Topic 16 Queues

"FISH queue: n.

[acronym, by analogy with FIFO (First In, First Out)] 'First In, Still Here'. A joking way of pointing out that processing of a particular sequence of events or requests has stopped dead. Also FISH mode and FISHnet; the latter may be applied to any network that is running really slowly or exhibiting extreme flakiness."

-The Jargon File 4.4.7

Queues

- A sharp tool, like stacks
- A line
 - –In England people don't "get in line" they "queue up".





CS314

Queues

Queue Properties

- Queues are a first in first out data structure
 - FIFO (or LILO, but I guess that sounds a bit silly)
- Add items to the end of the queue
- Access and remove from the front
 - Access to the element that has been in the structure the *longest* amount of time
- Used extensively in operating systems
 - Queues of processes, I/O requests, and much more



Queues in Operating Systems

- On a computer with N cores on the CPU, but more than N processes, how many processes can actually be executing at one time?
- One job of OS, schedule the processes for the CPU

Processes Resources File System	ms						
Load averages for the last 1, 5.	15 minutes	1.28, 1.4	9, 1.40				Quit Proce
Process Name	Status	% CPU	Nice	ID	Memory	A	Process
at-spi-registryd	Sleeping	0	0	3683	2.4 MiB		361
bonobo-activation-server	Sleeping	0	0	3158	332.0 KiB		366
bt-applet	Sleeping	0	0	3179	212.0 KiB		340
clock-applet	Sleeping	0	0	3241	676.0 KiB		371
dbus-daemon	Sleeping	0	0	3118	164.0 KiB		368
dbus-launch	Sleeping	0	0	3120	0 bytes		374
eggcups	Sleeping	0	0	3165	1.6 MiB		360
escd	Sleeping	0	0	3200	44.0 KiB		323
😻 firefox	Sleeping	0	0	15110	0 bytes		
🧕 firefox-bin	Sleeping	0	0	15126	143.5 MIB		
gam_server	Sleeping	0	0	3220	156.0 KiB		
gconfd-2	Sleeping	0	0	3126	504.0 KiB		
🞅 gedit	Sleeping	0	0	1464	6.9 MIB	-	
					End Proc		
						14	

	k≠ Filter		Windowe	tu Pro	cesses	•	
spect	Filter				Show		
Process	Name	User	v %	CPU	# Threads	Real Memory	VSIZE
K Fi	nder	steve		0.00	4	21.05 MB	238.26 MB
😸 Sa	fari	steve		0.00	6	23.53 MB	252.85 MB
Io Io	ginwindow	steve		0.00	2	3.80 MB	185.66 MB
di i i	unes	steve		4.00	10	22.03 MB	239.66 MB
Ad all	tivity Monitor	steve		2.80	2	20.11 MB	246.64 MB
The IP	noto	steve		0.00	3	33.39 MB	281.90 MB
III Te	rminal	steve		0.00	4	12.98 MB	244.08 MB
Sy Sy	stemUlServer	steve		0.00	2	5.35 MB	227.74 MB
M De	ock	steve		0.00	2	6.43 MB	200.11 MB
CPU	System Memory	Disk	Activity	D	isk Usage	Network)
kar 4 50		Thread	6: 213		_	CPU Usage	
		Processe	s: 65				
lice: 0.00	2					10000	
dle: 91.5					- Hala		
	spect Process I Fin Sa Jon In Ac Sy Do CPU ser: 4.50 em: 4.00	Process Name Process Name Safari Safari Guinwindow Trunes Activity Monitor Guinwindo Trunes Activity Monitor Guinwindo Trunes Activity Monitor Trunes Activity Monitor CPU System Memory set: 4.50 CPU Activity	Process Name Process Pr	Process Name Process Proce	Process Name User * KOU Finder * KOU \$ 000 Finder \$ 000 \$ 000 Finder \$ 000 \$ 000 Finder \$ 000 \$ 000 Topinselow \$ 5000 \$ 000 Activity Montor \$ 5000 \$ 000 Terminal \$ 5000 \$ 000 System/Utiferer \$ 5000 \$ 000 CPU System Memory Dik Activity D # 400 \$ Threads: 213 \$ 000	Filter Show Process Name User % CPU Threads Staff 0.00 4 5 10 Threads Staff seve 0.00 4 5 10 10 Ioginwidow seve 0.00 4 3 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 <t< td=""><td>Protects Name Distr Show Process Name Boar V CPU # Threads Real Momory Image: Stress 0.00 4 21.03 MB Image: Stress 0.00 4 21.03 MB Image: Stress 0.00 2.23 MB 2.23 MB Image: Stress 0.00 2 2.23 MB Image: Stress 0.00 3 3.33 MB Activity Montor steve 0.00 2 5.33 MB Terminal steve 0.00 2 5.33 MB Dock System/Memory Disk Activity Disk Usage Network CPU System Memory Disk Activity Disk Usage Network cst<4.30</td> Image: 13 CPU usage Network</t<>	Protects Name Distr Show Process Name Boar V CPU # Threads Real Momory Image: Stress 0.00 4 21.03 MB Image: Stress 0.00 4 21.03 MB Image: Stress 0.00 2.23 MB 2.23 MB Image: Stress 0.00 2 2.23 MB Image: Stress 0.00 3 3.33 MB Activity Montor steve 0.00 2 5.33 MB Terminal steve 0.00 2 5.33 MB Dock System/Memory Disk Activity Disk Usage Network CPU System Memory Disk Activity Disk Usage Network cst<4.30

