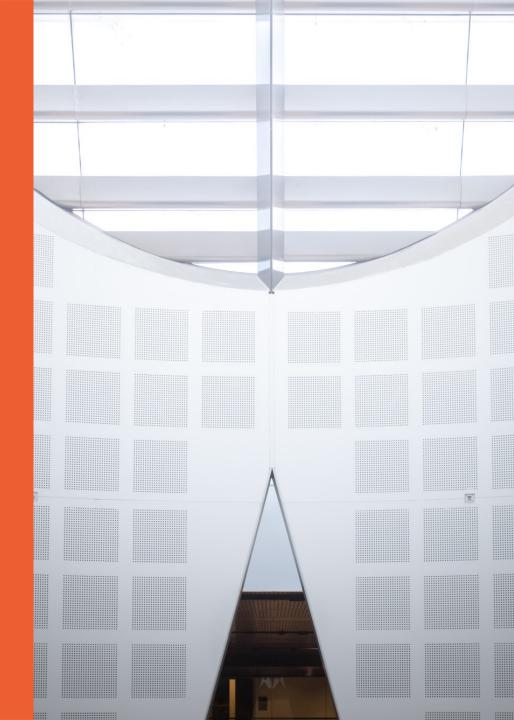
COMP2123

Week 13: Recap and Exam Review

A/Prof Julian Mestre
School of Computer Science





Quick announcements

Fill out online Unit of Study Survey

- https://student-surveys.sydney.edu.au/students/
- Use the free text to help us make this better for next years students. "Pay it forward"

Examples of changes based on past year's feedback:

- wrote the guide on how to approach algorithmic problems
- designed programming exercises

Week 13 Quiz

Quiz 10 will be about the final.

It's available until May 28th

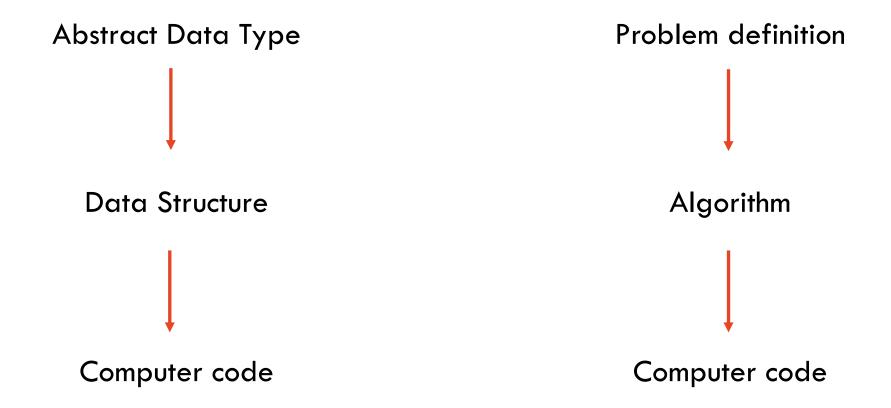
Looking back

We covered a lot of ground!

Lecture Slides

Week 1 - Welcome slide pack	PDF
Week 1 - analysis	PDF
Week 1 - introduction	PDF
Week 2 - lists	PDF
Week 3 - trees	PDF
Week 4 - binary search trees	PDF
Week 5 - priority queues	PDF
Week 6 - hashing	PDF
Week 7 - graph traversals	PDF
Week 8 - shortest paths and minimum spanning trees	PDF
Week 9 - greedy	PDF
Week 10 - divide and conquer i	PDF
Week 11 - divide and conquer ii	PDF
Week 12 - algorithms_in_practice	PDF
Week 12 - empirical analysis	IPYNB
Week 12 - mock coding interview	IPYNB

Core concept 1: Abstraction layers



Core concept 2: Algorithm analysis

A principled framework for evaluating algorithms:

- measuring performance of resource use
- proving correctness

These should inform your design and implementation choices

Learning outcomes

- 1. Proficiency in organising, presenting and discussing professional ideas [...]
- 2. Using mathematical methods to evaluate the performance of an algorithm.
- 3. Using notation of big-O to represent asymptotic growth of cost functions.
- 4. Understanding of commonly used data structures, including lists, stacks, queues, priority queues, search trees, hash tables, and graphs. This covers the way information is represented in each structure, algorithms for manipulating the structure, and analysis of asymptotic complexity of the operations.
- 5. Understanding of basic algorithms related to data structures, such as algorithms for sorting, tree traversals, and graph traversals.
- 6. Ability to write code that recursively performs an operation on a data structure.
- 7. Experience designing an algorithmic solution to a problem, coding it, and analysing its complexity.
- 8. Ability to apply basic algorithmic techniques (e.g. divide-and-conquer, greedy) to given design tasks.

