STATIC CS61B

- belongs to class instead of specific instance
- watch out for random instances of static being used.

Two different references
→ no reassignment means forever different objects

static type ↓ Dynamic type ↙

Dog Fido = new Beagle

method call is dynamic type.
instance variables are static type

All methods called by object must be present in static type.

super (x, y, z) in constructor

Reverse
```
for (int i=0; i<x.length/2; i++) {
    int j= x.length - i -1;
    int temp = x[i];
    x[i] = x[j];
    x[j]= temp;
}
```

2D Array
int [] []
↑ row ↑ col
Access row before col

String	Array	61B
.length()	.length	

```
class Dog {
    public String className;
    public Dog() {
        className = "dog";
    }
    public String getClassName() {
        return className;
    }
}
class Beagle extends Dog {
    public String className;
    public Beagle() {
        super();
        className = "beagle";
    }
}
class Chihuahua extends Dog {
    public String className;
    public Chihuahua() {
        super();
        className = "chihuahua";
    }
    @Override
    public String getClassName() {
        return className;
    }
}
```

```
Dog d= new Chihuahua();
Sop (d.getClassName());
>>> Chihuahua

Dog d = new Beagle();
SOP (d.className);
>>> dog.

Beagle d= new Beagle();
SOP (d.getClassName());
>>> dog
```

↳ no overriden method, inherits getClassName() from Dog
also takes Dog's className.

JUnit

assertEquals (x,y)

assertNotEquals (x,y)

assertTrue (boolean)

(a[i] < a[i+1])

Circular

(k + i) % message.length

(first + index) % items.length

```
IntNode p = sentinel;
while ( p != null && p.next != null) {
    if (p.next.item < 0) {
        p.next = p.next.next; //skips node
    }
    p = p.next;
}
```

61B

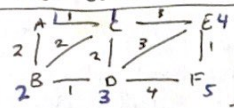
Sorting

N: # items | R: # unique items | W - length of largest item

	B	W	Space	St?
Selection	n^2	n^2	1	Y
Insertion	n (sorted, few inversions)	n^2 (backwards)	1	Y
Heap	dup n; same elements dist. NlogN	NlogN (bubble down on all removals)	1	N (max heap - heapify - del max)
Merge	NlogN	NlogN	N	Y
Quick	dup N; random pivot / shuffle dist NlogN	n^2 (smallest / biggest pivot)	logN	N (in place), Y (3-way)
Counting	N+R	N+R	N+R	Y
LSD	WN+WR	WN+WR	N+R	Y
MSD	N+R	WN+WR	N+WR	Y

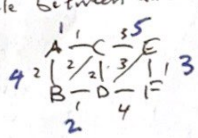
MSD: recurse within each bucket

MST
- one unique path between 2 vertices



Prim
- start w/ vertex {set}
- find min edge that connects something in set w/ something outside
- add edge to set $O(E\log V)$

Kruskal
- sort edges into PQ: priority = edge weight
- pop min edge if edge doesn't create a cycle between already popped edges



$O(E\log V)$ WQU
$O(E\log E)$ PQ
→ time: sorting, disjoint set management

WQU to track cycle: isConnected (potential edge)

Dijkstra - distance from start - SPT to all vertices

A^* - distance + heuristic. - SPT to one target

Iterator → hasNext() → next()

Trie

	contains best	contains worst
HS (Hash Set)	ℓ	ℓ
Trie	1	ℓ
TST	1	Rℓ

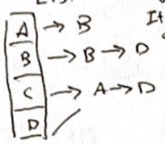
Naïve TrieNode
- char c;
- TrieNode[]
- boolean isWord

TST
- char c;
- 3 links: next char < curr → left; = → down; > → right

Graph

Adj Matrix vs List

	A	B	C	D
A	0	1	0	0
B	0	1	0	1
C	1	0	0	1
D	0	0	0	0



	List	Matrix
Edge Lookup	V	1
Iterate all edges	V+E	V^2
Neighbors	V	V
Space	V+E	V^2

$N! \in \Omega((\frac{N}{2})(\frac{N}{2}))$
$\log(N!) \in \Omega(N\log N)$
$N\log N \in \Omega(\log(N!))$
$\log(N!) \in \Theta(N\log N)$

Radix > Comparison — alphabet is well-defined and each object can be individually compared on each radix

