Lecture 23: Intro to Memory and Caching

CSE 373: Data Structures and Algorithms

Administrivia

P5 Seam Carving due June 12

- NO LATE DAYS
- Grade scope submissions limited to 1x hour
- Please please start early T_T

Exercise 5 More Graph Modeling due today

Exercise 6 goes out today, due

Post lecture questions coming, deadlines open until next week for extra credit

Midterm 2 Friday May 29th

- 1 week away!

Midterm 2

Midterm 2

- Same format as Midterm 1
- Canvas quiz
- Out on Friday morning 8:30am PDT
- Due Sunday morning 8:30am PDT
- NO LATE ASSIGNMENTS ACCEPTED
- ~1 hour of work
- Individual!

Topics

- Heaps
  - Array Representations
  - removeMin(), buildHeap()
  - percolations
- Graph implementations
  - Adjacency Matrix
  - Adjacency List
- Graph Algorithms
  - BFS
  - DFS
  - Dijkstra's
  - Prim's
  - Kruskal's
  - Topological sort
- Disjoint Sets
  - Array representations
  - Union, findSet
  - Path compression optimization
  - union by size optimization
- Sorting
  - Insertion
  - Selection
  - Heap
  - Merge
  - Quick
  - Best/worst case runtimes
  - Memory usage
  - stability

Big-takeaway question of the day

```
int sum = 0;
Node current = front;
while (current != null) {
    sum += current.data;
    current = current.next;
}
return sum;
```

```
int sum = 0;
int i = 0;
while (i < array.length) {
    sum += array[i];
    i++;
}
return sum;
```

Assuming that the array is the same size as the linked list:

Which do you think is most efficient given what we know?
(we have some tools to estimate runtime / what will take time, but we will learn more today)

A) Array code
B) Linked List code
C) Tied – both roughly equivalent

Goal for today:

connect your already existing programming knowledge to a newer understanding of memory and how it works today.

- demo in IntelliJ and Activity Monitor (Task Manager on Windows)

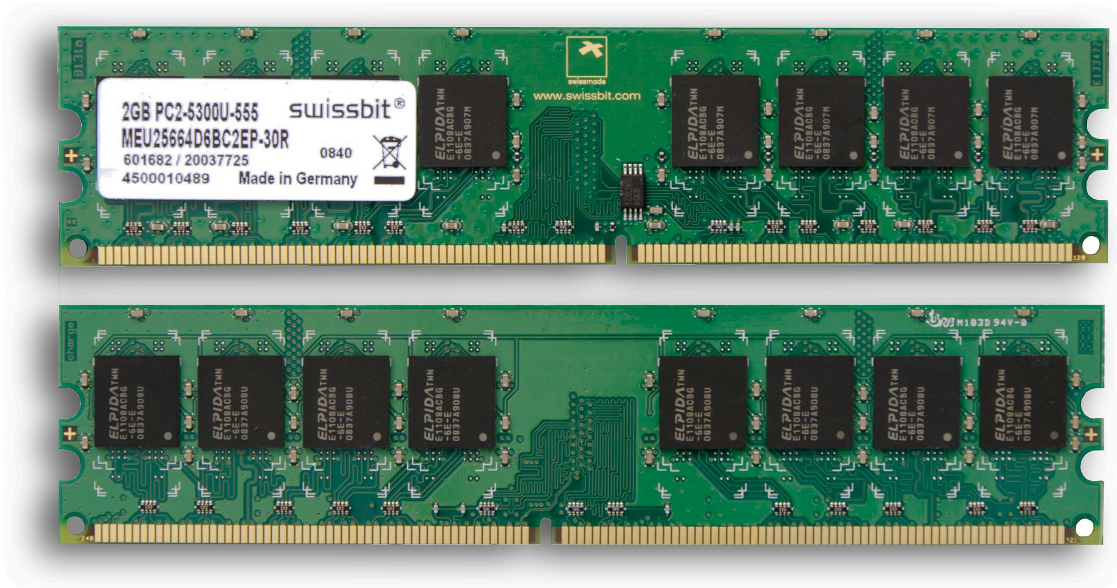
Roadmap

- RAM
  - What is RAM?
  - How do our programs interact with RAM? (visually)
    - arrays
    - linked lists
- caching
- design decisions with caching in consideration
- misc. vocabulary and setting up for next time

RAM (Random-Access Memory)

- **RAM is where data gets stored for the programs you run. Think of it as the main memory storage location for your programs.**

- RAM goes by a ton of different names: memory, main memory, RAM are all names for this same thing.