Beyond this unit of study

SCS offers many algorithmic units:

- COMP2022 Models of Computation (S2)
- COMP3027 Algorithm Design (S1)
- COMP3530 Discrete Optimisation (S2)
- COMP4445 Computational Geometry (S1)

New units coming soon:

- COMP4530 Discrete Optimisation (S2)
- COMP4270 Randomised Algorithms (S2)

Sydney Algorithms and Computation Theory group:

- weekly reading group on Algorithm research
- do a research project with us
- we are always looking for bright honours students

What is examinable?

Everything from the lectures, the referenced sections of the textbooks, the tutorials, the quizzes, the assignments. Exceptions to this rule:

- when explicitly labeled as non-examinable.
- probabilistic analysis of randomized algorithms

In general, if it happened during this unit, you are expected to know about it!

Focus on the things we put most emphasis on, as seen in tutorials and assignments

Final Exam Structure

2.5 hours writing plus 10 minutes reading

5 questions worth in total 100 points

Worth 60% of overall COMP2123 grade

Final exam has a 40% barrier

Restricted Open Book:

- You can bring handwritten or printed material (no limit)
- No electronic devices or any kind are allowed

Do's and Don'ts for CC students

Exam is in person.

Check you exam timetable for details on the venue!

Restricted open book exam:

- Can refer to slides, tutorial solutions, assignment solutions, books used in the unit
- Making a 2-page summary is highly recommended
- Never copy text verbatim from anywhere, including the slides (this is grounds for academic dishonesty case). If you refer to anything from the permitted material, write in your own words

Start your submission with your student ID

Don't include your name

Do's and Don'ts for RE students

Exam is online (Canvas quiz) and supervised using ProctorU (Live+).

Check your internet connection and webcam.

Restricted open book exam:

- Can refer to slides, tutorial solutions, assignment solutions, books used in the unit
- Making a 2-page summary is highly recommended
- Never copy text verbatim from anywhere, including the slides (this is grounds for potential academic dishonesty case). If you refer to anything from the permitted material, write in your own words

