$\lim_{n\to\infty} \frac{f(n)}{g(n)}$ | 0 | $c<\infty$ | $\mathbb{R}^+$ | >0 | ∞

bigO | little-o | Θ | Ω | ω

cmp. | $f<g$ | $f\le g$ | $f=g$ | $f\ge g$ | $f>g$

$\log_b a = \frac{\log_d a}{\log_d b}$, $3^{\log_2 n} = n^{\log_2 3}$

$T(n) = \sqrt{n}\,T(\sqrt{n}) + n$, $T(n) = \Theta(n \cdot L)$, $L = \log\log n$

idea: $n = 2^{\log_2 n}$, $\sqrt{n} = 2^{\frac{1}{2}\log_2 n}$, exponent halves

time $= \Theta(\frac{W}{P} + S) = \frac{n}{p} + d$, $W = n$ = # nodes, d = |path|

reduce ⇔ iterate, scan ⇔ iterate Prefix ↗ W: nm, S: log m

Contraction $C = \langle f(s[2i], s[2i+1]) : 0 \le i < \frac{n}{2} \rangle$

recursion (R, ans) = scan f b C ↗ if even, O(n) work, O(1) span

expansion $E[i] = R[i/2]$ or $f(R[\lfloor i/2 \rfloor], s[i-1])$

E.g. (merge cmp) ↳ if odd, $\sum_{i=0}^{n/2-1} O(m(i+1)) = O(n^2 m)$

then $W(n,m) = W(\frac{n}{2}, 2m) + O(n^2 m) \Rightarrow$ root, $O(n^2 m)$

$S(n,m) = S(\frac{n}{2}, 2m) + O(\lg nm) \Rightarrow$ bal., $O(\log n \cdot \log nm)$

$\sum_{i=1}^{n} r^{i-1} = \frac{(1-r^n)}{1-r}$, $H_n \le \ln n + 1$, LV correct, MC fast

$W(n) \in O(f(n))$ W.H.P if ∃ constants c, n_0 s.t.

$W(n) \in O(k \cdot f(n))$ $\forall n \ge n_0$, w.p. $\ge (1 - \frac{1}{n^k})$ $\forall k$

① insertion sort: on iteration i, first (i-1) elements sorted, insert i-th element by swapping left

$E[X] = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} E[X_{ij}]$ → I.r.v. 1 ⇔ i ↔ j, $\Pr[A[i] > A[j]] = \frac{1}{2}$

② $X_0 = n$, stop when $X_m \le 1$, current length = i

$E[\text{cut size}] = \sum_{j=1}^{i} \frac{j}{i} = \frac{i+1}{2}$, $E[\text{rem}] = \frac{i-1}{2}$

$E[X_n] = \sum_i E[X_n \mid X_{n-1} = i] \Pr[X_{n-1} = i]$

$= \sum_i \frac{i-1}{2} p_{X_{n-1}}(i) \le \frac{1}{2} \sum_i i\, p_{X_{n-1}}(i) = \frac{E[X_{n-1}]}{2} \le \frac{n}{2^n}$

$\Rightarrow \Pr[X_{k \log_2 n} \ge 1] \le \frac{E[X_{k\log_2 n}]}{1} \le n(\frac{1}{2})^{k \log_2 n}$ union bound

$= n(\frac{1}{2})^{\log_2 n^k} = \frac{1}{n^{k-1}}$ $(\Pr[\text{any bad}] \le \sum_{i=1}^{k} \Pr[i^{th} \text{ bad}])$

③ Quicksort: generate random priority p for each $e \in S$, choose e w/ highest p as pivot → i,j compared if either has highest p in range of ranks $i \le r \le j$

$E[X] = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \frac{2}{j-i+1} = 2nH_n \in O(n \log n)$

↗ a treap!

pivot tree = # recursive calls to place node in sorted spot, span = max depth = $O(\log n)$ W.H.P (# calls to quickselect)

- rank k = element k in sorted (ascending seq)
- $\Pr[\text{pivot tree of height } n] = \frac{2^{n-1}}{n!}$ → pivot either max/min repeatedly

↳ same for treaps!

BST & Treaps: balanced W.H.P.