→ Highly dissimilar strings - MergeSort is faster → won't look at most characters
Similar - LSD → comparison takes W time

			Runtime if (E>V)
Shortest Paths	Dijkstra	$O(E\log V)$	~Dijkstra's
MST	Prim	$O(E\log V)$	
MST	Kruskal	$O(E\log E)$	WQUPC
MST	K w/ pre-sorted edges	$O(E\log^* V)$	WQUPC

Prim	# times	Time per Op	Total Time
Insert	V	$O(\log V)$	$O(V\log V)$
Del Min	V	$O(\log V)$	$O(V\log V)$
~~Dec Pri~~ Decrease priority	E	$O(\log V)$	$O(E\log V)$

Kruskal	# times	Time per Op	Total Time
Build PQ edges	1	O(E)	O(E)
Del min	E	$O(\log E)$	$O(E\log E)$
connect	V	$O(\log^* V)$	$O(V\log^* V)$
isConnected	E	$O(\log^* V)$	$O(E\log^* V)$

Sorting
Merge → groups of numbers sorted

sorted except last x numbers → insertion

last x max elements in place → heap.

First element in place (pivot) → QuickSort

QuickSort, MergeSort, Selection - $\Omega(N\log N)$
Insertion, Heap - $\Omega(n)$

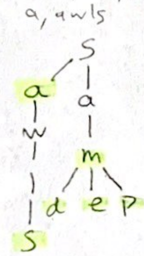
QuickSort, Merge, Insertion never compares same two elements twice.

Only selection sort performs $\Theta(N)$ swaps of elements.

QuickSort > MergeSort
- faster runtime
- better memory
- stability doesn't matter
- have duplicates

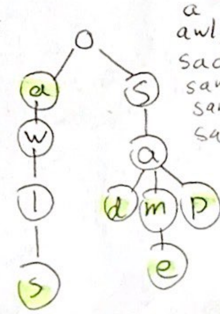
Trie > HashMap when:
- near misses
- string prefix ops
- mismatches early in key
- support ordered operations

TST
insert: sam, sad, sap, same, a, awls



Tries

a
awls
sad
sam
same
sap



Runtime Trie

	Insert (W)	Search (W)	Search (13)
Trie	$\Theta(\ell)$	$\Theta(\ell)$	$\Theta(1)$
HashTable		$\Theta(N\ell)$	$\Theta(\ell)$

$\Theta(\ell)$ (average)

Heaps use less memory than LLRB, handles duplicates better

selection: first x elements in final order; insertion: first x sorted, but may not be in final order ~ some shit yet to be sorted

- Inserting a single item into a bushy BST w/ N items takes Θ(log N) time in all cases.
- Height of BST w/ N items = O(N)
- All LLRB are BSTs
- Not all WQU trees are BSTs
- parent of parent of 3rd largest item is not always root
- height of perfectly balanced quadtree w/ N items is asymptotically same as height of 2-3 tree w/ N items
- Dijkstra's algorithm doesn't always find the shortest path in a directed acyclic graph, if there are negative edges.
- SPT may not have total weight ≤ MST
- last edge added to MST may not be highest weight edge.
- largest edge could be part of SPT
- DFS/BFS could visit in same order.

With equals you need hashCode method

Heapify - from bottom up, sinking nodes to correct position.

- replace (sink/bubble down) w/ choice of higher priority

```
      1                     9
  2       4               3   7
9   3   7   5    -->    2  1  4  5
```

Completeness property - no holes in heap as you go thru level-order traversal
At leaf level, all leaves pushed to left.

Heap-Order: each node has higher priority than children.

Get elements in trie.

```
private void collect(Node x, List<Integer> matches, int topDigits)
    if x == null, return;
    if x.exists, matches.append(topDigits *10 + x.dig)
    for (Node c : x.children){
        collect(c, matches, topDigits)
    }
```

SPT

To make incorrect, need to find an edge in the graph where neighboring node of node being visited doesn't get considered because it was already visited.

→ w/ all positive edge weights, should be able to disregard.

