CS 600.226: Data Structures Michael Schatz

Oct 3 2018 Lecture 16. More Graphs



Agenda

- I. Questions on HW4
- 2. Recap on Trees
- 3. Graphs

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment04/README.md

Assignment 4: Stacking Queues

Out on: September 28, 2018 Due by: October 5, 2018 before 10:00 pm Collaboration: None Grading:

Packaging 10%, Style 10% (where applicable), Testing 10% (where applicable), Performance 10% (where applicable), Functionality 60% (where applicable)

Overview

The fourth assignment is mostly about stacks and dequeues. For the former you'll build a simple calculator application, for the latter you'll implement the data structure in a way that satisfies certain performance characteristics (in addition to the usual correctness properties).

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment04/README.md

Problem 1: Calculating Stacks (50%)

Your first task is to implement a basic RPN calculator that supports integer operands like 1, 64738, and -42 as well as the (binary) integer operators +, -, *, /, and %. Your program should be called Calc and work as follows:

- You create an empty Stack to hold intermediate results and then repeatedly accept input from the user. It doesn't matter whether you use the ArrayStack or the ListStack we provide, what does matter is that those specific types appear only once in your program.
- If the user enters a *valid integer*, you *push* that integer onto the stack.
- If the user enters a *valid operator*, you *pop* two integers off the stack,
 perform the requested operation, and *push* the result back onto the stack.
- If the user enters the symbol ? (that's a question mark), you *print* the current state of the stack using its toString method followed by a new line.
- If the user enters the symbol . (that's a dot or full-stop), you *pop* the top element off the stack and *print* it (only the top element, not the entire stack) followed by a new line.
- If the user enters the symbol ! (that's an exclamation mark or bang), you exit the program.

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment04/README.md

\$ java Calc	\$ java Calc
?	? 10 ? 20 30 ? *
[]	? + ? . !
10	[]
?	[10]
[10]	[30, 20, 10]
20 30	[600, 10]
?	[610]
[30, 20, 10]	610
*	Ş
?	
[600, 10]	
+	
?	
[610]	
•	
010	
: ¢	
Ş	

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment04/README.md

Problem 2: Hacking Growable Dequeues (50%)

Your second task is to implement a generic ArrayDequeue class as outlined in lecture. As is to be expected, ArrayDequeue must implement the Dequeue interface we provided on github.

- Your implementation must be done in terms of an array that grows by doubling as needed. It's up to you whether you want to use a basic Java array or the SimpleArray class you know and love; just in case you prefer the latter, we've once again included it on the github directory for this assignment. Your initial array must have a length of one slot only! (Trust us, that's going to make debugging the "doubling" part a lot easier.)
- Your implementation must support all Dequeue operations except insertion in (worst-case) constant time; insertion can take longer every now and then (when you need to grow the array), but overall all insertion operations must be constant amortized time as discussed in lecture.
- You should provide a toString method in addition to the methods required by the Dequeue interface. A new dequeue into which 1, 2, and 3 were inserted using insertBack() should print as [1, 2, 3] while an empty dequeue should print as []

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment04/README.md

Bonus Problem (5 pts)

Develop an **algebraic specification** for the **abstract data type Queue**. Use new, empty, enqueue, dequeue, and front (with the meaning of each as discussed in lecture) as your set of operations. Consider unbounded queues only.

The difficulty is going to be modelling the FIFO (first-in-first-out) behavior accurately. You'll probably need at least one axiom with a case distinction using an if expression; the syntax for this in the Array specification for example.

Doing this problem without resorting to Google may be rather helpful for the upcoming midterm. There's no need to submit the problem, but you can submit it if you wish; just include it at the end of your README file.

Agenda

- I. Questions on HW4
- 2. Recap on Trees
- 3. Graphs





Level Order Traversals





More to come...

13 Sets, Iterators, Performance Analysis 13.1 Rewriting Unique 13.2 Basic Performance Measurements 13.3 Array-based versus List-based Sets 13.4 Advanced Performance Measurement: Profilers 13.5 Self-Organizing Sets	103 106 107 109 111 114
14 Ordered Sets, Heaps 14.1 Binary Search, Including Proof Outline 14.2 Heaps and Priority Queues	117 120 123
15 Maps 15.1 Binary Search Trees	129 131
16 Balanced Search Trees 16.1 Random Insertions 16.2 2-3 Trees 16.3 AVL Trees	140 140 140 142
17 Hash Tables 17.1 Collisions 17.2 Separate Chaining 17.3 Linear Probing 17.4 Quadratic Probing 17.5 Double Hashing 17.6 Hash Functions 17.7 Hash Table Size 17.8 Hacking the Hash Table 17.9 Benchmarks	150 151 152 154 156 156 156 157 159 159 159 163
18 Trees, Hashes, Sorting 18.1 Trees, Recursively. 18.2 Hash Trees (aka Merkle trees) 18.3 Sorting, Lower Bound. 18.4 Heap Sort 18.5 Merge Sort 18.6 Quick Sort	166 166 167 169 171 173
19 Bit Sets, Splay Trees, Treaps, Bloom Filters 19.1 Sets of Integers 19.2 Bit Sets	177 177 179

Inheritance

Why Inheritance?