Neatest summary page I came across

Algorithm Analysis	>peletion: If node has I child, promote	open Addressing Using Linear Probing	Connectivity - connected if there is a	For every v in vis, we keep the	bivide and conquer - divide, recurse,	First Attempts	FROM TUTORIALS
Primitive operations - Q(1)	the child and replace the node. If node has 2 children, find node y	open addressing colliding item placed in diff Cell of the table	path bet every pair of vertices in G convicted component of G: maximal	distance to closest neighbour and the	conquer or elivide, conquer, merge	*Divide. Find median, and split on \leq and \geq then the median \geq 0(n)	Finding lowerbound: Use of Split into oddreven Reverse traversal
6. Clincaritims	following w in an inorder traversal.	Linear probing: place colliding item		olusest neighbour in a table Similar time complexity as Dijikstra	"No" otherwise, compare * to	Kecur IF Esna Find K" element in 1	-add pointer to the end o(n2) "Acares" -add pointer Avery in o(n)
Dies tonstant (6) ningn-quantineur	y would have no children, replace w w/ u and remove u.	in the next (circularly) available cell	Tree(T): T is connected , T has no cycles Forest: graph w/ no cycles, its connected	uskal's Algo -consider edges in	middle element Alland. If Allanz.	If K) = find (k-2)th element ink (onquer return value =)o(1) ()T(2)	-add pointer avery in o(n)
6 logan logan (1) nalogn	o w/ y and remove y. Scomplexity: space o(n) ops : o(h)	scolliding items lump together	components are trees * Every tree on n vertices has n-1 edges	-If adding e to T creates a cycle,	search L → A[O] to A[[=]-1]. Else, if A[[=]] <x, &="" a[="]+1]+n</td" recov.="" search=""><td></td><td>Pre-order: 12, 14/18, 23,1641,81</td></x,>		Pre-order: 12, 14/18, 23,1641,81
Landwing Comple	=> puplicate keys: key(L) & key(node) < key(R)	consecutive locations until an	Spanning Tree + Partit	discard e. Otherwise, insert e. - Choose edges hased on order of	Recurrence pivide 20(1) pecurat(2)	Aftempt 2: Approx. the median	Post-order: 18,16/11,23/4, 343/1
	buse list to store duplicates	richi wi Ken K is found or	Subset of graph G which has all the vertices covered w/ min edges; no cycles	weights -> no need for lists/tables	(onquer → o(1) T(n) = (*(9)+ o(1) n>1	1A1/3 \(\left(\text{A}_1 \text{X} \)) \(\left(\text{2} \) \(\left(\text{A}_1 \text{X} \)) \(\left(\text{2} \) \(\left(\text{A}_1 \text{X} \)) \(\left(\text{2} \) \(\left(\text{A}_1 \text{X} \)) \(\left(\text{A}_1 \text{X} \	En-order: 13:14, 16:13:11,
	9 Range averies: search all keys k Such that K,≤K€K2	empty (ell is found or 11 cells have been probed.	-can't be disconnected	all costs are integral, if we add 1/2		$T(n) = \{T(\frac{2n}{3}) + o(n) \text{ and } = \} O(n)$	11/34, 13/31
Big-on Notation => upper bound on RT -n" is O(a") for any fixed x>0 + a>1.	> Key(v) < K, > R > Key(v) > K2 > L > K1 & Key(v) = K2 > L1 add v1 R	ODETUNCT replaces detailed elements	opth First search - follows outgoing edges leading to yet unvisited vertices	to each edge e; then any Mil	1: T(n) = T(1) + 2 () logn: T(n) = 1 *clogn 2: T(n) = T(2) + 2 () T(n) *(logn) i: T(n) = T(2) + i ()	T(n) = {T(\frac{19}{3}) + O(n) \text{ost} \text{ost} \text{ost} \text{ost} \text{ost} \text{ost} \text{ost} \text{ost} \qua	INDUCTION: (divide -and-conquer)
"logn" is O(logn) for any fixed x>0	* Yunning time: O(loutgut] + h)	get(k) must pass over cells w/ DBFUNCT and keep probing	-if edge discovers a new vertex, it's	under the perturbed weights is	ARECUTTENCE FORMULA includes recurreconquer	find the median let & be median	If (a) = 0 return faire
	Irinode Restructuring - Ralancing Trees	wout . If V is sound reules	called a DFS edge . Oth ., it's a back edge	Time complexity: sorting edges : Dominging	#Recurrence formula includes recurreconquer If linked list used: T(n) = fT(1)+0(n) not Werde Sout	of the medians	$mid = \lfloor \frac{lA!}{2} \rfloor$
Sin-Assess services (A) alterests based	- (Albic) a c > a b c > a b c	otherwise, store it at index i	8 D > 8 D = + DD = + 5 D =		+ Divide into 2 hauses merure an house	T(n) - T(2n/a) + T(n/a) + O(n) => O(n/cen)	IF A[mid] <v: (a[mid:],v)="" a[mid]="" binsearch="" if="">v: BinSearch (A[mid],v)</v:>
