.
• 1994P4Q6 describe the operation of RBTs. What is the maximum imbalance in the tree height?
• 1993P3Q7 prove upper and lower bound for number of nodes in strictly binary tree, generalise to trees with n children.

B-Trees:
• 2007P11Q9 def B-trees, insertion example, successor property of B-tree, deletion, deletion example
• 2002P5Q1 insert, update, inorder traversal, estimate number of transfers required.
• 1995P3Q5 (e)

Greedy / Dynamic Programming (DP):
• 2016P1Q8 (c) memoisation for Fibonacci numbers, variants reducing the number of recursive calls,
• 2015P1Q8 define greedy, if there exists a greedy and a DP algorithm which one would you choose?, coin-change problem.
• 2013P1Q6 Describe how to apply dynamic programming to solve the longest palindromic subsequence, bottom up algorithm, worst-case running time, recover all LPS sequences.
• 1994P3Q7 (e),(f) describe the principle of greedy and dynamic programming, give examples of problems solvable using these techniques.

Shortest paths:
• 2017P1Q10 (b),(c) describe Dijkstra’s algorithm, Dijkstra’s with Fibonacci heap, shortest path on DAG, \( \Omega(V \log V) \) for Dijkstra’s implementation.
• 2014P1Q10 All-pairs shortest paths using BF, Dijkstra’s, matrix multiplication and Johnson’s, mod-elling currency-exchange using shortest paths, interpretation of negative cycles
• 2006P5Q1 Describe Dijkstra’s algorithm, prove correctness, why non-negative weights, negative edge weights leaving the source, making edges positive by adding a constant
• 2005P5Q1 Matrix representation of a graph, Floyd-Warshall algorithm, reconstructing the optimal path
• 2001P5Q1 All-pair shortest paths, should check connectedness?, alternative for sparse graph
• 1998P4Q5 Describe and justify Dijkstra’s algorithm for non-negative length paths, Dijkstra with heuristic approximation.
• 1996P3Q5 Describe Dijkstra’s algorithm, extend Dijkstra’s to few negative edges, when is the shortest path well-defined for graphs with negative edges?
• 1995P3Q5 (d)]
• 1994P10Q7 (g),(h)
• 1993P4Q8 recursive formula for paths of length k, transitive closure for adjacency matrix
• 1993P10Q8 all-pairs shortest paths, also retrieving the paths

Minimum spanning trees:
• 2015P1Q9 (c) Find new MST when: (i) increase weight of edge not in MST, decrease weight of edge in MST, add new edge.
• 2006P4Q3 def MST, safe edge, cut, describe an MST algorithm and prove correctness, properties of unique MSTs
• 2003P5Q1 (b),(c) Describe Kruskal’s algorithm, unique weights \( \Rightarrow \) unique MST
Describe Kruskal’s and Prim’s algorithm and give running times.

Describe and justify Kruskal’s algorithm, what would happen if all edges above \( L \) were removed, how to find spanning tree for points on Euclidean plane.

MST consists of \( N - 1 \) smallest edges?

Disjoint set union:

Describe the DSU data structure

Flows and Matchings:

max flow min-cut theorem, example where Ford-Fulkerson takes at least \( k \) steps, one augmentation on a given graph, edge-connectivity.

def matching, prove that no augmenting path means no matching, max flow reduction

Graphs (representations, BFS/DFS):

define total order, describe how to compute in \( O(V + E) \).

adj list vs adj matrix, discovery and finishing time

Define directed graph and SCC, give example, DFS discovery and finish time, linear-time algorithm to find SCC (out of syllabus)

Define directed graph, undirected graph, bipartite graph.

representation of a sparse graph

describe BFS and give examples of tasks solvable using BFS.

Hashing:

define hash tables, hash functions, collisions, define open addressing, probing sequence, advantages and disadvantages of quadratic probing, expression for quadratic probing, find values for quadratic probing.

Fast way to randomly sample an element from a hash table.

Additive hash function example with chaining, with linear probing, implementation of chaining

closed hash-table insert and lookup, evaluate different hash functions, deletion in closed hash-tables

Hash-table implementation

open/closed hashing

Splay trees (out of syllabus):

splay tree faster than linked list?

describe the splay tree operations, when would you use a splay tree instead of a RBT?

structure, insert, delete, state properties,

describe insert, lookup and delete, compare with hash tables

Skiplists (out of syllabus):

Skiplist node, search, when is skiplist preferred to a hash table?

Huffman coding:
• 2005P3Q2 (a)-(c) describe Huffman coding, estimate bits needed

Matrix operations (out of syllabus):
• 2010P1Q6 matrix multiplication, solve Strassen’s algorithm recurrence relation

String algorithms (out of syllabus):
• 2013P1Q5 FSA string matching
• 2005P3Q2 (d),(e) arithmetic coding
• 2004P3Q3 Lempel-Ziv algorithm
• 2004P4Q3 (d) Trie?
• 2003P6Q1 Burrows-Wheeler transform
• 2002P6Q1 Arithmetic encoding
• 2001P4Q5 (b),(c) string matching, estimate time complexity.

Computational geometry (out of syllabus):
• 2004P6Q1 determine if two line segments intersect, Graham scan, heuristic elimination.
• 2001P4Q5 (a) Graham scan
• 1998P3Q5 (same as 1993P11Q8)
• 1996P3Q6 describe Graham scan algorithm, state complexity, heuristic to eliminate points
• 1994P3Q6 (j) Running time of convex hull algorithms
• 1994P10Q7 (j) Extremal points are on the convex hull.
• 1994P10Q7 (f) “All straight lines from the inside of a polygon to the outside intersect the points on the edges forming its boundary an odd number of times”
• 1994P11Q6 (b) determine if two line segments intersect, determine if a half-line intersects a polygon other than at a vertex.
• 1993P11Q8 determine if point in simple polygon, randomised algorithm for convex hull

Randomness:
• 1994P3Q7 (d) Give examples where randomness helps in algorithms.

Other:
• 2002P3Q3 Memory organisation (malloc, etc).