3

Queue operations			Queue interface, version 1				
▶ void en	queue(E item)		publ	lic interface Queue314 <e> {</e>			
-a.k.a. add(E item)			<pre>//place item at back of this queue public usid approve (F item);</pre>				
▶ E front				<pre>public void enqueue(E item);</pre>			
– a.k.a. E				//access item at front of thi	is queue		
► deque	-			//pre: !isEmpty() public E front();			
-							
<pre>- a.k.a. E remove() boolean isEmpty()</pre>			//remove item at front of thi	is queue			
			<pre>//pre: !isEmpty() public E dequeue();</pre>				
Specify m	nethods in an interface, allo	ow multiple		public i dequeue(),			
implemen	tations.			<pre>public boolean isEmpty();</pre>			
			}				
CS314	Queues	5	CS314	Queues	6		
Implementing a Queue			Clicker 1				
Given the internal storage container and			If implementing a queue with a singly linked list				
choice for	front and back of queue wh	nat are	-	eferences to the first and last n	-		
the Big O d	of the queue operations?			ail) which end of the list should	,		
	avList LinkedList Link			queue in order to have all que			
		edList					

		queue operatio		and tail) which end of the list should be the	`
	ArrayList	LinkedList (Singly Linked)	LinkedList (Doubly Linked)	of the queue in order to have all queue operations O(1)?	
enqueue				A. The front of the list should be the front of the	queue.
front				B. The back of the list should be the front of the	queue.
dequeue				C. Either end will work to make all ops O(1).	
isEmpty				D. Neither end will allow all ops to be O(1).	
CS314		Queues	7	CS314 Queues	8

