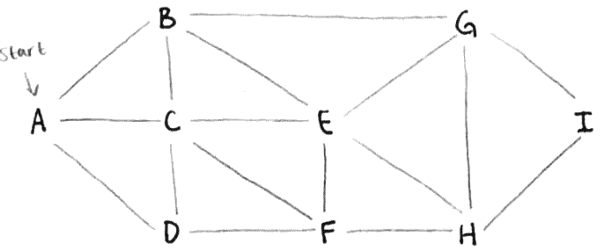
GRAPH TRAVERSALS



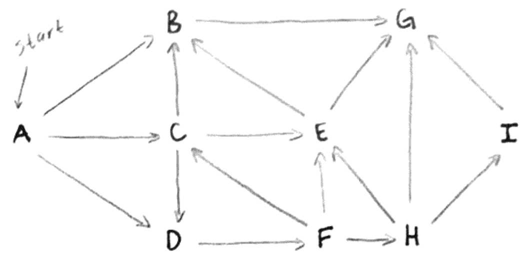
BFS QUEUE

A
B C D
C D E G
D E G F
E G F H
G F H
F H
H
I

ORDER VISITED: A B C D E G F H I

BFS TRIVIA:

- CAN BE USED TO FIND SHORTEST PATH
- $\Theta(V+E)$ RUNTIME, $\Theta(V)$ SPACE



DFS STACK

A B G
A B
A C D F E
A C D F H I
A C D F H
A C D F
A C D
A C
A

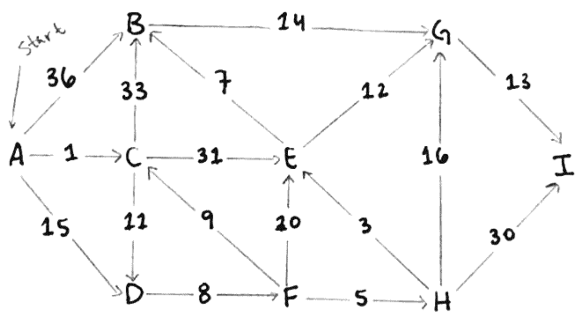
PREORDER: A B G C D F E H I

POSTORDER: G B E I H F D C A

DFS TRIVIA:

- CAN USE REVERSE DFS POSTORDER FOR TOPOLOGICAL SORT
- $\Theta(V+E)$ RUNTIME, $\Theta(V)$ SPACE

SHORTEST PATHS



DIJKSTRA'S TRIVIA:

- ONLY WORKS W/ NON-NEG WEIGHTS
- RUNTIME $\Theta(E \log V)$ (ASSUMING $E \geq V$), $\Theta(V)$ SPACE

DIJKSTRA'S

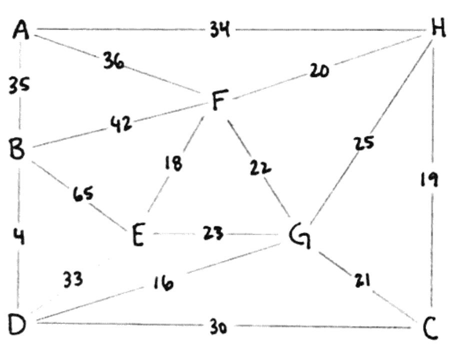
v	A	B	C	D	E	F	G	H	I
distTo[]	0	~~36~~ 34	1	~~15~~ 12	~~32~~ ~~36~~ 28	20	~~41~~ 40	25	~~55~~ 53
edgeTo[]	null	~~A~~ C	A	~~A~~ C	~~C~~ ~~F~~ H	D	~~H~~ E	F	~~H~~ G

ORDER VISITED: A C D F H E B G I

PROCESS: VISIT VERTICES IN ORDER OF BEST KNOWN DISTANCE TO START, RELAXING (ADDING TO SPT IF BETTER) EACH EDGE FROM THE VISITED VERTEX

- A* RUNTIME DEPENDS ON HEURISTIC, $\Theta(V)$ SPACE

MINIMUM SPANNING TREES



PRIM'S ALGORITHM: STARTING FROM ANY ARBITRARY SOURCE, REPEATEDLY ADD THE SHORTEST EDGE THAT CONNECTS SOME VERTEX IN THE TREE TO ONE OUTSIDE IT.