→ traversals: inorder(L,N,R), preorder(R,L,R) (root)

→ joinMid $\in O(\log(|T_1|+|T_2|))$ work & span from rebalances

split, find, insert, delete T k $\in O(\log|T|)$ ← depth

treap: BST with priority function $p: U \to \mathbb{Z}$, U = set of keys

1. BST invariant: ∀ node(L,k,R), $\ell < k < r$
2. heap invariant: $p(k) > p(x)$ $\forall x \in (L \cup R)$

let $A_j^i = 1 \Leftrightarrow S[i]$ ancestor of $S[j]$ in pivot tree

↳ depth(j) $= \sum_{i=0}^{n-1} A_i^j$ (cd), size(j) $= \sum_{i=0}^{n-1} A_j^i$, $E[d] = H_{j+1} + H_{n-j} - 1$

$\rightarrow \sum_{j=1}^{n} A_j^i = \sum_{j=1}^{n} \frac{1}{|j-i|+1} = H_i + H_{n-i+1} - 1$

1. ∃1 unique path between 2 vertices in a tree
2. (n-k) edges in forest of n nodes & k trees
3. in DAG, every vertex is SCC: either $u \not\rightsquigarrow v$ or $v \not\rightsquigarrow u$
   ↳ by def! no nontrivial CC (⇐ not true!!!)
4. V_j reachable from $V_i \Rightarrow i < j$ in topological ordering
   $a \le_p b \Rightarrow a \le_t b$, $\le_t$ total ordering

SETS: implemented via arrays/hash tables/balanced trees

TABLES: sets of key-value pairs

↳ restrict T S = restrict domain to ones in S

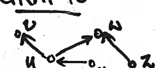
subtract T S = $\{k \to v \in T \mid k \notin S\}$

- Ord tables: ordered keys (total ordering)
- Aug tables: values reducible w/ combining function

e.g. type t = int, f = Int.max, I = ∞, toString

Ex: allval(T) = Table.reduce Set.union ∅ T

GRAPHS



- v is outneighbor of u (u...in...v)
- $N^+(u) = \{v, w, y\}$, $d^+(u) = 3$
- $N^-(w) = \{u, z\}$, $d^-(w) = 2$

→ path: seq. of adj vertices, simple if no repeats (edges)

cycle: same start & end vertex, no repeat edges ↳ trivial = self-edge

undirected G connected if $\forall u,v \in G$, $u \rightsquigarrow v$

directed G strongly connected if ↗

A. adjacency matrix: good on dense ($m \in O(n^2)$) graphs, bad on sparse ($n \in \omega(n^2)$) graphs, nb query O(n)

B. adjacency sets: (V × V set) table ← lookup u's nb, check if v in set.

C. adjacency sequence: int seq seq. (tree-based)

→ -joinMid on Node(L,k,v,R) requires that L<k<R

- BST desires: BST property & balance ($h \in O(\log n)$) $\ge \lceil \lg(n+1) \rceil$

DFS: fully explore all nodes reachable from vertex before moving on to next vertex ("tunnel vision")

→ applications: detecting cycles, finding SCC (Kosaraju), topsort: partial/dependency order, reverse wrt finish time (require DAG)

↗ encodes edges traversed during DFS search

DFS tree: tree, forward, back, cross edges

DFS ~~numbering~~: increment counter when visit/finish

E.g. critical edges, $\Sigma = (i, T_v, C, L, pLim)$

update $L[p] \leftarrow \min(T_v[v], L[p])$ when revisit non-self

finish: if $L[v] \not< T_v[p]$, $C \leftarrow (v,p)::C$, → i.e., not in the same group

else, $L[p] \leftarrow \min(L[v], L[p])$

DFS G ((Σ,X), v) = DFSAll G Σ = iterate (DFS G) (Σ, ∅) V

if $v \in X$ then (revisit (Σ, v), X)

else let add v to visited set } explore all reachable from v }

$\Sigma' $ = visit Σ v, $X' = X \cup \{v\}$

Σ'', X'' = iterate (DFS G) (Σ', X') $N_G^+(v)$

in (finish Σ'' v, X'')

operation *DFS sequential	# times computed
$v \in X$ → O(1) for adj. seq.	n+m
$X \cup \{v\}$, N_G^+, visit, finish	n
revisit → O(log n) work for AT	m

} sum for total DFS cost

BFS: explore all neo nb. of current frontier in parallel

→ diameter: length of longest shortest-path = # layers to search graph from source

BFS G s = $\|F\| = \sum_{v \in F}(1 + d_G^+(v))$

let explore X F = ↗ frontier empty = nth to explore

if |F| = 0 then X ↗ new frontier = out-nb - visited vertices

else let $X' = X \cup F$, $F' = N_G^+(F) \setminus X'$

cost-analysis: compute work/round in terms of ||F|| & sum across d = diameter rounds

	work	span
sequential BFS	O(m+n)	O(m+n)
parallel w/ tables	O(m log n)	O(d log² n)
parallel w/ STSeq	O(m+n)	O(d log n)

use queue, iteratively pop & add nb. to frontier

↳ better update time complexity on enumerable graphs.