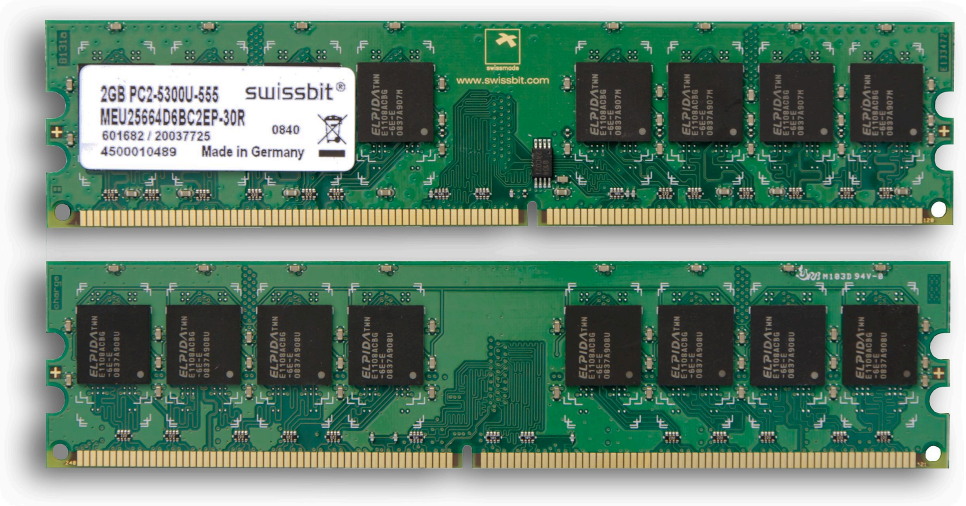
Process Name	Memory	Compressed M...	Threads
kernel_task	1.19 GB	0 bytes	144
IntelliJ IDEA	1,018.0 MB	194.7 MB	56
Microsoft PowerPoint	545.1 MB	238.9 MB	18
WindowServer	330.7 MB	170.9 MB	8
nsurlsessiond	320.8 MB	239.4 MB	3
Mattermost Helper	315.4 MB	32.0 MB	19
Google Chrome	291.7 MB	17.5 MB	31
Google Chrome Helper (Rend...	243.4 MB	91.5 MB	14
zoom.us	239.7 MB	61.8 MB	20
Google Chrome Helper (Rend...	236.6 MB	26.7 MB	14
Google Chrome Helper (GPU)	235.2 MB	19.7 MB	10
Google Chrome Helper (Rend...	203.4 MB	27.9 MB	16
Sublime Text	186.5 MB	170.9 MB	12
spindump	158.4 MB	80.0 MB	3
SystemUIServer	148.5 MB	24.9 MB	4
Finder	139.9 MB	56.3 MB	4
java	128.2 MB	61.3 MB	24
java	126.3 MB	110.3 MB	23
java	124.4 MB	27.8 MB	28
mds_stores	115.5 MB	36.2 MB	4
Mattermost	112.3 MB	37.5 MB	44
Cold Turkey Blocker	109.1 MB	49.2 MB	9
Google Chrome Helper (Rend...	102.8 MB	33.0 MB	16
Mail	91.4 MB	25.6 MB	7
Google Chrome Helper (Rend...	90.1 MB	62.4 MB	13
Google Chrome Helper (Rend...	88.1 MB	54.8 MB	13
Mattermost Helper	82.5 MB	44.8 MB	5
Google Chrome Helper (Rend...	77.4 MB	32.5 MB	13
Google Chrome Helper (Rend...	72.7 MB	51.4 MB	13

MEMORY PRESSURE

Physical Memory:	16.00 GB
Memory Used:	9.81 GB
Cached Files:	1.94 GB
Swap Used:	628.0 MB

RAM can be represented as a huge array

RAM:
- addresses, storing stuff at specific locations
- random access



=

Arrays
- indices, storing stuff at specific locations
- random access

This is a main takeaway

If you're interested in deeper than this : https://www.youtube.com/watch?v=fpnE6UAfbtU or take some EE classes?

Roadmap

- RAM
  - What is RAM?
  - How do our programs interact with RAM? (visually)
    - arrays
    - linked lists
- caching
- design decisions with caching in consideration
- misc. vocabulary and setting up for next time

A rough view of int and char data

```
int a = 5;
char letter = 'z'
```

RAM

A rough view of arrays and linked lists

```
int[] array = new int[3];
array[0] = 3;
array[1] = 7;
array[2] = 3;
```

```
Node front = new Node(3);
front.next = new Node(7);
front.next.next = new Node(3);
```

Activity (1 min in the chat)

Take 30 to think about what you remember about arrays and linked lists and how they relate to memory. Then in the 2nd half of 30 seconds, paste anything you recalled in the chat and we'll go through some of them

If you have time, try to start thinking about what you recalled and how these data structures look in memory.

Arrays	Linked Lists

A rough view of arrays and linked lists

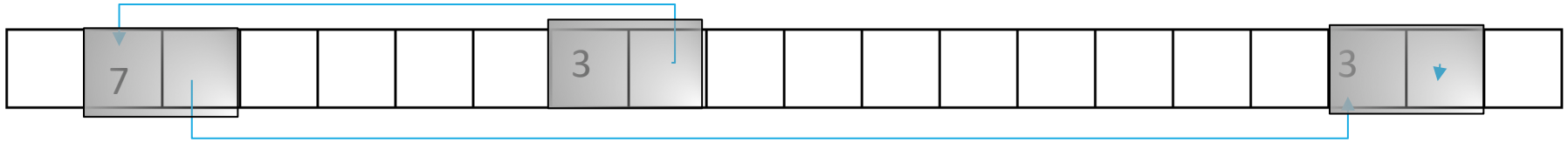
Arrays	Linked Lists
- Contiguous in memory - Simple	- Each node is linked to the next, so not contiguous in memory - Each node takes up more space than just the data itself – also stores pointers to the next (and prev if it's doubly-linked) - More flexible with memory / maybe can squeeze in nodes in places where we couldn't have a full contiguous array of that size

A rough view of arrays and linked lists

```
int[] array = new int[3];
array[0] = 3;
array[1] = 7;
array[2] = 3;
```


```
Node front = new Node(3);
front.next = new Node(7);
front.next.next = new Node(3);
```

(drawing singly linked list instead of doubly because drawings are hard / the two are similar)



A rough view of some other types

(not the main focus but nice to know)

Binary trees?

- they look similar to linked lists: each tree node object is going to be created randomly somewhere in memory

Strings?

- Strings are really a bunch of characters in a particular order. Could probably use either, but many languages choose to implement a String as an array of characters (Java)

Objects?

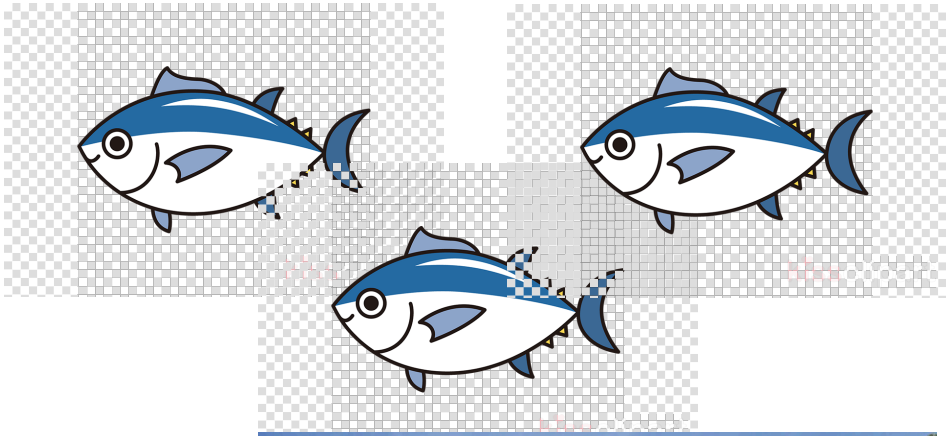
-An object in memory stores its fields all right next to each other (contiguous in memory) so the shape of it is like an array. There a couple of extra things objects also store, like a reference to its method instructions.