Sig-Theta Notation(0) => asymptotically Transitivity:	-takes o(1) since you just have to upage +	performance: wc : get, put, remove o(h)	Performance AS AS	Union Find ADT Keep track of an evolving partition of A.	Feet track of smallest element in coch	brecursive call on % elements. Getrid of 20% elements in each call.	if a [mid] = V) return True
If f = 0(9) + 9 = 0(h), then f = 0(h)	AND Trees r(v) is height of subtree	If randomly distributed, expected # of probes is 1/2-0 where a = 1/4	assuming adjust rep : O(men) -main DFS function: visits all vertices 20	Simple union-find:			Proof: Base care: IAI+0 which returns Passe
sums of functions:		It & 15 constant & I propert at it out	-main DFS function: visits all vertices to	make-sets (A) O(n) where neldle find(u) O(1); union(u,v) -) O(n)	-divide: U(n); recor: 21(2); conquer o(n) T(n) = {27(4) ro(n) n > 1	3/A1/10 Crownk(4/x) < 7/A8/10	: Algo is correct
	every internal node differ by at most 1 >height: O(logn); space O(n); ops O(logn)	and 2 hash to les	-DFS-visit(u): 0 (deg (a)) - depends on deg-of- -> called a times so O (Edeg(u)) = 0 (m)	Kristal's alan: a(n2) => find: 2m calls	Sample Recommence Formulae	T(n) = T(71/10)+T(1/5)+O(n) =>O(n)	comple can arrang of Size Alek
	eriority augus: can only remove the	4) get, remove - p(1)	prognot PPS (Let covint) the the Countering of a reps. voltect of the process of	Moto head at the distant and and seek	fatelly action of	Grecursive call on 1/5 eleverits.	use 14 to prove Binsearch for arrays of size 121 = K.
	MINERALKINI, LEMONE-MINIS MINIS	Sevict previous item timsert new; the evicted goes to its other possible	-Edges (tu, parentho) in in cof form a span	when taking unit n of 2 sets, change the smallest. Element can change	= 27(2)+0(1) => 0(1)	-choose random element as a pivot and partition into 3: (i) < (ii) = (iii) > (in)	case to a finish out we have three i county
- W = B . se P. see = Wande	traceyand Council (bu prictity)	LINCE	cut edger. In a constitution of span off	sets o(logn); seq. of n union ops of		"Recor select right element from list 160)	case 2: A[mid] < v we'll do Bin Search on array of size [A]
	size/sewely o(t) o(t)	Eviction cycle: keep counter/put Plag	is not connected	*Kruskals : O(mlogn)	= 7(n-1) + O(n) = O(n1) $O(n1)$		By 1H, this must return correct result
LiC+C - Abrievast mass Tune I defined	PERMOUT WAY min O(N) 9(1)	Set - unordered collection of elements w/o duplicates; ops are traditional	4) 0 (m²): For each edge, remove (ww).	Greedy Algo Fractional Knapsack - given a set Sof	= t(n-1) + o(1) =) o(n) A	$B[T(n)] = \begin{cases} B[T(n')] + O(n) \Rightarrow O(n) \\ O(1) \end{cases}$	for that sub-array. If present in sub-array, it must be present in
Data Structure - concrete rep. Benaviour	Priority aueue sorting insort keys in insert, in remove-min remove min to sort	ser operanons: union, contains, etc.	check if Off is still connected, put back	n items w/ each item (having bi (+ve benefit) t wi (+ve weight),	Au(ick copy). Bivide change roundom element as proof. Partition into 3: (i) < (ii) = (iii) > 1. Recurring up to 4×10^{-1} (iii) = (iii) > 1. Recurring up to 4×10^{-1} (iii) = (iii) > 1. Congress John 3 (iii) together $\Rightarrow O(n)$ the	Randomisation hands	array correct present in subarray
>size(), get(i), set(i,e), add(i,e), remove	"So o(n2) Selection Some -some and first	-map to store keys; ignore value	6) O(nm) - bnly test edges in a PFS tree of G 6) O(vitm) - Compute PFS tree of G for every v in V, compute level[v]	choose items w/ max total benefit	3. conquer Join 3 111 to gether 3 0(n) np	Input: integer 1; Output: permof (12, m)	then it must not be present in the array since the given value must
set(), get() => o(1) ind. of size	>sort unsorted seavence >can be implace	-contains(k) answered by get(k)	for every v in V, compute level[v]	of weight at most w Shest: items w/ nighest bi/w; rano.	1 000	charge maisermaly at random 1110R)	be in the right half of the urray