Code Reuse

• Subclass gets to use all methods of the parent class "for free" by extending the parent class

Overriding

• Subclass can have more specific implementation than the parent class

Design constraints

• Subclasses get all of the features of the parent, whether you like them or not!

Saying B inherits from A is a very strong relationship: anytime that A could be used, B could be instead



Inheritance Types



Multilevel Inheritance



Multiple Inheritance



B isa A

Square isa Rectangle C isa B, B isa A

Square isa Rectangle Rectangle isa Shape => Square isa Shape C isa A, C isa B

Mike isa CS Prof Mike isa Bio Prof

(not in Java!)

Inheritance versus Encapsulation



When to use static?





Graphs are Everywhere!



Computers in a network, Friends on Facebook, Roads & Cities on GoogleMaps, Webpages on Internet, Cells in your body, ...

Graphs



- Nodes aka vertices
 - People, Proteins, Cities, Genes, Neurons, Sequences, Numbers, ...
- Edges aka arcs
 - A is connected to B
 - A is related to B
 - A regulates B
 - A precedes B
 - A interacts with B
 - A activates B

Graph Types





Definitions (2)



- A *path* is a sequence of edges e₁, e₂, ... e_n in which each edge starts from the vertex the previous edge ended at
 - A path that starts and ends at the same node is a *cycle*
 - The number of edges in a path is called the *length* of the path
 - A graph is *connected* if there is a path between every pair of nodes, otherwise it is *disconnected* into >1 *connected components*

Network Characteristics

	C. elegans	D. melanogaster	S. cerevisiae
# Nodes	2646	7464	4965
# Edges	4037	22831	17536
Avg. / Max Degree	3.0 / 187	6.1 / 178	7.0 / 283
# Components	109	66	32
Largest Component	2386	7335	4906
Diameter	14	12	11
Avg. Shortest Path	4.8	4.4	4.1
Degree Distributions	100 100 100 100 100 100 100 100	soon and a soon and a	100 100 100 100 100 100 100 100 100 100

Diameter: Maximum length of shortest path between two nodes Scale Free: Power law distribution of degree => Avg. Shortest Path between nodes is small (Most people have ~100 twitter followers, but some have >1M)

Network Characteristics



Diameter: Maximum length of shortest path between two nodes Scale Free: Power law distribution of degree => Avg. Shortest Path between nodes is small (Most people have ~100 twitter followers, but some have >1M)

Network Motifs

Network Motif

- Simple graph of connections
- Exhaustively enumerate all possible 1, 2, 3, ... k node motifs
- Statistical Significance
 - Compare frequency of a particular network motif in a real network as compared to a randomized network
- Certain motifs are "characteristic features" of the network

Network	Nodes	Edges	Nreal	Nrand 2 SD	Z score	Nreal	Nrand ± SD	Z score	Nreal	Nrand 2 SD	Z score
Gene regulat (transcriptio	tion n)			X Y Y	Feed- forward loop	X X	Å,	Bi-fan			
E. coli 5. cerevisiae*	424 685	519 1,052	40 70	7±3 11±4	30 14	203 1812	47±12 300±40	13 41			
Neurons				х У У	Feed- forward loop	N N	Å.	Bi-fan	K K	K ^X K ^Z	Bi- parallel
C elegant	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs				× ¥ ¥	Three chain	K, K	××××	Bi- parallel			
Little Rock Ythan St. Martin Chesapeake	92 83 42 31	984 391 205 67	3219 1182 469 80	2 3120 ± 50 1020 ± 20 450 ± 10 82 ± 4	2.1 7.2 NS NS	7295 1357 382 26	2220 ± 210 230 ± 50 130 ± 20 5 ± 2	25 23 12 8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5	I		
Skipwith B. Brook	25	189	184	150 ± 7	5.5	397	80±25	13	I		
Electronic ch	6.7	104	101	N N	Fired.	- 207 X	N N	Pic free	<u> </u>	x	10
(forward logi	k chips)			¥ ¥	forward loop	↓ ×	Ś.		K, K	w ^z	parallel
s15850	10,383	14,240	424	2 ± 2	285	1040	1 ± 1	1200	480	2 ± 1	335
x38584	20,717	34,204	413	10 ± 3	120	1739	6±2	800	711	9 ± 2	320
\$38417	23,843	33,661	612	3 ± 2	400	2404	1 = 1	2550	200	2 = 2	340
+13207	8,651	11,831	405	2 + 1	225	4445	1+1	4950	264	2 + 1	200
Electronic ci	incults	11000		6	Three-	x	v	Bi-fan	x-	->v	Four-
(digital fract	ional mult	(pliers)	1	- z	node feedback loop	1 x	S.		Ţ.	<u> </u>	node feedback loop
s208	122	189	10	1 = 1	9	4	1 ± 1	3.8	5	1 = 1	5
s420	252	399	20	1 ± 1	18	10	1 ± 1	10	11	1 ± 1	11
s838‡	512	819	40	1 ± 1	38	22	1±1	20	23	1±1	25
World Wide	Web			NOTON	Feedback with two mutual dyads	NY	Sz ≮	Fully connected triad	14	×z €	Uplinked mutual dyad
ad a day	325,729	1.45e6	1.1e5	2e3 + 1e2	800	6.8+6	Sededa2	15,000	1.24	1e4 ± 2e	5000
	Contract State	ALC: 1.1.1		200.00		1		a series of the	1.10.27	101100000	