Roadmap

- RAM
  - What is RAM?
  - How do our programs interact with RAM? (visually)
    - arrays
    - linked lists
- caching
- design decisions with caching in consideration
- misc. vocabulary and setting up for next time

How memory is used and moves around


shutterstock.com • 298428176

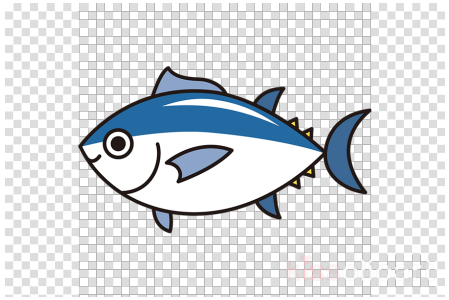


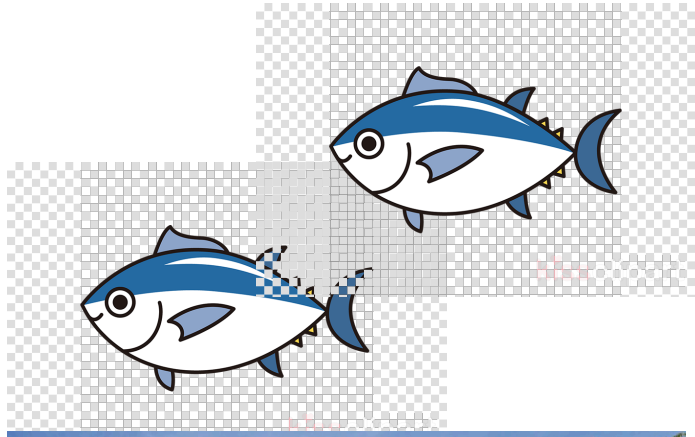

SAFEWAY


shutterstock.com • 298428176


SAFEWAY

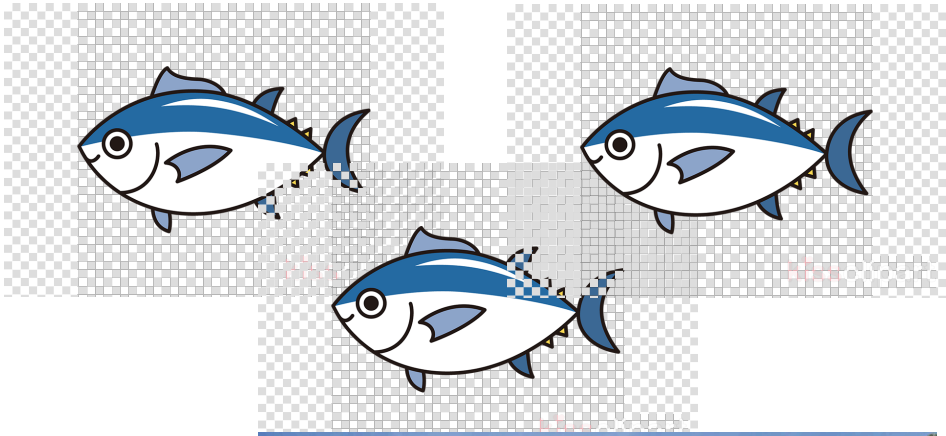

shutterstock.com • 298428176






SAFEWAY


shutterstock.com • 298428176

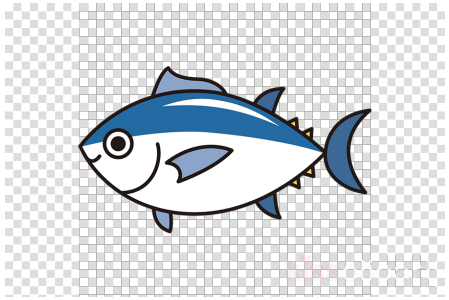