add() remove() => o(=) . cuicling	A[0,i): sorted; A (i,n]:pa In inserts - o(n); n vemove.min - o(n2)	Graph - consists of a pair (v.E)	down-and-upsyl: neight of vertexy that can be reached by taking DFS tree	Ocomplexity: O(nlogn) for Sorbing	Maxima - 266 - H be is max it all other bes	then we want any 2 permutations to	as its correct and Almido < v.
space: O(N) richange size as you add	1100k at the Smallest after given	Eage (E): directed (u,v) > v origin/ u>v	down the and then one back edge up # DFS edge (UIV) is a cut edge where v is	(USE PO NEAD TO EOUN removal takes		be denerated by the same to be execut.	to a first of
	index + then swap places insertion sort: sort front first	undirected avanue - edges connect endets		Task scheduling - awen a set of		Fisher-Yales snuffle je (1,, n-1) - swap ACI I with ACJ where I	GXCHANGE ARGUMENT:
=> first(), (ast(), before(p), after(p),	>Best: sorted ast; worst: sorted desc.	reduces are incident an audust	10 Back lauter Guck 1 1 a sect 1 1 . verficts	TASK scheduling given a set of hope hitchurts. Lecture i starts at s; and finished at fit find min be of classing min fit.	3. Recursively find MS in both halves 4. Conquer. Compute MS of MS_UMSR	of element will either stoly in place or	a b input a* b* be optimal soln
insert Before (pie), insert After (pie), remove singly contract est - reference to first node insert Before.	on insert > 0(nt); n remove.min > 0(n)	"Adjacent vertices are connected when e degree = normedges on a vertex		of classrooms to sched all rectures s.t. no 20ccur @ same time tplace	Stind nights + pt - p of MSR. compare	snoffle w/ something after	a'b' in sorted order erreturned by
			26-2-p -> 28-2-0 - 8-2-0 - 27-	S.t. no 20100 @ same time tplace	every pe q in MSL to p. If qy >py, add	# of executions = 1x 2xx n +n'. # of permutations = 1x 2xx n =n'.	assume at \$ A', b' = b'.
Shas a header + trailer =75pace: o(n)	given index and the insert index before greater value	- Self-loop: only I endpt - Simple graphs: no parallel/self-loops		Interval Partitioning	q to merged Ms. Add every pt in MSp to merged Ms. T(n)=2T(1/2)+0(n)=0(nlogn)		there must exist i such that a >ait
Gall ops - O(1)	Heap : BT where pointers are only at	Directed around - edges : tail to head	会一会一会一会。	-Sort intervals by starting hime - when space is available, put it	0 O(nlogn) @ O(n) @ 2T(1) (0 O(n))	the outcome.	10 - 6 1 + 10 + - 6 + 1
STACK - Last in, first out spop() remove	rank + lact inspetted node	-Out degree: num edges out of vertex In degree: num edges into a vertex	Properties: Bes visite au	in a class room w/o caring about which is best. othi open new classro.	Integer multiplication given 2 n-digit	Finding Prime numbers	btis bin: swaper
	>heap-order: key(m) ≥ key(parent(m)) >complete BT: every level i <h full<="" is="" td=""><td></td><td></td><td></td><td>integers 2+4, compute the product 24. ATTEMPT 1: Compute by making 4 recursive</td><td>Distribution of Primes</td><td>[a = - b = + (a = + b = 1 ≥ a = - b = 1</td></h>				integers 2+4, compute the product 24. ATTEMPT 1: Compute by making 4 recursive	Distribution of Primes	[a = - b = + (a = + b = 1 ≥ a = - b = 1
to en a ueue(e): insert at end; deaueue()	Gremaining nodes take leftmost pos.	-self-loop same tail+nead heads or anti-parallel same endpt but opp. T	Path in Ts from 3 to v with i edges.	maintain the finish time of last job add (d. Keep classrooms in PQ	calls on a digit numbers t combining it	Let ((n) be the # of primes =n,	
remove at eront; first(), size(), isemptu()	>root has smallest key, neight is logn >Upheap: restore to by swapping keys		-For each v in Li, any path in G from	efficient encode it to smaller y	x=x,2 ³ + x ₀ ; y=y,2 ³ + y ₀ xy=x,y,2 ⁿ + x, y ₀ 2 ³ x + x ₀ y,2 ³ + x ₀ y ₀	then IT(n) = 0 (1) probability of n to be prime if in where ne firming	After swapping, at is now similar to a without reducing optimality