Network Motifs: Simple Building Blocks of Complex Networks Milo et al (2002) Science. 298:824-827

public interface Graph<V,E> {

Position<V> insertVertex(V v);

Position<E> insertEdge(Position<V> from, Position<V> to, E e)
throws InvalidPositionException, InsertionException;

V removeVertex(Position<V> p)
 throws InvalidPositionException, RemovalException;

E removeEdge(Position<E> p)
+

throws InvalidPositionException;

Separate generic

types for vertices <V>

and edges <E>

Iterable<Position<V>> vertices();

Iterable<Position<E>> edges();

Representing Graphs



Incidence List Good for sparse graphs Compact storage A: C, D, E D: F B: D, E E: F C: F, G G:



Note the labels in the edges are really references to the corresponding node objects!

Representing Graphs



Incidence List Good for sparse graphs Compact storage

A: C, D, E	D: F
B: D, E	E: F
C: F, G	G:

Complexity Analysis

If n is the number of vertices, and m is the number of edges, we need O(n + m) space to represent the graph

When we insert a vertex, allocate one object and two empty edge lists: O(1)

When we insert an edge we allocate one object and insert the edge into appropriate lists for the incident vertices: O(1)

Remove a node?	O(1); Only after edges removed
Remove an edge?	O(d) where d is max degree; O(n) worse case
Find/check edge between nodes?	O(d); O(n) worst case

Representing Graphs



<u>Tools</u>

Graphviz: <u>http://www.graphviz.org/</u> Gephi: <u>https://gephi.org/</u> Cytoscape: <u>http://www.cytoscape.org/</u> digraph G { A->B B->C A->C } \$ dot -Tpdf -o g.pdf g.dot



public interface Graph<V,E> {

Position<V> insertVertex(V v);

Position<E> insertEdge(Position<V> from, Position<V> to, E e)
throws InvalidPositionException, InsertionException;

V removeVertex(Position<V> p)
 throws InvalidPositionException, RemovalException;

E removeEdge(Position<E> p)
+

throws InvalidPositionException;

Separate generic types for vertices <V>

and edges <E>

Iterable<Position<V>> vertices();

Iterable<Position<E>> edges();

public interface Graph<V,E> {

Position<V> insertVertex(V v);

Position<E> insertEdge(Position<V> from, Position<V> to, E e)
throws InvalidPositionException, InsertionException;

V removeVertex(Position<V> p)
 throws InvalidPositionException, RemovalException;

E removeEdge(Position<E> p)

throws InvalidPositionException;

Separate generic types for vertices <V>

and edges <E>

Iterable<Position<V>> vertices();

Iterable<Position<E>> edges();