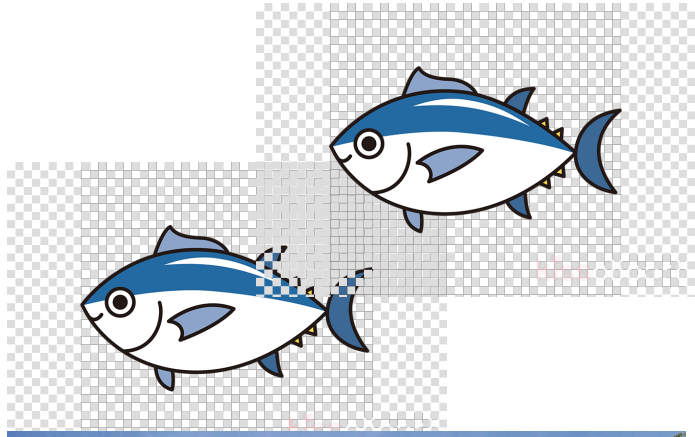

SAFEWAY


shutterstock.com • 298428176


SAFEWAY

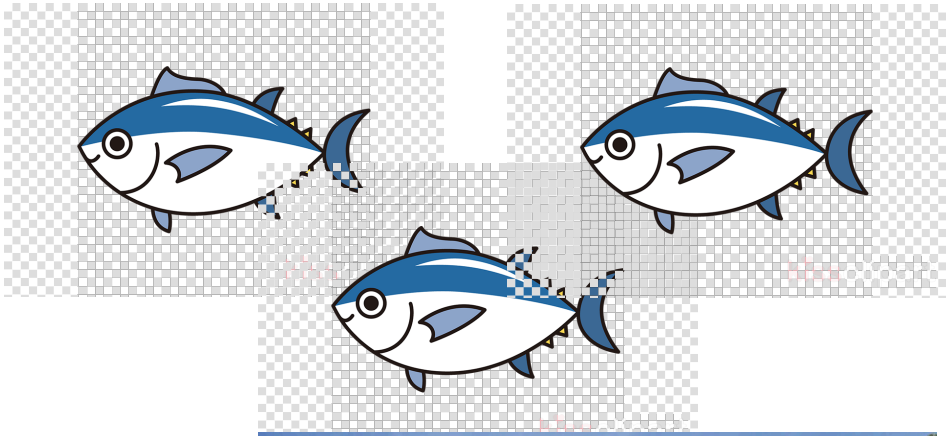

shutterstock.com • 298428176






SAFEWAY


shutterstock.com • 298428176

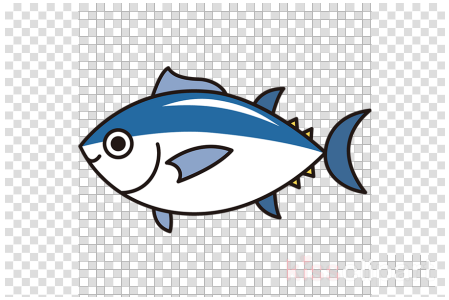


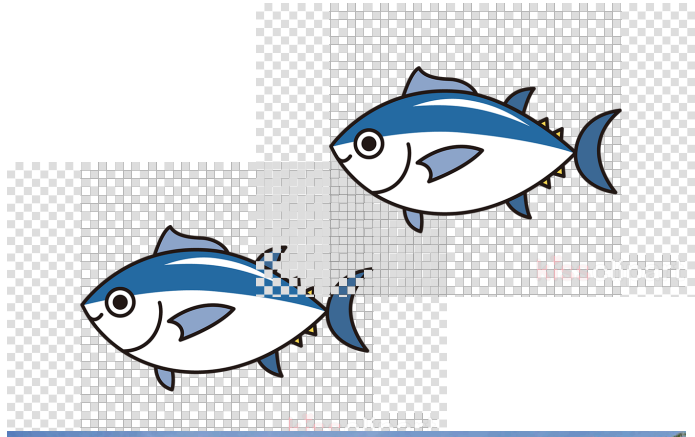

SAFEWAY


shutterstock.com • 298428176


SAFEWAY

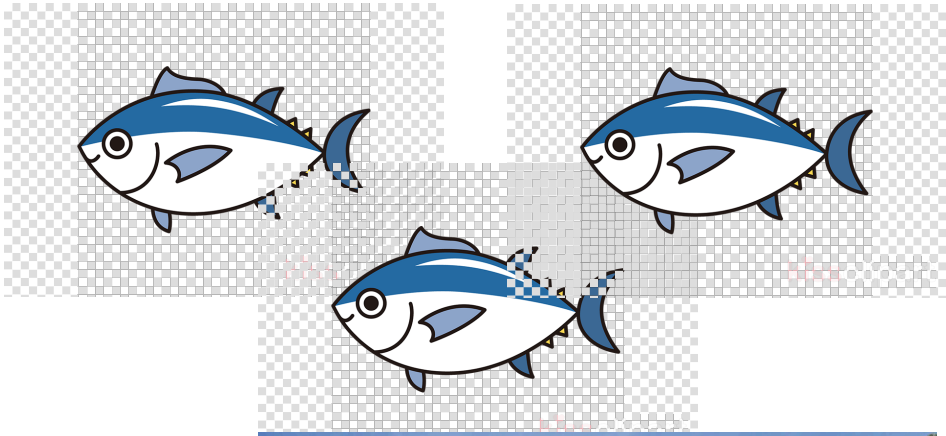

shutterstock.com • 298428176






SAFEWAY


shutterstock.com • 298428176

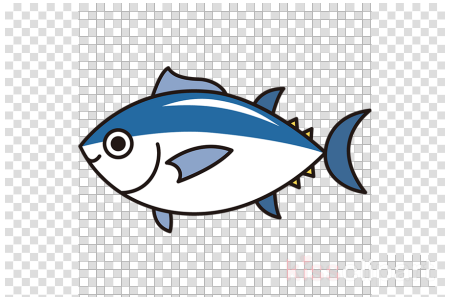


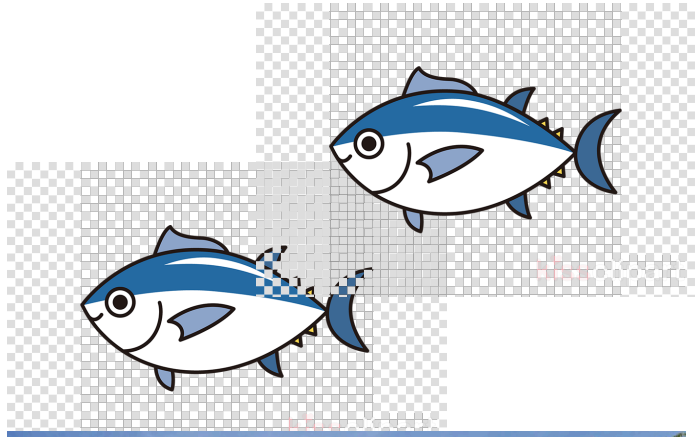

SAFEWAY


shutterstock.com • 298428176


SAFEWAY

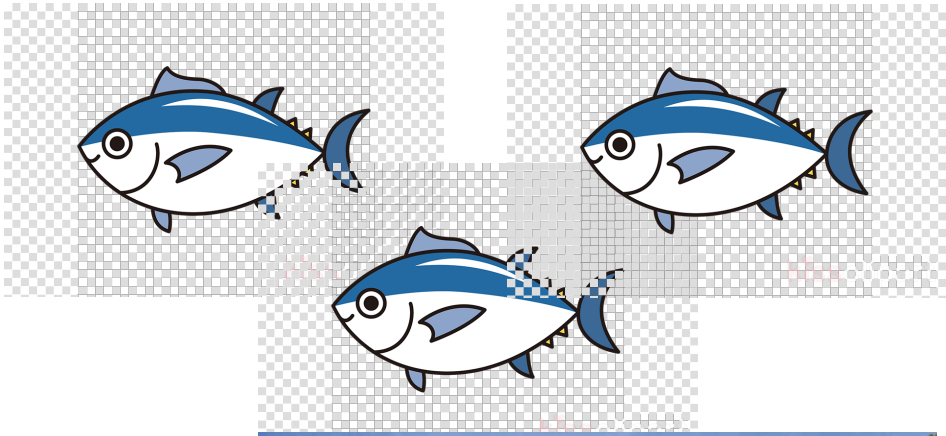