Shased on arrays: end=(startsize)model (stand unward outh evolution insertion ob-	but can have anti-parallel edges	s to what at least i edges. Performance, setting things up of ni)		T(n) = [4T(2)+cn n>1 =) 0(n2)	2 functions: Find primes) among a primes	:- By exchange argument, also is correct
Gallow insertions + deletions @both ends	FEMOUP _ min : replacatout Kew w/	62 simple path: all vertices are distinct	processing each layer O(Edeg(u)) = o(m)		ATTEMP 2: COMPUTE by making 3 rec. caus	-helper is_prime runs T(n) so find-prime	Divide Seep
9 getfirst/last, kadf/L, remove F/L => O(1)	last inserted; restore his budownheap	Cucie: park that starts + ends at	- adj. (15f - 0 (mm); adj mostrix 0(nt)	Huffman encoding the c be the set	xy= x,y,2" + (x,yo+ x,y,)2"/2 + x,040	-helper is-prime runs t(n) so had prime runs in O(T(N)logn) awas a bounded Rabin-Miller -testing primality	
Trees (Tree ADT) node has at most	Sawap Keys along downward paHafrom root. → 0(logn) 0(logn)	vertex but not same edge.	BFS: snortest paths; Both: cycles, paths	of characters in X. compute freq. fcg for each character c in C. Encode	(x, + >(+)(+)(4, +40) = x,4, +x,40 + x,4, +x040	Given n+k, If n is prime, RM(n,k) ->T.	array within each recurrence. You can simply make note of the
- Internal node: node w/ at least I child		Gsimple cycle: all vertices are distinct	weighted Graphs: each edge has a weigh	high freq. char. w/short code words. No code words is a prefix	T(n) = (3T(4)+0(n) n>1 => orn 109:3)	Else, if n is composite, KM(n,k) returns	start and end index of the smaller
* Fxternal/Leaf node: node w/o children *Ancestors, pescendants, Siblings	HEAD SOFE - O(hlogn) => P(x sorting	Properties of a graph - Evinv deg(v)=2m -n (# of v) i m (# of e); A max degree		Encoding Tree - code: mapping of	6:0(1) 21 =1 =1 (10gh 1) 1 3 = 2 (0g(1) 1) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	True w/ prob 1/4k; False otherwise def witness (xm): # check if n is comp.	array. This makes it run in o(i).
* Depth of a node : # of ancestors	Heap-in-array - root at index of array - 10st node at index (n-1)		eachstage in the nope of finding a global optimal roln. Promis the min.	each char, to a binary code word teach external node stores a char.	2: 5(37(%)+2)+n=97(%)+30+n) = 2		Ageonnote principle
not incl. itself - Height: max depth - Level: set of nodes w/a given depth	mode at index i: 1. 2 21+1: 8-721+7	- simple directed: M ≤ n(n-1) Subaraph: Let (n*(V,E) be a graph.		-code word: path from root to external node (03L; 13R)	2: 3(37(\frac{1}{2}) + \frac{1}{2}) + n : 97(\frac{1}{2}) + \frac{1}{2} + n) \frac{1}{2} = 2 1: 3' T(\frac{1}{2}) + \frac{1}{2}(\frac{1}{2})' n	if zemedn=1, return True & prime	-if items are put into containers, then attenst one container contain
"Edge: pair (u,v) where one is the parent	Oparent → L(i-1)/2]	S=(u,r) is a subgraph of a if USV, FSt	there is a tree of shortest paths from a start vertex to all other vertices	put smaller tree as left child	logn: 310gn + 2 (1) cn = 0 (310gm + (1) cn)	for i in fil	then attemst one container contain more than I item. (majority)
	Hash Tables Mad - searchable collection of key pairs	Subset subset use induces a graph	-edge weights are nonegative	when combining trees	= 0 (3,000 tu T 3,000 to 1,000 tu)	return faise	Graphs: Bipartite: versex set can be
post-order visit node after descendants	=> net(), put(), remove, size()	GEW = (U, FEW) where EW are edges in t w/ endpts in u . subset FSE induces		-Huffman's Tree : O(n+dlugd)	= $O(u_{pod_{1}y} + u_{pod_{1}y_{2}}^{n}) + o(u_{pod_{1}y} + (u_{pod_{1}y} + u_{pod_{1}y}^{n}) + o(u_{pod_{1}y} + (u_{pod_{1}y} + u_{pod_{1}y}^{n}))$	frf n>2 is comp., there are sni values of x s.t. witness(x, n) = True.	partitioned into 2 Jets At 15
Smary Free - each node has at most 2 ch. Sproper BT: every internal node has 2 ch.	a) put: if key already present, replace	a graph GCF] =(V[F].F) where V[F]	- without PEs 1 = 0; DEV] = 00 for all vin	where n is the size of X and d is the # of distinct char. of X		LIE WE CALL WITHESS(xin) w/k diff values	s.t. 18th ECA×B -intra-layer: edge w/in layer