Iterable

📴 JHJMal 🏥 (EDaily 🔛 🖬 🖉 🔕 🖽 🔂	schatzab 💩 SL (E) cell (E) Pu (E) pu (E) XR (E) Media (E) tool () edit () Remove NYT Cookes 🛊 Booknaks	Et Other Booker
entee Package 🔀 Use Tree Depre	called Index Help	100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100
ev Class Next Class Frames No Frame	n Al Classes	
mmary: Nested Field Constr Method Detail: Fie	ed Gener Method	
java.lang		
Interface Iterable <t></t>		
Type Parameters:		
$\boldsymbol{\tau}$ - the type of elements returned by the iteration	or and a second s	
All Known Subinterfaces:		
BeanContext, BeanContextServices, Blocking	DequerE>, BickingQueuerE>, ColectionrE>, DequerE>, DirectoryStreamrT>, ListrE>, NavigableSetrE>, Path, QueuerE>, SecureDirectory	Stream <t>, Set<e>, SoriedSet<e>, TranslerQueue<e></e></e></e></t>
All Known Implementing Classes:		
AbstractCollection Abstract int AbstractCom	a Deterflamental ist Deterflat AmolitickingCone Areafonic Area ist Mithold ist Bathlindelingerten Bashforterflamine	Surrout BaanContextSurrout Consument Inter/Deput
ConcurrentLinkedQueue, ConcurrentSkipList5	Set, CopyOnWriteAmayList, CopyOnWriteAmaySet, DataTruncation, DelayGueue, EnumSet, HashSet, JobStateReasons, LinkedBlockingDegue,	LinkedBlockingQueue, LinkedHashSet, LinkedList, LinkedTransferQueue,
PriorityBiockingQueue, PriorityQueue, RoleLis SQLInvalidAuthorizationSpecException, SQLI	II. Roschmeiserebust, Rowbernaming, Senatticoption, ServiceLoader, Sul Clentinoticoption, Sul Deatticoption, Sul Exception, Sul Featimeter Sul Clentinoticoption, Sul Deatticoption, Sul Exception, Sul Featimeter Sul Clentinoticoption, Sul Deatticoption, Sul Exception, Sul E	SQLTransactionRolbackException, SQLTransientConnectionException,
PriorityBlockingQueue, PriorityQueue, RoleLa SQLInvalidAuthorizationSpecException, SQL SQLTransientException, SQLWarning, Stack,	e, Rosumesovedue, Rovoennaming, Senetskopton, Servosubeër, SulLikerinockopton, SulLikerikopton, SulLikerikopton, NonTansientConectonitikoepton, SüllikonTansientEkkopton, SüLKevinetketkopton, SüLSyntaxEmorEkkopton, SüLTimeoutEkopton, SyncFactoryEkoeption, SynchronousQueue, SyncProviderEkoeption, TreeSet, Vector	SQLTransactionRollbackException, SQLTmansertConnectionException,
PriorityBlockingQueue, PriorityQueue, RoleLa SQLInveidAuthorizationSpecException, SQLI SQLTnansientException, SQLINaming, Stack,	e, Rost-Intelovedue, Rovberhamm, Senetziopton, Dervoluseer, SulLikerthocksepton, SulLiketziopton, SulLikerthocksepton, SulLikertho	SQLTransactionRolbackException, SQLTransentConnectionException,
PhontyBlockingCueue, PriorityCueue, RoseLi SQLInveidAuthorizationSpecException, SQLI SQLTnansientException, SQLWarning, Stack, public interface Iterable <t></t>	e, RoeUnresovedue, RowSenhammg, Senestxoption, Dervosubeeder, SulLCienthocxxeption, SulLDestException, SulLText NonTransientConnectionRicogion, SQLNonTransientException, SQLRecoverableException, SQLSynlaxEmorException, SQLTmeoutException, SyncPactoryException, SynchronousQueue, SyncProviderException, TreeSet, Vector	uniteds.pponedcxoption, SGLImagryConstrainty outpont xoption, SGLTransactionRolbackException, SGLTransentConnectionException,
PhorityGlockingQueue, PriorityQueue, RoseLi SQLInveidAuthorizationSpecException, SQLI SQLTransentException, SQLWarning, Stack, public interface iterable <t> implementing this interface allows an object to be</t>	e, Roburnsovedus, Rovberhamm, Seinstzkopton, Dervokuseder, SulLikerthockspton, SulLiketkopton, SulLikerthockspton, SulLikerthockspt	uninkotsupponed sception, S-Limegrig-Constrainty outpont sception, SQLTransactionRol/backException, SQLTransentConnectionException,