shutterstock.com • 298428176






SAFEWAY


shutterstock.com • 298428176

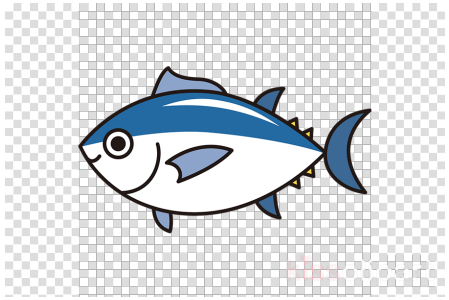


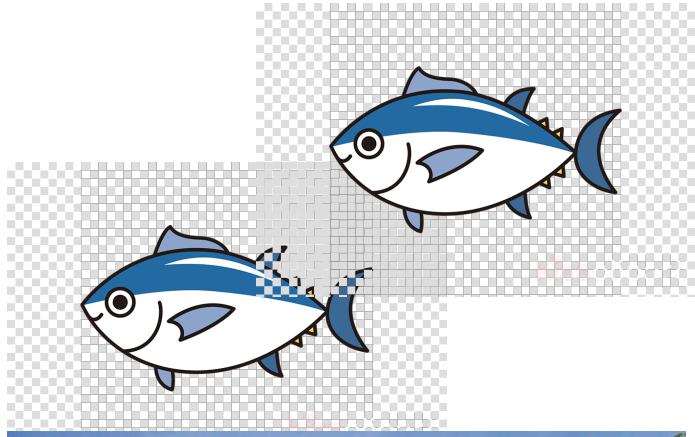

SAFEWAY


shutterstock.com • 298428176


SAFEWAY

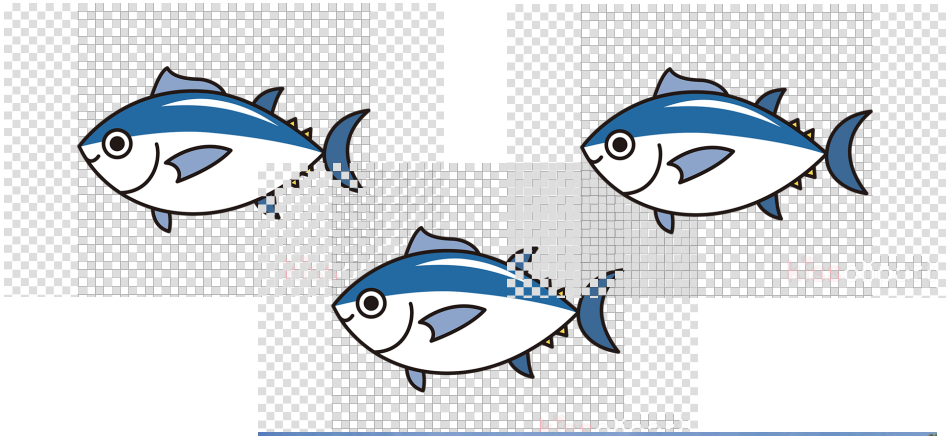

shutterstock.com • 298428176






SAFEWAY


shutterstock.com • 298428176



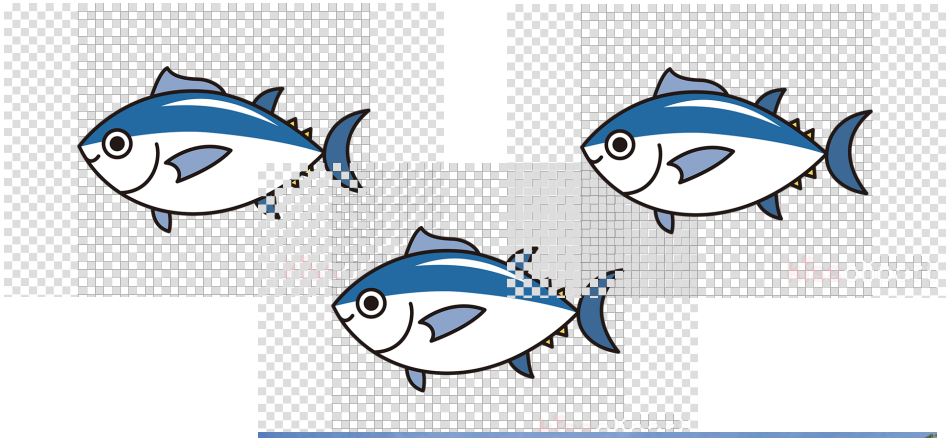

SAFEWAY

Solution to Mercy's traveling problem

If we know Mercy is going to keep eating tuna . . . Why not buy a bunch during a single trip and save them all somewhere closer than the store?

Let's get Mercy a refrigerator!