SHORTEST PATHS

- subpaths property: $\delta_G(s,v) \le \delta_G(s,u) + w_G(u,v)$
- if unweighted: BFS (single source)
- Dijkstra's property: if all edge weights ≥ 0, $y \notin X$ minimizes $\delta_G(s,x) + w_G(x,y)$, then $\delta_G(s,y) = \delta_G(s,x) + w_G(x,y)$

↗ priority-first search

Dijkstra: start with d(s)=0, pop (d(v),v) from PQ & save d(v) as min. dist to v, for each neighbor u of v, add (d(v)+w(v,u), u) to PQ

NOT parallel: work = span = O(m log n)

↳ with Fibonacci Heaps: instead of adding duplicate nodes to PQ, decrease dist of existing entry in O(1)

⇒ work ∈ O(m + n log n) ↘ n inserts & deletes, m decreaseKey

* inserting only if unvisited ⇒ same asymp. costs.

dijkstraPQ (G, s) = ↗ (minimum PQ) lower value ⇔ higher priority

let dijkstra (X, Q) = ↗ if PQ empty ⇒ done

case deleteMin Q of ↗ if alr visited, ignore & continue

(NONE, _) ⇒ X

| (SOME (d,v), Q') ⇒ if v ∈ X then dij(X,Q')

else let $X' = X \cup \{v \mapsto d\}$, w(u,v)

relax (Q, (u,w)) = insert(Q, (d+w), u)

in dij(X', Q'') end

in dij({}, singleton (0,s)) end

1. multi-source: BFS w/ U as initial frontier S
2. shortest paths to U: reverse edges (flatten & collect)
3. minimize max weight: insert(max(d(u),w), v) Q

Bellman Ford: ∀v ∈ V, shortest path from s to v that uses ≤ k edges (table of k-hop distances)

↗ (longest cycle has (n-1) edges)

- converge in ≤ n iterations ⇔ no neg. cycle

sequences W ∈ O(mn) S ∈ (n log n)

tables W ∈ O(mn log n) S ∈ (n log n)

updating: $D(v) \leftarrow \min(\text{existing } D(v), \min_{u \in N_G^-(v)} D(u) + w(u,v))$ ↖ in-nb of v

Johnson's: for all-pair shortest paths.

- create dummy node connected to all vertices, (<∞)
- run BF, $w' \leftarrow w - \phi(b) + \phi(a)$ just need to be well-defined
- Dijkstra from every vertex, W ∈ O(mn log n), S ∈ m log n

any weights: only care abt shortest path to each vertex

Edge Partition: contract edges with priority greater than neighboring edges

e.g. (u,v): $Pr[\text{contracted}] = \frac{1}{d(u)+d(v)-1}$

alt: contract edges w/ H & all nb are T

$Pr[\text{selected}] = \frac{1}{2^{d(v)+1}} \rightarrow \log_{\frac{8}{7}} n$ contractions if $d=2$

$Pr[\text{vertex removed}] = c$

Star Contraction: flip coin for each vertex.

if H, star center, else satellite ⇝ contracts into adjacent SC, o.w. becomes a SC ⇝ $c \geq \frac{1}{4}$ $(n \rightarrow \frac{3n}{4})$

starPartition(V, E) = let

  TH = $\{(u,v) \in E \mid \neg$ heads u $\wedge$ heads v$\}$

  $P_s = \bigcup_{(u,v) \in TH} \{u \rightarrow v\}$ say V: int seq, E: (int, int) seq

  $V_c = V \setminus$ domain P_s $V' = \langle j : 0 \leq j < |V| \rangle$

  $P_c = \{u \rightarrow u : u \in V_c\}$ P = nInject V' TH

in $(V_c, \underbrace{P_s \cup P_c}_{\rightarrow P})$ end $V_c = \langle j \in P \mid P[j] = j \rangle$

V_c = seq of SC, P = mapping of v to star center

SP: $w \in O(m)$, $s \in O(\log n) \Rightarrow$ SC $\rightarrow s \in O(\log^2 n)$, $\rightarrow w \in O(m \log n)$ — over all rounds