ORDER ADDED: A-H, H-C, H-F, F-E, C-G, G-D, D-B

KRUSKAL'S ALGORITHM: CONSIDER EACH EDGE IN INCREASING ORDER OF WEIGHT AND ADD IT TO THE MST IF IT DOES NOT CREATE A CYCLE.

ORDER ADDED: B-D, D-G, E-F, H-C, F-H, G-C, H-A

CUT PROPERTY: IF YOU DIVIDE THE VERTICES INTO TWO SETS, THEN THE MIN EDGE THAT CROSSES BETWEEN THEM IS IN THE MST.

TRIVIA: MST NOT NECESSARILY SPT FOR ANY PARTICULAR VERTEX. MST V-1 EDGES. PRIM'S RUNTIME $\Theta(E \log V)$. KRUSKAL'S RUNTIME $\Theta(E \log E)$ IF EDGES UNSORTED, ELSE IT'S IN $\Theta(E \log V)$. MSTs ARE NOT ALWAYS UNIQUE SO PRIM'S AND KRUSKAL'S CAN PRODUCE DIFFERENT MSTs.

DYNAMIC PROGRAMMING

DEFINITION: THE PROCESS OF IDENTIFYING A COLLECTION OF SUBPROBLEMS, SOLVING THEM FROM SMALLEST TO LARGEST, USING THE SMALLER PROBLEMS TO SOLVE THE LARGER.

AN APPLICATION: FINDING THE SPT OF A DIRECTED ACYCLIC GRAPH FASTER THAN DIJKSTRA'S ($\Theta(E+V)$), THE DAGSPT IS AN EXAMPLE OF DYNAMIC PROGRAMMING. FIND TOP. ORDERING & RELAX IN THAT ORDER. WORKS WITH NEGATIVE EDGES. DYN. PROG. B/C SOLVE DIST FROM S TO S, THEN USE RESULTS FOR OTHER V.

COMPLEXITY CASES

P CLASS:

- DECISION PROBLEM (A YES OR NO PROBLEM)
- AN ANSWER CAN BE FOUND IN $\exists_k\ O(N^k)$ TIME
- EX: ARE THERE TWO ITEMS IN AN ARRAY WHOSE SUM IS ZERO?
- CLOSED UNDER + AND ∘

NP CLASS:

- DECISION PROBLEM
- A "YES" ANSWER CAN BE VERIFIED IN $O(N^k)$ TIME FOR SOME K
- EX: IS THERE AN IND. SET OF SIZE K? TO VERIFY, CHECK THAT ALL VERT ADJ TO SOME SET ARE NOT THE SAME COLOR AS IN THE SET

NP-COMPLETE CLASS

- A PROBLEM IS IN NP-COMPLETE IF
  - IT IS IN NP
  - IT CRACKS ALL OTHER PROBLEM IN NP
- EX: 3SAT: DOES THERE EXIST A TRUTH TABLE FOR BOOLEANS THAT OBEYS A SET OF 3-VAR DISJUNCTIVE CONSTRAINTS?