shutterstock.com • 298428176

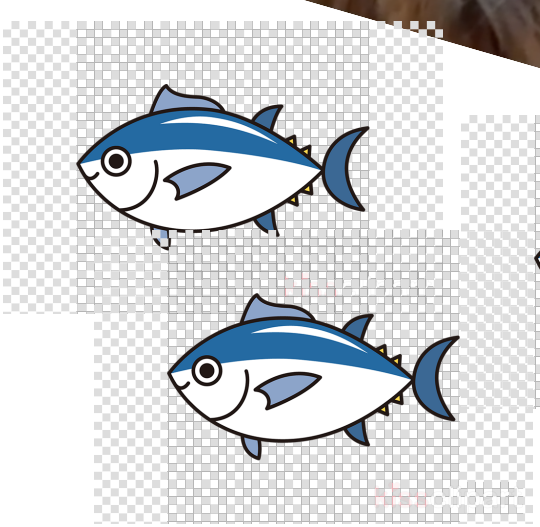


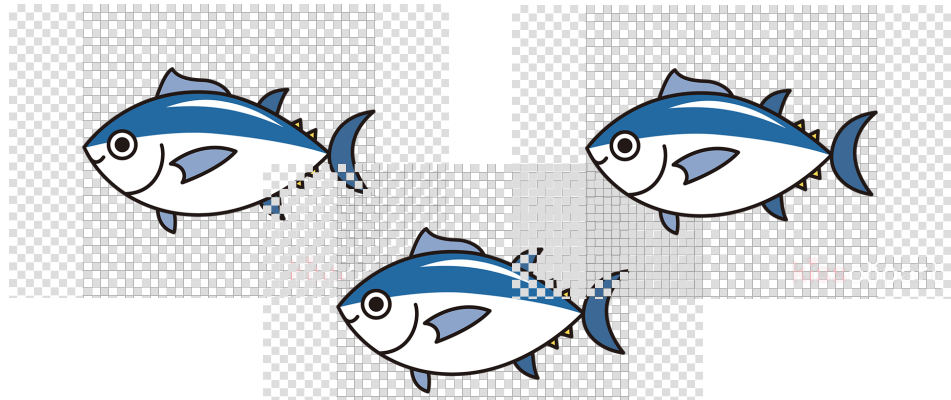

SAFEWAY


shutterstock.com • 298428176


SAFEWAY


shutterstock.com • 298428176

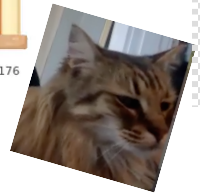


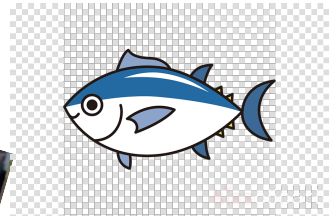



SAFEWAY

Recap + connecting analogy back to computer

Before





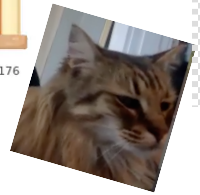
CPU

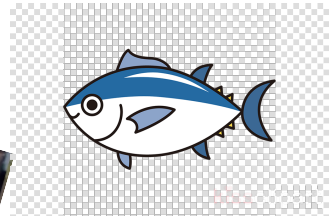
CPU – kind of like the home / brain of your computer. Pretty much all computation is done here and data needs to move here to do anything significant with it (math, if checks, normal statement execution).

Data travels between RAM and the CPU, but it's slow

RAM

After CPU





Bring a bunch of food back when you go all the way to the store

Cache!

Bring a bunch of data back when you go all the way to RAM

RAM

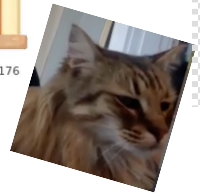
Cache

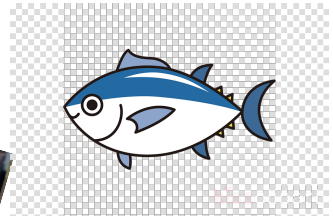
- Rough definition: a place to store some memory that's smaller and closer to the CPU compared to RAM. Because caches are closer to the CPU (where your data generally needs to go to be computed / modified / acted on) getting data from cache to CPU is a lot quicker than from RAM to CPU. This means we love when the data we want to access is conveniently in the cache.

- Generally we always store some data here in hopes that it will be used in the future and that we save ourselves the distance / time it takes to go to RAM.

- Analogy from earlier: The refrigerator (a cache) in your house to store food closer to you than the store. Walking to your fridge is much quicker than walking to the store!

After CPU





Bring a bunch of food back when you go all the way to the store

This is a big idea!

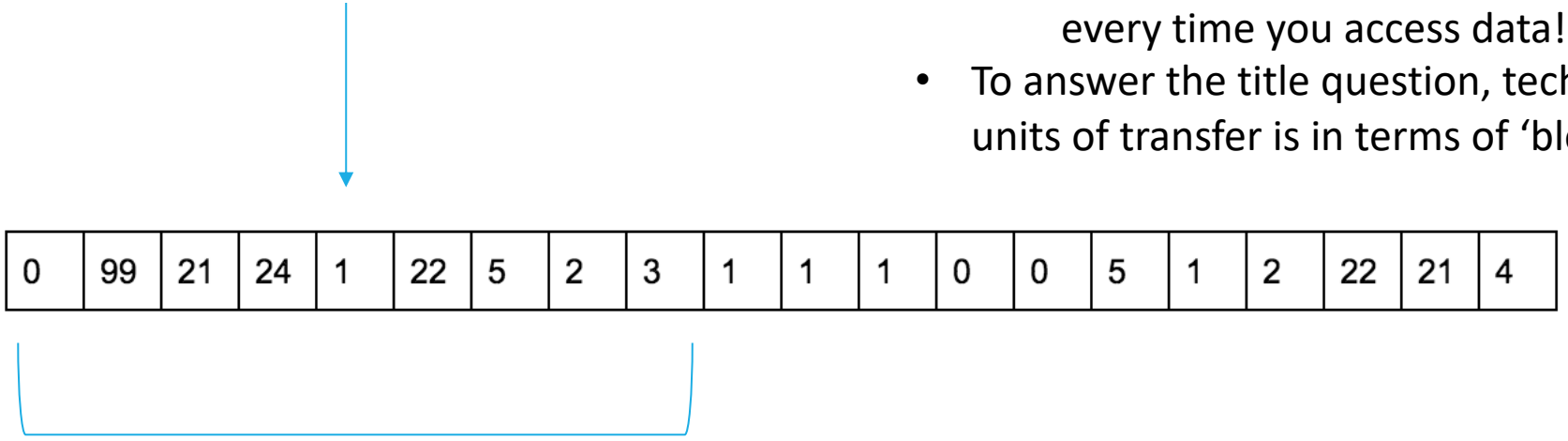
Cache!

Bring a bunch of data back when you go all the way to RAM

RAM

How is a bunch of memory taken from RAM?

- Imagine you want to retrieve the 1 at index 4 in RAM
- Your computer is smart enough to know to grab some of the surrounding data because computer designers think that it's reasonably likely you'll want to access that data too.
  - (You don't have to do anything in your code for this to happen – it happens automatically every time you access data!)
- To answer the title question, technically the term / units of transfer is in terms of 'blocks'.