→ pf: AFSOC ∃ 2 MSTs with ≥1 differing edge

MST: unique MST ⇔ unique edge weights

→ pf: (unique edge ⇒ trivial) AFSOC $e' \in T$, $T - e' + e$

- light edge property: the edge with the min. weight crossing the cut $U, V \setminus U$ must be in the MST
- cycle/heavy edge property: cycle $C \subseteq G$, then heaviest edge in C cannot be in the MST

$O(m \log n)$ w/ PQ, $O(m + n\log n)$ w/ FibHp

① Prim's Algo: sequential, start from single vertex, in MST — GRD

select lightest edge crossing cut $(T, V \setminus T)$

↳ ~Dijk: keep vertices & shortest distances to MST in PQ

→ $O(m \log n)$ to sort edges — GRD

② Kruskal's Algo: sequential, consider edges in increasing order of weight, add to MST if no cycle created → terminate early if graph connected.

③ Boruvka: parallel: each iteration, each vertex selects lightest incident edge & contract (tree/star) → add contracted edges to MST

↳ vertex bridge for v (lightest across $\{v\}, V \setminus \{v\}$)

e.g. (v,u) contracted if v tails, u heads ↗ i.e. selected by v.

- $w \in O(m \log n)$ in expectation $E[v \text{ contracted}] = \frac{n}{4}$
- $s \in O(\log^3 n)$ for tree, $O(\log^2 n)$ for star

$\geq \frac{n}{2}$ edges selected & contracted

work: $O(m)$ edges per round, $O(\log n$ rounds$)$

span: per round: $\log^2 n$ span for tree, $\log n$ for star

Dynamic Programming → time efficient

- top down: use memoizer, start from original problem
- bottom up: recursively compute larger problems in topological order, start from smallest

  ↳ space efficient, throw away unneeded results

span: sum of spans of longest dependency chain

improved with Fib heap — GRD

Heaps: implementation of PQ (for Dij., Prim)

	insert	deleteMin	meld	fromSeq
unsorted list	1	n	m+n	n
sorted list	n	1	m+n	n log n
balanced trees	log n	log n	m log(1+n/m)	n log n
binary heaps*	log n	log n	m+n	n
leftist heap	log n	log n	log m + log n	n

* complete binary tree satisfying heap invariant → holds for every node (!!)

- leftist heap: heap property $(p(p) > p(c))$ + leftist property (rank of left subtree ≥ rank of right subtree) → $r(\text{Node}) = 1 + r(R)$
- leftist rank lemma: $r(\text{root}) \leq \log_2(n+1)$
- meld traverses right spine: take at most $r(A) + r(B)$

* quickselect: get rank r element in $O(n)$ work & $\log^2 n$ span, terminates in $O(\log n)$ rounds WHP

! E[size of subtree (of treap)] $\in O(\log n)$, but size of node's subtree $\notin O(\log n)$ WHP — n=2, node chosen as root w.p. $\frac{1}{n} > \frac{1}{n^2}$

Ex1: DP, matrix mult (parens), D = seq of (height, width)

MCP'(i, j) =   cost(a,b,c) ⇝ a□ × b□

  if $j - i \leq 1$ then 0 ← single matrix   call: MCP'(0, n)

  else $\min_{i<k<j}$ (MCP'(i,k) + cost(h(i), h(k), w(j−1)) + MCP'(k,j))

work: $O(n^2)$ subproblems, each $O(1) \cdot (j-i) \in O(n)$ work

span: longest chain $\in O(n)$, subp. of length $k = O(\lg k) \in O(\lg n)$

Ex2: optimal BST ⇔ lowest expected cost $O(n \cdot n^2)$

dp(S, d) = if |S| = 0 then 0 else   call dp(A, 1)

$\min_{1 \leq i \leq |S|}$ (dp(S[1, i−1], d+1) + d p(S[i]) + dp(S[i+1, |S|], d+1))

* union of 2 treaps is $O(m \lg(\frac{n}{m}+1)) \in O(n)$ ← $\ln(x+1) \leq x$ — where m ≤ n

* back edge ⇔ cycle, cross edge ⇔ directed

tree/forward, u anc. v: $s_u < s_v < f_v < f_u$

back, v anc. u: $s_v < s_u < f_u < f_v$ — edge (u,v)

cross: $s_v < f_v < s_u < f_u$ → v visited (& finished) first before u started; else by DFS, would have visited v via (u,v)

* squaring edge weights ⇒ same MST

↳ MST only depends on ordering among edges ⇔ only thing we do w/ edges is compare them