PhontyGlockingQueue, PriortyQueue, RoseLi SQLInvaldAuthorizationSpecException, SQLI SQLInvanientException, SQLIWarning, Stack, public interface Iterable <t> Implementing this interface allows an object to be Since:</t>	e, Rostmesovedual, Rovberhamm, Seinestxopton, bervosubeed, bull.cherthockopton, SQLDoerthockopton, SQLDeepton, SQLTeepton, SQLTeepton, SQLSepton, SQL	uninedsupponedsusption, Sullingerg/Lonetrary-outions.soppion, SQLTransactionRolBackException, SQLTransentConnectionException,
PhontyGlockingCueue, ProntyCueue, RiseLa SQLInveidAuthorizationSpecException, SQLI SQLInveidAuthorizationSpecException, SQLI SQLInveidAuthorizationSpecException, SQLIVerning, Stack, public interface Iterable <t> implementing this interface allows an object to be Since: 1.5</t>	e, Roshmesovedue, Rowberhammg, Seneszkopton, Servoskoeder, SulLicentrockopton, SulLicentrockopton, SulLicentro KenTransentiConectionRicontexcepton, Sullicentrockopton, SulLicentrockopton	uninedsupported xception, SuLImagrig/Lonatranty-outlont xception, SQLTransactionRolBackException, SQLTransentConnectionException,
PhontyGlockingQueue, ProntyQueue, ProntyGlocking SQLInveidAuthorizationSpecException, SQLI SQLInveidAuthorizationSpecException, SQLI public Interface Iterable <t> implementing this interface allows an object to be Binoe: 1.5</t>	e, Roshmesovedue, Rowberhammg, Senesticopton, Dervoluseer, SulLicentrockopton, SulLicentrockopton, SulLicentro NonTransentiConnectionRicontexplore, SQLNonTransentException, SQLNoversiteException, SQLSynlaxEmorException, SQLTmeoutException, ByncPactoryException, SynchronousQueue, SyncProviderException, TreeSet, Vector	SQLTransactionRolbackException, SQLTransentConnectionException,
PhontyGlockingQueue, ProntyQueue, Picetz SQLInveidAuthorizationSpecException, SQLI SQLInveidAuthorizationSpecException, SQLI SQLInveidAuthorizationSpecException, SQLINeiming, Stack, public interface Iterable <t> implementing this interface allows an object to be Since: 1.5</t>	er, RoeUnresovedue, RowSenhammg, Senesticopton, Servosubeer, SulLCentrockxepton, SulLCentrockxepton, SulLCentro NonTransientConnectonitXcepton, SQLNonTransentTixcepton, SQLReventiveException, SQLSyntaxEmrException, SQLTmeoutException, SyncFactoryException, SynchronousQueue, SyncProviderException, TreeSet, Vector	SQLTransactionRotbackException, SQLTransentConnectionException,
ProntyGookingCueue, ProntyGueue, ProntyGueue, PiontyGookingCueue, ProntyGueue, SQLJ SQLInverldAuthorizationSpecException, SQLJ gobile interface Iterable <t> implementing this interface allows an object to be Since: 1.5 Method Summary</t>	e, RobUntesovedue, Rovderhammg, Senetzkopton, Servouldeer, SulLicentrockeepton, SulLicentrock	SQLTransactionRotbackException, SQLTransentConnectionException,
PhontyBookingQueue, ProntyQueue, PiontyBookingQueue, ProntyGueue, SQLJ BQLInveidAuthorizationSpecException, SQLJ BQLInveidAuthorizationSpecException, SQLJ public interface Iterakle <t> implementing this interface allows an object to be Binoe: 1.5 Method Summary Vectoris</t>	e, Roburnskovedus, Rovderhamm, Benatzkopton, Servesubeet, SulLikerindokkepton, SulLikerindokepton, SulLike	SQLTransactionRotbackException, SQLTransactConnectionException,
PriorityGockingQueue, PriorityGueue, RiseLa SQLInveidAuthorizationGpecException, SQLI SQLInveidAuthorizationGpecException, SQLI Instance, SQLInveidAuthorizationGpecException, SQLI Instance, SQLInveidAuthorizationGpecException, SQLI Instance, SQLInveidAuthorizationGpecException, SQLI Instance, SQLInveidAuthorizationGpecException, SQLINVeidAuthorizationGpecExcept	er, Roek-Interprotected, Roek-Detromating, Senatz-Logiston, Detrobuceded, SubJ.Cherthood Acaption, SubJ.Cherthood Acaptio	SQLTransactionRotbackException, SQLTransenConnectionException,