How is a bunch of memory taken from RAM? (continued)

CPU

original data (the 1) we wanted to look up gets passed back to the cpu

cache

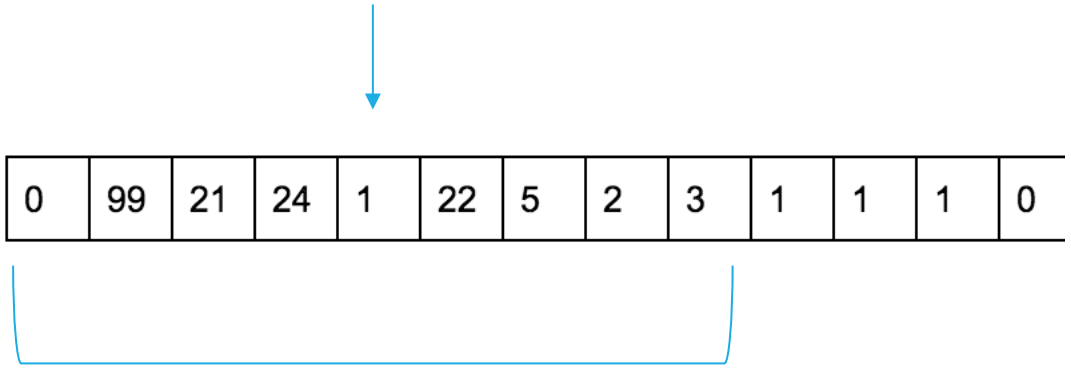
all the data from the block gets brought to the cache

0	99	21	24	1	22	5	2	3	1	1	1	0	0	5	1	2	22	21	4

How does this pattern of memory grabbing affect our programs?

- This should have a major impact on programming with arrays. Say we access an index of an array that is stored in RAM. Because we grab a whole bunch of contiguous memory even when we just access one index in RAM, we'll probably be grabbing other nearby parts of our array and storing that in our cache for quick access later.

Imagine that the below memory is just an entire array of length 13, with some data in it.



Just by accessing one element we bring the nearby elements back with us to the cache. In this case, it's almost all of the array!

Activity (3 min in the pollev)

#1) Come up with a scenario (pseudocode or a situation) where you'd immediately benefit from caching (grabbing a bulk amount of data and bringing all of it to be cached) when using arrays.

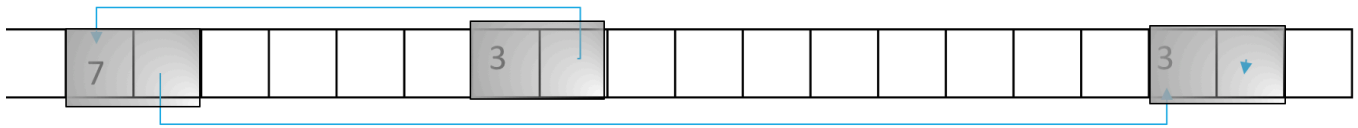
possible answer: Looping through a loop starting at index 0, and just incrementing the index by 1 each time

#2) Come up with 1 scenario where you might not benefit from caching when using arrays.

possible answer: Jumping around in the array outside of the block of memory grabbed from the recent trip to RAM or only ever accessing the array in RAM once

How does caching / your answer to the 2nd question affect Linked Lists or node data structures in general?

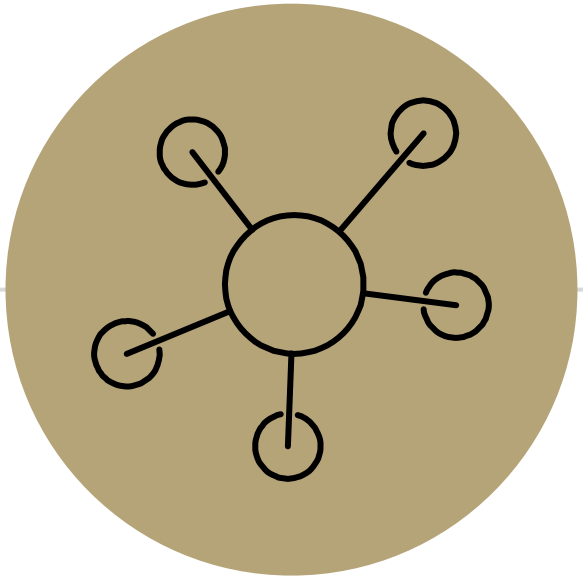
possible answer: The data of a Linked list is inherently spread out in memory so grabbing nearby data probably won't grab any other nodes in the linked list than the intended one – only by rare chance would we benefit and get another node close enough nearby.



Another demo, but timed

https://repl.it/repls/MistyroseLinedTransformation

(takes about 15 seconds to run)



Questions?

Roadmap

- RAM
  - What is RAM?
  - How do our programs interact with RAM? (visually)
    - arrays
    - linked lists
- caching
- design decisions with caching in consideration
- misc. vocabulary and setting up for next time

ArrayMap vs LinkedListMap (before (lecture 3) and now)

ArrayMap	LinkedMap
- put O(n) - get O(n) - containsKey O(n) - remove O(n)	- put O(n) - get O(n) - containsKey O(n) - remove O(n)

Both asymptotic runtimes are the same but we know all those methods have to loop through all the pairs.. which are either stored in a array or a linked list. So which is better? In terms of who benefits from caching, ArrayMap.

Hashing: Separate chaining vs Open addressing / Probing in the context of caching

Java uses separate chaining for their hash tables…

… but Python uses a variant on probing (open addressing)!

Why? It turns out that if you use open addressing, you can gain more caching benefits than if you use separate chaining.

(Note: there are plenty of reasonable motives to still use chaining, don't worry.)

Summary of caching benefits

- Helps arrays in many typical use cases (especially just looping through it normally) because we grab a bunch of the array data at once, and can look up that saved data quicker in the future.

- Doesn't especially help linked list use cases, as grabbing nearby memory probably won't include the next node or any of the other nodes.

Whenever you consider data structure implementations for your ADTs, arrays almost always have a plus because they have caching benefits. This is one of the most common points made online to support array implementations of ADTs.

Roadmap

- RAM
  - What is RAM?
  - How do our programs interact with RAM? (visually)
    - arrays
    - linked lists
- caching
- design decisions with caching in consideration
- misc. vocabulary and setting up for next time

Some new but related vocabulary

Spatial locality – the tendency for programs/processors to access data in nearby locations to previously accessed locations

Most of our programs do have spatial locality, just because they do things like looping through an array going index by index.

Programs that exhibit spatial locality will benefit from the caching that happens by your computer automatically (the programmer doesn't have to do anything to make this happen).