Compression

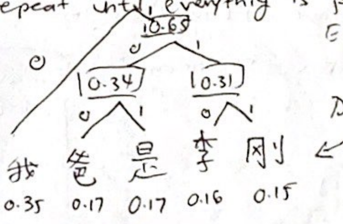
Shannon-Fano Coding

- count relative frequencies of all characters in text
- split into left, right half of roughly equal frequency

Symbol	Frequency	S-F Code
我	0.35	00
爸	0.17	01
是	0.17	10
李	0.16	110
刚	0.15	111

Huffman

- Assign each symbol to a node w/ weight = relative frequency
- take two smallest nodes and merge them into a super node with weight equal to sum of weights
- repeat until everything is part of a tree



Encoding: Map<Char, BitSequence>
Array (index = char value)

Decoding - trie

Compression - pass in decoding tree, then sequence of bits following encoding map/array

Decompression - use codeword bits to walk down trie, outputting symbols every time you reach a leaf.

↳ longest prefix. e.g. 00111111 → 我我刚刚

$1, 2, 3, 4, 5, \ldots, (N-1), N \to \Theta(N^2)$

N terms

$1, 2, 4, 8, 16, \ldots, (\frac{N}{2}), N \to \Theta(N)$

log N terms.

to make obj iterable, implement Iterable<T>

↳ method

Iterator<T> iterator()

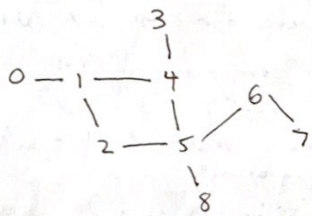
hasNext(), next(), remove()

Generics; Class<T> extends HashMap<T>

Balanced BST

	BST B	BST W	RB B	RB W
Find	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(\log n)$
Insert	$\Theta(1)$	$\Theta(N)$	$\Theta(\log n)$	$\Theta(\log n)$
Delete	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(\log n)$

Graphs DFS - return when no more children (unmarked)



DFS calls: 0 1 2 5 4 3 6 7 8

DFS returns: 3 4 7 6 8 5 2 1 0

DFS Preorder - calls

DFS Postorder - returns   Stack

Level-order - order of increasing distance from src

0 1 2 4 5 3 6 8 7   → BFS Queue

Topological Sort (directed, acyclic graph)

All edges in one direction. = reverse postorder.

BST

Insert: create, set link

Delete: - 1 child

→ remove node

→ move child up

- 2 children

→ rightmost of left subtree ||

→ leftmost of right subtree

2-3(-4) tree

insert: $\Theta(H)$

$= \Theta(\log N)$

swim-up - swap

Sink - swap w/ higher priority child

MinPQ - child w/ smaller value.

Heaps

Spot 0 empty

leftChild: k*2

rightChild: k*2+1

parent: k/2

61B

Depth First Traversal - Trees

Pre-Order: visit node, then traverse children

In-Order: left child, node, ~~tra~~ right child

Post-Order: traverse children, visit Node

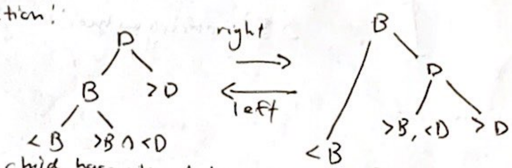
Level: by "level" of tree

2-3-4: ≤ 3 keys per node, any non-leaf node has one more child than # keys

Red-Black tree: (root is black, each red node has ≤ 2 black children)

2-3: ≤ 2 keys per node, any non-leaf has one more child than # keys

Tree Rotation:



child has no knowledge of parent if singly linked

LLRB:

- no node has 2 red links
- every path from root to leaf has same number of black links.
- red links lean left
- black links connect 2-3 nodes in 2-3 tree.

LLRB : 2-3 :: RB : 2-3-4

2-3: once a leaf has 3 or more values, middle goes up, other 2 split.

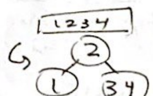
2-3-4: once a leaf has 4 or more values, middle left goes up; left, right 2 split.

x-node: x = # children

reach 3 node, split

reach 4 node, split.

dark are own roots



Splitting trees maintain balance.

```
public static void p3(int N) {
    if (N <= 1) return;
    p3(N/2);
    p3(N/2);
```

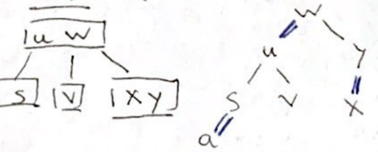
log N levels

2^N nodes

$\Theta(N)$

Iterator: hasNext(), next()

2-3 → LLRB



Binary Heap is perfectly balanced.