How about imple native array?	e Implementation ementing a Queue with a	а	 Radix Sort radix is a synd 	cation of Queues	base 2
- Seems like a ste	P backwards		at individual di Use queues a Create an arra Starting with t value in queues empty queues	ting algorithm that onl gits during each pass s <i>buckets</i> to store eler ay of 10 queues he least significant dig that matches digit back into array g to next least significa	nents it place
CS314	Queues	9	CS314	Queues	10
Radix Sort	in Action: 1s plac	e	Radix	Sort in Action: 1	S
		e			S
original values ir	n array	e	original values	s in array	-
 original values ir 9, 113, 70, 86, 12, 	n array 93, 37, 40, 252, 7, 79, 12	e	 original values 9, 113, 70, 86, 	s in array 12, 93, 37, 40, 252, 7, 79,	-
 original values in 9, 113, 70, 86, 12, Look at ones plate 	n array 93, 37, 40, 252, 7, 79, 12 ace	e	 original values 9, 113, 70, 86, Look at ones 	s in array 12, 93, 37, 40, 252, 7, 79, place	12
 original values in 9, 113, 70, 86, 12, Look at ones pla <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, 	n array 93, 37, 40, 252, 7, 79, 12 nce 9 <u>3</u> , 3 <u>7</u> , 4 <u>0</u> , 25 <u>2</u> , <u>7</u> , 7 <u>9</u> , 1 <u>2</u>	e	 original values 9, 113, 70, 86, Look at ones 9, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 	s in array 12, 93, 37, 40, 252, 7, 79,	12
 original values in 9, 113, 70, 86, 12, Look at ones pla <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, 	n array 93, 37, 40, 252, 7, 79, 12 ace	e	 original values 9, 113, 70, 86, Look at ones <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, Queues: 	s in array 12, 93, 37, 40, 252, 7, 79, place	12
 original values in 9, 113, 70, 86, 12, Look at ones pla <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, 	n array 93, 37, 40, 252, 7, 79, 12 ace 9 <u>3</u> , 3 <u>7</u> , 4 <u>0</u> , 25 <u>2</u> , <u>7</u> , 7 <u>9</u> , 1 <u>2</u> 5 (all empty initially):	e	 original values 9, 113, 70, 86, Look at ones 9, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 	s in array 12, 93, 37, 40, 252, 7, 79, place 1 <u>2, 93, 37, 40, 252, 7</u> , 7 <u>9</u> ,	12
 original values in 9, 113, 70, 86, 12, Look at ones pla <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, 	n array 93, 37, 40, 252, 7, 79, 12 ace 9 <u>3</u> , 3 <u>7</u> , 4 <u>0</u> , 25 <u>2</u> , <u>7</u> , 7 <u>9</u> , 1 <u>2</u> 5 (all empty initially): 5	e	 original values 9, 113, 70, 86, Look at ones <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, Queues: 	s in array 12, 93, 37, 40, 252, 7, 79, place 1 <u>2, 93, 37, 40, 252, 7</u> , 7 <u>9</u> , 5	12
 original values in 9, 113, 70, 86, 12, Look at ones pla <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, Array of Queues 0 1 	n array 93, 37, 40, 252, 7, 79, 12 ace 9 <u>3</u> , 3 <u>7</u> , 4 <u>0</u> , 25 <u>2</u> , <u>7</u> , 7 <u>9</u> , 1 <u>2</u> 5 (all empty initially): 5	e	 original values 9, 113, 70, 86, Look at ones <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, Queues: 0 7<u>0</u>, 4<u>0</u> 1 	s in array 12, 93, 37, 40, 252, 7, 79, place 1 <u>2, 93, 37, 40, 252, 7, 79</u> , 5 6 8 <u>6</u>	12
 original values in 9, 113, 70, 86, 12, Look at ones pla <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, Array of Queues 0 1 2 	n array 93, 37, 40, 252, 7, 79, 12 ace 9 <u>3</u> , 3 <u>7</u> , 4 <u>0</u> , 25 <u>2</u> , <u>7</u> , 7 <u>9</u> , 1 <u>2</u> 6 (all empty initially): 5 6 7	e	 original values 9, 113, 70, 86, Look at ones <u>9</u>, 11<u>3</u>, 7<u>0</u>, 8<u>6</u>, Queues: 0 7<u>0</u>, 4<u>0</u> 1 2 1<u>2</u>, 25<u>2</u>, 1<u>2</u> 	s in array 12, 93, 37, 40, 252, 7, 79, place 1 <u>2, 93, 37, 40, 252, 7, 79,</u> 5 6 8 <u>6</u> 7 3 <u>7, 7</u>	12

Radix So	ort in Action: 10s	Radix Sort in Action: 10	0s
array	n order from 0 to 9 back into 2, 113, 93, 86, 37, 7, 9, 79	 Empty queues in order from 0 to 9 back in 7, 9, 12, 12, 113, 37, 40, 252, 70, 79, 86, 93 Now look at 100's place 	to array
Now look at 10's		7,9,12,12, <u>1</u> 13,37,40, <u>2</u> 52,70,79 VQueues:	9, _86, _93
• Queues:		0 _7, _9, _12, _12, _37, _40, _70, _79, _86, _93	5
0 _7, _9 1 <u>1</u> 2, <u>1</u> 2, 1 <u>1</u> 3	5 2 <u>5</u> 2 6	1 <u>1</u> 13 6 2 <u>2</u> 52 7	
2 3 <u>3</u> 7	7 <u>7</u> 0, <u>7</u> 9 8 <u>8</u> 6	3 8 4 9	
4 <u>4</u> 0 ^{CS314}	9 <u>9</u> 3 _{Queues} 13	CS314 Queues	14

15

Radix Sort in Action: Final Step

Empty queues in order from 0 to 9 back into array

Queues

7, 9, 12, 12, 40, 70, 79, 86, 93, 113, 252

Radix Sort Code

```
public static void sort(int[] list){
    ArrayList<Queue<Integer>> queues = new ArrayList<Queue<Integer>>();
    for(int i = 0; i < 10; i++)
         queues.add( new LinkedList<Integer>() );
    int passes = numDigits(list[0]); // helper method
    // or int passes = (int) Math.log10(list[0]);
    for(int i = 1; i < list.length; i++){
        int temp = numDigits(list[i]);
        if( temp > passes )
             passes = temp;
    for(int i = 0; i < passes; i++){
         for(int j = 0; j < \text{list.length}; j++)
             queues.get(valueOfDigit(list[j], i)).add(list[j]);
         int pos = 0;
         for(Queue<Integer> q : queues) {
             while(!q.isEmpty())
                 list[pos++] = q.remove();
  CS314
                                                                    16
                                 Queues
```