Some new but related vocabulary

Temporal locality – tendency for program / processor to access one piece of data repeatedly

Just like they did with spatial locality, programs that exhibit temporal locality will benefit from the caching that happens by your computer automatically (the programmer doesn't have to do anything to make this happen). Because you accessed data 'recently', it should be in the cache for you to access quickly the second time around.

Many programs exhibit temporal locality:

```
int result = someMethodCall(…);
for (int i = 0; i < 10; i++) {
    System.out.println(result);
}
```

Here the variable 'result' is accessed repeatedly so after the first time it is loaded it's put into the cache, and afterwards the lookups can just look in the cache

Activity: (3 min) why m2 would be faster than m1? What changes could you make to m1 to make it faster? Discuss!

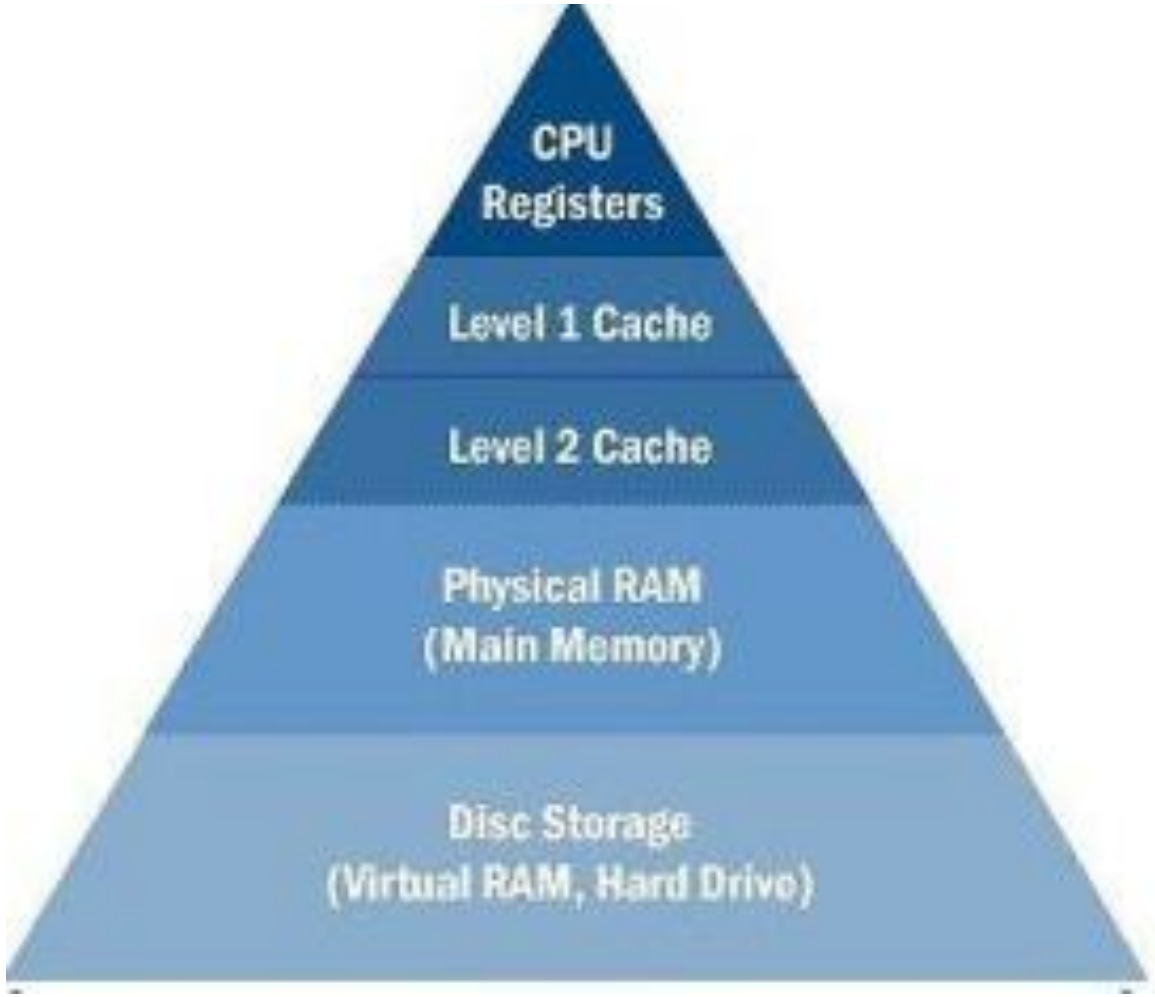
```
public void m1(String[] strings) {
    for (int i = 0; i < strings.length; i++) {
        strings[i] = strings[i].trim();
    }
    for (int i = 0; i < strings.length; i++) {
        strings[i] = strings[i].toLowerCase();
    }
}
```

```
public void m2(String[] strings) {
    for (int i = 0; i < strings.length; i++) {
        strings[i] = strings[i].trim();
        strings[i] = strings[i].toLowerCase();
    }
}
```

M2 exhibits good temporal locality, because we access strings[i] repeatedly so we know strings[i] is in the cache for when we call toLowerCase on the next line. M1 on the other hand … doesn't have a guarantee strings[i] is still in the cache and easy to access.

Remember that caches are small, and so not everything fits into the cache forever. When caches get full and want to store new data, they will evict old data. Often, it's the least recently used data that gets pushed out – see LRU caching

Memory hierarchy, slowness + size increase,going towards RAM and DISK



There are actually multiple caches, and a whole hierarchy of places to store data on your computer. The more memory a layer can store, the slower it is to access.

In a sense, RAM is even a cache for Disk storage.

Main Idea Recap

Main ideas:

- RAM / memory is like an array and knowing how data looks inside RAM
- data needs to go to CPU to be processed
- caches help speed up CPU's need for data from RAM by serving as a closer resource of data
- caching bonuses are usually just for arrays instead of linked lists/node data structures

References / other reading

CSE 351 Caching Slides

Wikipedia pages on CPU, RAM, and Locality

Another useful metaphor for locality and caching: https://medium.com/@adamzerner/spatial-and-temporal-locality-for-dummies-b080f2799dd

https://www.youtube.com/watch?v=fpnE6UAfbtU