SORTING

	PROCESS	STABLE?	MEMORY	BEST RT	WORST RT	NOTES
SELECTION SORT	REPEATEDLY IDENTIFY THE MAX ELEMENT AND MOVE TO THE END.		$\Theta(N)$	$\Theta(N^2)$	$\Theta(N^2)$	
INSERTION SORT	SWAPS ITEMS 1-BY-1 TOWARDS THE LEFT UNTIL THEY LAND IN RIGHT PLACE LEFT OF i SORTED.	YES	$\Theta(1)$	$O(N)$ $\Theta(N)$ INVERSIONS OR $N < 15$	$\Theta(N^2)$	
HEAPSORT	HEAPIFY FROM BOTTOM RIGHT. REPEATEDLY DELETE THE MAX ITEM, SWAPPING IT WITH LAST IN HEAP. PLACE MAX AT THE END INTO THE SORTED PART OF LIST.	NO	$\Theta(1)$	$\Theta(N \log N)$	$\Theta(N \log N)$	
MERGE-SORT	REPEATEDLY SPLIT ITEMS INTO TWO ROUGHLY EVEN PIECES AND RECURSIVELY MERGE-SORT THEM.	YES	$\Theta(N)$	$\Theta(N \log N)$	$\Theta(N \log N)$	
QUICK-SORT	PARTION ON SOME PIVOT & QUICKSORT ON BOTH SIDES OF PIVOT.	DEPENDS ON PARTITIONS	$\Theta(\log N)$	$\Theta(N \log N)$	$O(N^2)$ SORTED ARRAY VERY IMPROB.	
LSD RADIX SORT	SORT DIGIT-BY-DIGIT FROM RIGHT TO LEFT WITHOUT BUCKETS. RELYS ON STABILITY	YES	$\Theta(N+R)$	$\Theta(WN+WR)$	$\Theta(WN+WR)$	R: SIZE OF ALPHABET W: WIDTH OF LONGEST KEY
MSD RADIX SORT	WORK FROM LEFT TO RIGHT SOLVING EACH SUBPROBLEM INDEPENDENTLY	YES	$\Theta(N+WR)$	$\Theta(N+R)$ DISTINCT 1ST CHARS	$\Theta(WN+WR)$	

PARTITIONING

3-WAY PARTITIONING: PUT SMALLER THINGS IN AN ARRAY, EQUAL THINGS IN AN ARRAY, LARGER THINGS IN AN ARRAY → MERGE

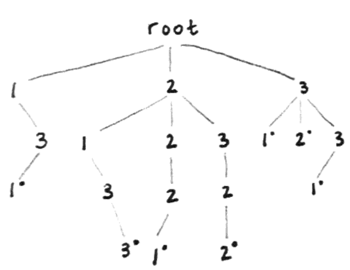
HOARE PARTITIONING: LEFT PTR LOVES SMALL ITEMS, RIGHT PTR LOVES LARGE THINGS. STOP AT SOMETHING THEY DON'T LIKE AND SWAP WHEN BOTH HAVE STOPPED. END RESULT IS THAT THINGS < PIVOT ON LEFT, == PIVOT IN BETWEEN, > PIVOT ON RIGHT.

SHUFFLING: ASSIGN A RANDOM FLOAT TO EVERY OBJECT, SORT ON THAT

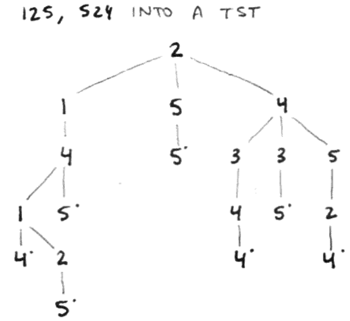
OPTIMIZING SORTS: CAN SWITCH TO INSERTION SORT IF $N < 15$. EXPLOIT EXISTING ORDER (CALLED "ADAPTIVE" SORTING) LIKE TIMSORT. FOR WORST CASE $\Theta(N^2)$ SORTS, SWITCH TO $N \log N$ SORT IF THEY DETECT THAT THEY HAVE EXCEEDED A REASONABLE NUMBER OF OPERATIONS.

TRIES

INSERT **32, 2133, 2221, 31, 131, 331, 2322** INTO A R=3 MULTIWAY TRIE (• ENDS A WORD)



INSERT **255, 435, 344, 115, 114, 125, 524** INTO A TST



R-WAY TRIE PARALLELS LSD

TSTs PARALLEL LLRBs.

RUNTIME FOR contains()

	WORST	BEST(MISS)	MEMORY
HASH TABLE	$\Theta(L)$ AMORTIZED		$\Theta(NL)$
BST	$\Theta(L \log N)$	$\Theta(1)$	$\Theta(NL)$
TRIE (ARRAY MAP)	$\Theta(L)$	$\Theta(1)$	$\Theta(NLR)$
TRIE (TREEMAP)	$\Theta(L \log R)$	$\Theta(1)$	$\Theta(NL)$
TST	$\Theta(NL)$	$\Theta(1)$	$\Theta(NL)$
TRIE (HASH MAP)	$\Theta(L)$	$\Theta(1)$	$\Theta(NL)$

N KEYS, L DIGITS PER KEY, R ALPHABET SIZE.