PriorityGockingQueue, PriorityGueue, RiseLi BQLInveidAuthorizationGpecException, SQLI BQLInveidAuthorizationGpecException, SQLI BQLInveidAuthorizationGpecException, SQLI BQLInveidAuthorizationGpecException, SQLI BQLInveidAuthorizationGpecException, SQLI BQLInveidAuthorizationGpecException, SQLI BQLInveidAuthorizationGpecException, SQLI Instrumenting this interface allows an object to be Binder: 1.5 Method Summary Methods Modifier and Type Therefore:T>	er, Noetheresoveckue, Rowdeenhamme, Benezizeopton, Bervoor.oeder, BuilLowentoockoepton,	SQLTransactionRotbackException, SQLTransactConnectionException,
PhorityBookingQueue, PriorityQueue, RoseLa BQLInvesidAuthorizationQpecException, SQLI BQLTensisentException, SQLIWarning, Stack, public interface Iterable <id- implementing this interface allows an object to be Rince: 1.5 Method Summary Methods Modifier and Type Tterater<id-< td=""><td>er, Roshnesovedue, Rowberhammg, Benesitskopton, Bervou beet, but Unerthockeepton, But Deetherbore, But Deeth</td><td>SQLTransactionRolbackException, SQLTransentConnectionException,</td></id-<></id- 	er, Roshnesovedue, Rowberhammg, Benesitskopton, Bervou beet, but Unerthockeepton, But Deetherbore, But Deeth	SQLTransactionRolbackException, SQLTransentConnectionException,
PriorityGlockingQueue, PriorityGueue, RiseLa SQLIvensionLawrondpectionepetion, SQLIVerning, Stack, public interface Iterable <t> implementing this interface allows an object to be Since: 1.5 Method Summary Methods Modifier and Type Iterator<t></t></t>	By Roburnsboredus, Rowderhammg, Senatzkopton, Dervoudeeder, SubLicentinockopton, SubLicentin	SQLTransactionRolbackException, SQLTransactConnectionException,
PhortyBookingOunue, PriortyGueue, RoseLa BOLInveidAuthorizationSpecEcomption, SQL/ BOLInveidAuthorizationSpecEcomption, SQL/ BOLInveidAuthorizationSpecEcomption, SQL/ Weblic Enterface Elevable <t> Inplementing this interface allows an object to be Binde: 1.5 Method Summary Methods Modifier and Type Iterator<t> Method Detail</t></t>	er, Roehnmetorveduel, Rowberhamme, Benetikkopton, Bervoukoeder, SubLicentinockoepton, BulLibertappon, BulLiber	SQLTransactionRolbackException, SQLTransaentConnectionException,
PhorthyBookingOunue, PhorthyGueue, RoseLa BOLInvestionAuthorisationSpecificorepton, SQLINaming, Stack, public interface Iterable <id- implementing this interface allows an object to be Since: 1.5 Method Summary Methods Modifier and Type Iterator</id- 	By Roburnsboredus, Rowdenhamme, Benetikkopton, Bervouceeter, SubLicentinooceepton, SubLicentinoo, SubJineeter By Rohmen Connection Roburnsbore, Synchronous Queue, SynchroviderException, TreeSet, Vector In the target of the "tonsach" statement.	SQLThereactionRolbackException, SQLThereartY-rolationException,
PriorityGockingQuinue, PriorityGuinue, PriorityGuinue, PriorityGuinue, Statu BQLInumientException, SQLINeming, Stack, public interface Iterable <t> implementing this interface allows an object to be Since: 1.5 Method Summary Nethods Modifier and Type Iterator<t> Iterator Iterator</t></t>	By Roburnsboredual, Roedberhamme, Benatzkooption, Barrookudeer, Sudu Ceentrookudeeton, Budu Deeksteppion, Budu Teeksen, ByncProviderException, SQL ByndaeEmorException, SQL TemeoutException, ByncProviderException, TreeSet, Vector In the target of the "tonsach" statement. Method and Description Liserster [] Returns an iterator over a set of elements of type T.	SQLThensectionRolbackException, SQLThensentConnectionException,
PromyGockingCurrue, PromyGurue, PromyGurue, PromyGurue, PromyGurue, Statu SQLInansamException, SQLINaming, Stack, public interface Iterable <t> implementing this interface allows an object to be Since: 1.5 Method Summary Methods Modifier and Type Iterator<t> Iterator Iterator () Return an iterator () Return an iterator over a set of elements of the</t></t>	Non TransmitConnectionStageton, SylubaritansemitScappton, SulLivershietScappton, SulLiversh	SQLThensectionRolbackException, SQLThensentConnectionException,