Inorder : node visited after L, before R	=) put -O(1); get, removeo(n)	edge list structure ville E: FILLA	-In each iteration, add to 5 vertex u in VIS with Smallest DEu]; update	St(c) x depthr(c)	Connectic Series Let KER, KER	of w evelopility & Yet	-inter-layer: edge bet. layer
-on the left(pre), from below(in),	index; map w/ n items + N keys	-vertex list : sea - of vertices, vertex	D-values for vertices adi. to u	x-abracadabra a b c d 2	### 1 + + + + + + + + + + + + + + + + +	main thelper = O(klogn)	To the state of th
on the right (post)	ops are all O(1) but N can be big (space)	objects keep track of its pos. in the	Performance: o(m) on everything Except Pa pperanence: A (net plane)	2 7 5 3	MASTER THEOREM		
	to map keys corresponding indices	-Edge list: Sea edges; edge objects	except Pa operations: o (m+nlogn) heap as Pa: o(mlogn), Fibonacci heaps;	5 cdbr ab Edc	T(n) = far(2)+f(n) n>1	GIF pi is chosen VAR from [0,1] then	
-your : element, pointer to parent, sea, of		keep track of its position in the sea; points to the edges + the endpt	minimum spanning Tree . tree whose som of edge weights is minimised.	11	Off f(n) = O(n)0gha - E) for E>0, then	for a treapon f(vi,pi)sil	
	ideally, there should be no collisions	adarence not			T(n) = A(n10969). (T(n) is dominated	E[Treapheight] = O(109n) Treap insert: O(109n)	A .
") get(), put(k,v), remove(k) => <u>sorted wap ADT</u> : keys have sorted order	ideally, there should be no collisions (Items Stored at the same hash value)	-tach vertex Keeps a seq. of edges	and let e be the min cost edge w/ exact lender in S. Then, the MST contains e	c d b r 0 c 3 1 8 2	by the last level)	-00 regular BST insertions and	
	of 2 functions: h(x) = h1(h1(x))	-edge objects keep ref to pos in the	-cycle property : let C be any cycle and	8-0 C-100 r-111 C d b r	@ cf f(n) = 0 (n) 1984 [0g (n)) for x 80 , rest T(n) = 0 (n) 1984 [0g kn n) . (T. c same for [8][0]) @ cf f(n) = 0 (n) 1984 [0g (n)) (T. c same for [8][0]) @ cf f(n) = 0 (n) 1984 [0g (n)) (T. c same for [8][0])	restore neap property by doing	
=) internal modes store Key-valve pairs	>hash code h.: Keus -in baers	incidence sea. Of its endpts	let f be the max cost edge belonging to C. Then, f mast not be in MST.	of leaves that are siblings	SIF f(n) = 1 (nicquare) + af(1) sg for 8>0+	(5,10) (6,4)	
	>compression func. ha:integers >[0,N-1] - Probability of collision: /N where N. if - Probability of collision and more prime	-good for sparse graphs	- Cut, nonempty S CV; cutset Dis :	the control of bill c, it deprojust suchit	3.0, were the tenerty	VY (P14) (P10)	
			the subset of edges w/ exactly lendpt -Cycle + cutset intersect in an even	- 2 siblings furthest from root have		A A A	
Gruns in O(h), worst case O(n)	Ocreate a list wiin hashvalue	-good for dense graphs	# of edges . > Every time, we add an	towest freq 18 we combine 2 towest freq. char. to	If ningea > f(n), () tf ningea < f(n), (3)	treap Height: Suppose we sorted values so that visves Evn 12	
=) Insertion: If present, replace value.	GLOAD FACTOR & = 1/4 Gexpected: 0(1+4); worst-case: 0(n)	and territory, some graphs Ferformance cage use Adj. use Adj. viatro Force memory of the some new new new new new new new new new ne	frim's Alag property. We add the	get a new instance legt in an optimoly	If n'ester . f(n) , (2) Selection Given unsorted array A holding	bullit the root] = " pulloge fre]	
Principal made out nous year commence	when all items coulde to a single		Type teastf interfer in any add an front algo people, we follow our property. We add the min cost edge (u.y.) S.t. u. in S. Androve and the An	The we combine 2 lowest free, char. to, get a new instance (c.f.), an optimized the test for c'can be expanded to get optimal tree, for c see the primal tree, for c see the primal tree, in each up, 2 removes that it is a combined to the coxulating free, is o(n).	a numbers + an integer K, find kth	(Vi,Pl) =) Most nodes are	
~ 2	Chain	removed frittion in degree of	notes select distances of Adj. cettered	oth, calculating freq. is o(n).	Statement of the first	(V1,Vi) (VictVin) =) 0(10gn) height	