* joinM rebalances ⇔ priorities violated

valid leftist heap

- star contraction preserves $d \leq 2$, connected & acyclic, but not ≤ 3, cyclic
- reduction: shortest superstring (given a set of strings, find shortest string that includes all given strings), TSP (find shortest spanning cycle) → both $\Omega(n!)$
- union(T_1, T_2), $n = |T_1| > m = |T_2|$, $w \in m \log \frac{n}{m}$
- Seq.scan Table.union Table.empty S, n = |S|, m = |table|
  - contraction: $O(nm)$ (2 tables of size m) $(m \leq n)$
  - recursion: $w(\frac{n}{2}, 2m)$ → $(2i+1)^{th}$: union(m, 2im)
  - expansion: $\sum_{i=0}^{n/2} m \log(\frac{2im}{m}) \in O(nm \log n)$ → $n \log^2 n$
- cost of inserting node w/ augfun merge?

  $w(n) = w(\frac{n}{2}) + O(n)$ ⇝ $O(n)$ → only update 1 child
- priorities unique ⇔ unique treap — 2 (equiv.) ways of creating treap.

① begin w/ empty tree, sequentially insert keys in priority order, each new key is a leaf (just need to keep BST inv.)

② run quicksort, create new node every time pivot is chosen

Deletion: find key, set priority to $-\infty$, rotate down leaves, delete

let R_d = # rotations = (# ancestors in T') − (# ancestors in T)

$X_j^i = \begin{cases} 1 & i \text{ anc. of } j \text{ in } T \\ 0 & \text{o.w.} \end{cases}$, $Y_j^i = \begin{cases} 1 & \ldots T' \\ 0 & \text{o.w.} \end{cases}$, $E[X_d^i] = \frac{1}{|i-d|+1}$

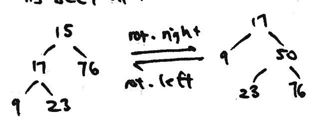
$E[Y_d^i] = \begin{cases} 1 & \text{if } i = d \text{ (node is its own anc.)} \\ 1/|i-d| & \text{o.w.} \end{cases}$ → idea: $p(d) = -\infty$, no chance of higher p

$E[R_d] = \sum_{i=0}^{n-1} E[Y_d^i] - \sum_{i=0}^{n-1} E[X_d^i] = \left(\sum_{i=0}^{d-1} \frac{1}{d-i} + \cancel{E[X_d^d]}^{1} + \sum_{i=d+1}^{n-1} \frac{1}{i-d}\right)$

$- \left(\sum \frac{1}{d-i+1} + E[X_d^d] + \sum \frac{1}{i-d+1}\right)$ deletion still log n! bc. finding

$= H_d + H_{n-d-1} - H_{d-1} + 1 - H_{n-d} + 1 = \frac{1}{d} - \frac{1}{n-d} + 2 \leq 2$ rotations

SCC: if we contract each SCC into single vertex, we get DAG; lemma: if u is the first vertex to be visited in its SCC, all vertices reachable from u finishes b4 u.



split(T, k) ∈ log|T|

table/set/etc via BST: for filter, map, tabulate,

$w \in O\left(\sum_{(k,v) \in T} w(f(v))\right)$, $s \in O\left(\lg|T| + \max_{(k,v) \in T} s(f(v))\right)$

sets: filterKey (Key.t → bool) → set → set

ordsets: supports first (least), last, prev/next S k

split(S, k) = (ℓ, m, r), m ← true if k ∈ S

join(a, b) = (a ∪ b) if ∀a < ∀b, else order

getRange S(x, y) = $\{k \in S \mid x \leq k \leq y\}$

rank(S, k) = $|\{k' \in S \mid k' < k\}|$, splitRank(S, i) = (ℓ, r) — i smallest elements; |r| = |S| − i

select(S, i) = i^{th} smallest element.

Table: insertWith f(t, (k, v)) = t ∪ {(k → v)} if k ∉ t ↳ else → f(v, v')

tabulate f set = {k → f(k) | k ∈ S}

filterKey p t = table of (k, v) ∈ t satisfying p(k, v)

mapKey, iterate, iteratePrefixes. → Seq.iterate f b (range t) → reduceKey avail. for sets.

Ordtable: first, last, prev, next, split(t, k) = (ℓ, v or NONE, r)

Augtable: domain T = set of all keys, range T = seq. of vals.

reduceVal, f must be associative!