Iterable

Creater Package Creater Uses Traines No Frances All Classes Burnnary Needed Feld Constr Method Classes Interface Iterable <t> Type Parameters: 1 - Pe type of elements returned by the Ilenstor All Known Subinterfaces: BeanContext, BeanContextServices, BlockingOueue<e>, Colection<e>, Deque<e>, DirectoryStream<t>, Lat<e>, NavigabieSet<e>, Path, Cusue<e>, SecureDirectoryStream<t>, Set<e>, SondeSet<e>, Transfe All Known Implementing Classes: AstractCollection, AbstractSequentiaList, AbstractSequentiaList, AbstractSequentiaList, AbstractSequentiaList, AbstractSequentiaList, AmyBiockingOueue, ArrayDeepa ArrayLat, All Novel, Jak, BatchUpdateException, BeanContextServicesSupport, BeanContextServicesSupport,</e></e></t></e></e></e></t></e></e></e></t>	And " Per Standard I
ev Class Next Class Frames No Frames Al Classes mmary Nexted Field Constr Method jeva.lang Interface Iterable <t> Type Parameters:</t>	
java.lang Interface Iterable <t> Type Parameters: 1 - The type of elements returned by the iterator All Known Subinterfaces: BeanContext, BeanContextServices, BlockingDeque<e>, BlockingDueue<e>, Collection<e>, DirectoryStream<t>, Liel<e>, NavigabilisEet<e>, Path, Queue<e>, SecureDirectoryStream<t>, Set<e>, SortedSet<e>, Transle All Known Implementing Classes: AbstractColection, AbstractLiet, AbstractDeques, AbstractSequentiaLiet, AbstractSet, AmayBiockingDueue, ArrayDeque, ArrayDeque, AmayLat, All HouteLiet, BatchUpdateException, BeanContextServicesSupport, BeanContextSupport, Concurrent</e></e></t></e></e></e></t></e></e></e></t>	
Interface Iterable <t> Type Parameters:</t>	
Type Parameters:	
T - The type of elements returned by the iterator All Known Subinterfaces: BeanContext, BeanContextServices, BlockingDeque <e>, BlockingQueue<e>, Collection<e>, Deque<e>, DirectoryStream<t>, Lei<e>, NavigabieSet<e>, Path, Queue<e>, SecureDirectoryStream<t>, Set<e>, SortedSet<e>, Transfe All Known Implementing Classes: AbstractCollection, AbstractServices, BlockingQueue, AbstractSet, AmyBlockingQueue, ArrayDeque ArrayLet, A Industration, BeanContextServicesSupport, BeanContextServicesSupport, BeanContextServicesSupport, BeanContextServicesSupport, Concurrent</e></e></t></e></e></e></t></e></e></e></e>	
All Known Subinterfaces: BeanContext, BeanContextServices, BlockingDeque <e>, BlockingQueue<e>, Collection<e>, Deque<e>, DirectoryStream<t>, List<e>, NavigabieSet<e>, Path, Queue<e>, SecureDirectoryStream<t>, Set<e>, SortedSet<e>, Transfe All Known Implementing Classes: AbstractCollection, AbstractList, AbstractSequentiaList, AbstractSet, AmyBlockingQueue, AmyDeque AmyLat, ApributeList, BatchUpdateException, BeanContextServicesSupport, BeanContextServicesSupport, BeanContextServicesSupport, Concurrent</e></e></t></e></e></e></t></e></e></e></e>	
BeanContext, BeanContextServices, BlockingDeque <e>, BlockingDueue<e>, Collection<e>, Deque<e>, DirectoryStream<t>, List<e>, NavigableSet<e>, Path, Queue<e>, SecureDirectoryStream<t>, Set<e>, SortedSet<e>, Transh All Known Implementing Classes: AbstractCollection, AbstractSet, AnayBrockingDueue, AnayBrockingDueue, AnayDeque AmayLat, ApributeList, BatchUpdateException, BeanContextServicesSupport, BeanContextServicesSupport, BeanContextServicesSupport, Concurrent</e></e></t></e></e></e></t></e></e></e></e>	
All Known Implementing Classes: AbstractCollection, AbstractList, AbstractSequentialList, AbstractSet, AmsyBiockingQueue, AmsyDeque AmsyList, Aprovalue, BatchUpdateException, BeanContextServicesSupport, BeanContextSupport, Concurrents	<b+sueudia< td=""></b+sueudia<>
AbstractCollection, AbstractList, AbstractDasue, AbstractDequentialList, AbstractDet, AmayBiockingQueue, AmayDeque A	
ConcurrentLinkeQueue, ConcurrentBkpLixtSet, CopyOnWiteArrayList, CopyOnWiteArraySet, DataTruncation, DelayDown, LinkeBcopton, SQL DataTruncation, DelayDown, LinkeBcopton, SQL DataTruncation, DelayDown, LinkeBcopton, SQL DataTruncation, SQL DataTruncation, DelayDown, SQL DataTruncation,	und Int, InkedTransferQueu InterdTransferQueu IntertConnectionException,
15	
Method Summary	
Nethods Notifier and Tuna Nethod and Description	
Notation and Type Method and Description Therefore ()	
Returns an iterator over a set of elements of type T.	
Method Detail	
Haratar	
Herator	
Herator Tterator<> iterator()	

-

public interface Graph<V,E> {

Position<V> insertVertex(V v);

Position<E> insertEdge(Position<V> from, Position<V> to, E e)
throws InvalidPositionException, InsertionException;





Separate generic

types for vertices <V>

and edges <E>

public interface Edge<T> extends Position<T> {}
public interface Vertex<T> extends Position<T> {}

```
public interface Graph<V,E> {
```

. . .

```
Vertex<V> insertVertex(V v);
```

```
Edge<E> insertEdge(Vertex<V> from, Vertex<V> to, E e)
throws InvalidVertexException, InsertionException;
```

V removeVertex(Vertex<V> p)
 throws InvalidVertexException, RemovalException;

E removeEdge(Edge<E> p)

throws InvalidEdgeException;

Iterable<Vertex<V>> vertices();

```
Iterable<Edge<E>> edges();
```

Now clients can check at compile time if their types are correct

What else is missing from the interface?

```
public interface Graph<V,E> {
    ...
    Vertex<V> fromVertex(Edge<E> e) ...
    Vertex<V> toVertex(Edge<E> e) ...
    ...
}
```

Now clients can check their code to see where the edges go!

Don't just define an interface, try to use it!

Graph Searching





Search



Mr Kevin Bacon Plays Six Degrees Of Kevin Bacon

40,198 views

🖌 479 🚚 14 🏕 SHARE ≡∔ SAVE •••

https://www.youtube.com/watch?v=Rmn-amJ9UA4



CASTING CALLS TOM HARDY

Won 1 Golden Globe. Another 14 wins & 26 nominations. See more awards »

845 photos | 129 videos »

Kevin Bacon and Bipartite Graphs

72 Find the **shortest** path from Kevin Bacon 60 to **Jason** Lee 35 Breadth First Search: 4 hops 3 Bacon Distance: 2 45



BFS

BFS(start, stop)

// initialize all nodes dist = -1
start.dist = 0
list.addEnd(start)
while (!list.empty())
cur = list.begin()
if (cur == stop)
print cur.dist;
else
foreach child in cur.children
if (child.dist == -1)
child.dist = cur.dist+1
list.addEnd(child)



<u>0</u> <u>A</u>,B,C <u>B</u>,C,D,E <u>C</u>,D,E,F,L <u>D</u>,E,F,L,G,H <u>E</u>,F,L,G,H,I <u>F</u>,L,G,H,I,J <u>L</u>,G,H,I,J,X <u>G</u>,H,I,J,X,O <u>H</u>,I,J,X,O

[How many nodes will it visit?]