20 points

Analysis of given algorithms

Easy problem. Make sure you nail it!

20 points

Analysis of given algorithms

Easy problem. Make sure you nail it!

20 points

Design or modify an ADT/algorithm that solves a problem

Medium difficulty problem.

Remember to:

- Describe your approach
- Prove correctness
- Analyze complexity (if there's a space requirement, don't forget to analyze this as well)

20 points

Design or modify an ADT/algorithm that solves a problem

Medium difficulty problem.

Remember to:

- Describe your algorithm
- Prove correctness
- Analyze complexity

20 points

Design or modify an ADT/algorithm that solves a problem

Hard problem.

Remember to:

- Describe your algorithm
- Prove correctness
- Analyze complexity

Problem 3, 4, and 5

Check if you're supposed to use a specific technique:

- "design a greedy algorithm"
- "design a divide and conquer algorithm"

(Using a different technique will cost you a significant number of marks, but may still be better than a poorly explained incorrect attempt)

Let the running time requirement guide you:

- If we ask O(1) time, this limits your options considerably
- If we ask O(n) time, you can't sort the input

Exam technique

Read all questions to see which ones you can answer quickly

Plan how you will allocate time (wisely)

Start with easy problems and move to harder ones

Write clearly and efficiently

- Start with outline/bullet points, then expand if you have time
- No need for fancy style or overly formal

Pragmatic Advice (for CC students)

- It's a good idea to check the exam venue ahead of time
- Plan to arrive ahead of time (don't rely on public transport running smoothly on the day of the exam)
- Bring water, spare pens, and ID
- Have clothing in layers
- Start by writing your student ID. Do not write your name on the exam (marking is anonymous)
- Breathe and relax
- Follow the instruction of the invigilator

More info:

https://www.sydney.edu.au/students/exams/in-person.html

Pragmatic Advice (for RE students)

- Be alone in your room
- Let housemates know when your exam is to avoid distractions
- Bring water
- Have clothing in layers
- Breathe and relax
- Make sure your webcam is working
- Follow the instructions in Canvas

More info:

https://www.sydney.edu.au/students/exams/online.html

Quick announcements

Fill out online Unit of Study Survey

- https://student-surveys.sydney.edu.au/students/
- Use the free text to help us make this better for next years students. "Pay it forward"

Examples of changes based on past year's feedback:

- wrote the guide on how to approach algorithmic problems
- designed programming exercises