HashMap Key must be immutable!

Program halts after exception is thrown but not caught. Exception not printed.

Regex

.	- wildcard
[A-Z a-z]	- upper/lowercase letters
+	- at least one
*	- 0 or more
{a, b}	- a ≤ _ ≤ b occurences
\	- escape
?	- 0 or 1
^ ...... $	

\d - space
\n - line break
\w - any word
\[

1) parenthesis ( )
2) *
4 ~~3~~) or "|"
3 ~~4~~) concatenation.

Performance

	Construct	Connect	isConnected
QuickFindDS	$\Theta(N)$	$\Theta(N)$	$\Theta(1)$
QuickUnionDS	$\Theta(N)$	$O(N)$	$O(N)$
WQUDS	$\Theta(N)$	$O(\log N)$	$O(\log N)$

Bushy BST search : $\Theta(\log N)$

BSTMap - best case : bushy
worst : LL

	contains	insert
LL	$\Theta(N)$	$\Theta(N)$
Bushy BST	$\Theta(\log N)$	$\Theta(\log N)$
Unordered Array	$\Theta(N)$	$\Theta(N)$
DataIndexed	$\Theta(1)$	$\Theta(1)$
External Chaining	$\Theta(Q)$	$\Theta(Q)$ → length of longest list
Hash Table	$\Theta(1)$	$\Theta(1)$

Heap Implementation of PQ

	Ordered [ ]	Bushy BST	HT	Heap
add	$\Theta(N)$	$\Theta(\log N)$	$\Theta(1)$	$\Theta(\log N)$
getSmallest	$\Theta(1)$	$\Theta(\log N)$	$\Theta(N)$	$\Theta(1)$
removeSmallest	$\Theta(N)$	$\Theta(\log N)$	$\Theta(N)$	$\Theta(\log N)$

---

can take $\Theta(n^2)$ time to insert N items
- LL, BSTMap, External Chaining HM, AL

NOT 2,3 Tree Set, Heap Min PQ, LLRBST Set

	Access	Search	Insertion	Deletion
Array	$O(1)$	$O(N)$	$O(N)$	$O(N)$
Stack	$O(N)$	$O(N)$	$O(1)$	$O(1)$
Queue	$O(N)$	$O(N)$	$O(1)$	$O(1)$
SLL	$O(N)$	$O(N)$	$O(1)$	$O(1)$
DLL	$O(N)$	$O(N)$	$O(1)$	$O(1)$
HT	—	$O(N)$	$O(N)$	$O(N)$
BST	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
RBT	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$

BFS $\Theta(V+E)$
DFS $\Theta(V+E)$
Top Sort $\Theta(V+E)$

Disjoint Sets

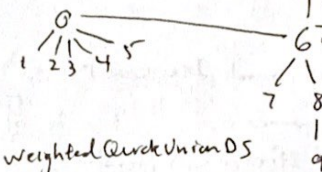
isConnected( )
Connect( )

parent

0	0	0	0	0	0	0	6	6	8
0	1	2	3	4	5	6	7	8	9

size

10	1	1	1	1	1	4	1	2	1
0	1	2	3	4	5	6	7	8	9



weightedQuickUnionDS
↳ connect root of smaller tree to bigger tree
↳ track tree size
Best Case Height: 1
Worst Case: log N.

TreeSet - tree set that is sorted by natural order

TreeMap - sorted map

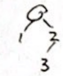
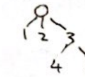
QuickFind - p and q are connected iff id[p] = id[q]
- all sites in component must have same value in id[ ]

QuickUnion - id[ ] entry is name of another site in same component
Find( ) - follow links to another site until reaching root (site that self links)
Union( ) - find roots, rename one component by linking one root to other.

WeightedQU - link smaller tree to larger
- minimizes height

Hash
- deterministic, good distribution, two objects equal( ) must have same hashcode

Worst Case Height : union of two sets of size $\frac{N}{2}$ that have worst case height.

N = 4:  N = 6: 

N ... R(N)
$R(N) = a N^b$
Take 2 points, solve system

[ ]
[wolf, 12, cat, 41]
[dog, 9001, cat]

\[((([a-z]+, *\d+, *)*[a-z]+(, *[0-9]+)?)?\]