[What's the running time?]

[What happens for disconnected components?]

BFS

BFS(start, stop)

// initialize all nodes dist = -1
start.dist = 0
list.addEnd(start)
while (!list.empty())
cur = list.begin()
if (cur == stop)
print cur.dist;
else
foreach child in cur.children
if (child.dist == -1)
child.dist = cur.dist+1
list.addEnd(child)



<u>A</u>,B,C <u>B</u>,C,D,E <u>C</u>,D,E,F,L

<u>0</u>

<u>D</u>,E,F,L,G,H <u>E</u>,F,L,G,H,I <u>F</u>,L,G,H,I,J <u>L</u>,G,H,I,J,X <u>G</u>,H,I,J,X,O <u>H</u>,I,J,X,O

<u>I</u>,J,X,O,M

<u>J</u>,X,O,M

X,O,M,N

<u>O</u>,M,N

<u>M</u>,N

Ν

DFS

DFS(start, stop)
// initialize all nodes dist = -1
start.dist = 0
list.addEnd(start)
while (!list.empty())
 cur = list.end()
 if (cur == stop)
 print cur.dist;
 else
 foreach child in cur.children
 if (child.dist == -1)
 child.dist = cur.dist+1
 list.addEnd(child)



<u>0</u> A,B,<u>C</u> A,B,G,<u>H</u> A, B, G, MA,B,G A,B,<u>L</u> A,B,<u>O</u> A,B,<u>N</u> A,B,J A,B,E,<u>F</u> A,B,E,<u>K</u> A,B,<u>E</u> A,<u>B</u> <u>A</u> <u>D</u> I



public interface Graph<V,E> {

. . .

}

boolean marked(Vertex<V> v); boolean marked(Edge<E> e); void mark(Vertex<V> v); void mark(Edge<E> e); void clearMarks(); ...



```
public interface Graph<V,E,L> {
    ...
    L label(Vertex<V> v);
    L label(Edge<E> e);
    void label(Vertex<V> v, L l);
    void label(Edge<E> e, L l);
    void clearLabels();
    ...
}
```

More flexible, but client will have to use a consistent type for all labels



```
void label(Edge<E> e, Object label);
```

```
void clearLabels();
```

Very flexible, but client will have to cast Object to correct type

Note use of overloading: compiler will figure out which version you meant based on parameters passed on. Good for simple, closely related methods



DFS: Stack

// in What is the runtime complexity? DF star What is the space complexity? list. A,B,G,<u>H</u> while (!list.empty()) cur = list.end() A,B,G,<u>M</u> if (cur == stop) print cur.dist; A,B,G else A,B,L foreach child in cur children A,B,<u>O</u> if (child.dist == -1) A,B,N child.dist = cur.dist+lA,B,J list.addEnd(child) A,B,E,FA,B,E,<u>K</u> A,B,<u>E</u> D:2 A,<u>B</u> E:7 A D G:2

Breath First Searching



Breath First Searching



 2^d search space: 2^{20}

Bi-Directional Breath First Searching



More to come...

	19.3 19.4 19.5	Splay Trees 1 Treaps 1 Bloom Filters 1	81 83 85
20	Gra	ph Algorithms 1	88
	20.1	Topological Sorting	88
	20.2	Minimum Spanning Trees 1	91
	20.3	Shortest Paths	92
A	How	to write code 1	97
	A.1	One Change at a Time aka Incremental Coding aka Baby Steps	97
	A.2	Prototype to Learn aka Explore Small Examples	98
	A.3	Don't Repeat Yourself aka Once and Only Once	98
в	Hac	king AVL Trees 2	00
	B.1	Play with an AVL tree applet 2	00
	B.2	Understand height/balance calculations 2	00
	B.3	Figure out single rotations, implement, test	00
	B.4	Figure out double rotations, implement, test	01
	B.5	Write a general balancing method	02
	B.6	Add balancing to your insert/remove code	02
	B.7	Test, test, and test again	02
C	Data	Compression 2	03
-	C.1	Communication 2	03
	C2	Encoding and Decoding	04
	0.0	C 2.1 Fixed Length Codes 2	04
		C 2 2 Variable-Length Codes 2	05
	C3	Puplength Coding	06
	6.0	C 3.1 Implementation Notes 2	07
	CA	Huffman Coding	07
	0.4	C.4.1 Implementation Notes	10
		C.4.1 Implementation roles	10

Next Steps

- I. Work on HW4
- 2. Check on Piazza for tips & corrections!

