Lecture 5

Vectors

- Speedrunning Input
- Vector data structure
- `Vector` in `structure`

Speedrunning Input

A *tool-assisted speedrun* of Tetris played on Nintendo Entertainment System (NES) hardware by TASBot at the Games Done Quick charity marathon in Summer 2019.

Capturing Input

In this lecture, we'll start by considering the programming problem of capturing input in a retro video game emulator.

The NES controller has 8 buttons.

The bottom-left of the screen shows that the SELECT and A buttons are pressed on this frame.

Then we'll consider the problem as a data structure designers, and take a look at the `Vector.java` program in `structures`.

Goals and Questions as Programmers

Goals

1. The program captures one *data point* (i.e., the button presses) per frame.
2. No limit on the number of data points to be stored (except for the user's memory).
3. The program should not allocate much more space than it may need.
4. The program shouldn't lag too much.
5. The user can pause and rewind and see the data at any frame (instantly).
6. The user can replay their gameplay from any frame (without recording a movie).

Point 5. suggests that we need to use an array, but points 2. and 3. suggest that the array must be resizable, which is not standard.

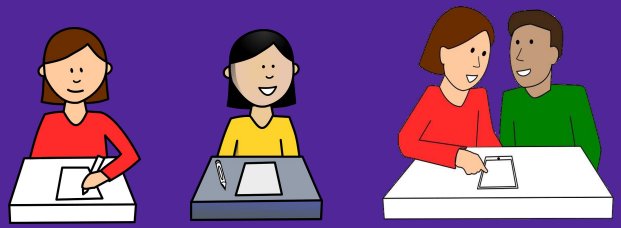
Questions

- How can we store a single data point? In other words, what is stored each frame?
- How large should we make the array initially?
- When the array is full, how much larger should we make it?
- Will it be possible to avoid lag using an array-based approach?

Discussion: Designing our Program

Let's discuss how we can design and implement input capturing in our program.

Some of the questions from the previous slide appear below.



Think on your own for 2 minutes.
Discuss your ideas with a neighbor for 3 minutes.

- How can we store a single data point? In other words, what is stored each frame?
- How large should we make the array initially?
- When the array is full, how much larger should we make it?

Vector Data Structure

Vector Data Structure

A *vector* data structure functions like an array that resizes itself. It typically supports the following:

- New elements can be added at the end.
- Elements can be accessed based on their index.

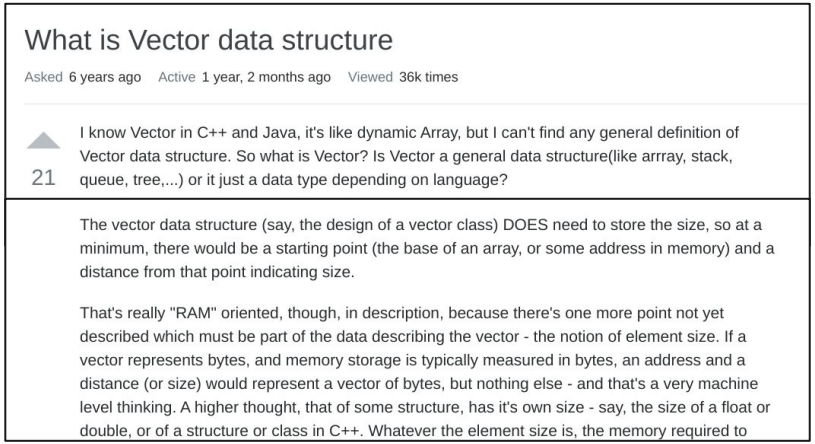
Implementation decisions must be made:

- If the data is stored internally in an array, then when and how should we resize it? A common approach is to double the capacity when it fills.

The term *vector* is borrowed from mathematics.

- Vectors have a starting point and continue some distance in a single direction.

(Honestly, it isn't the great name for the data structure!)



What is Vector data structure

Asked 6 years ago Active 1 year, 2 months ago Viewed 36k times

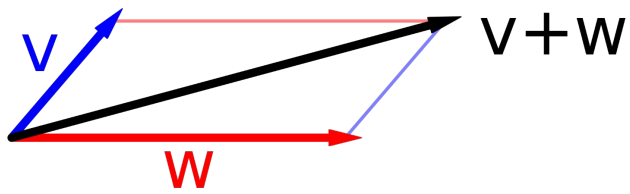
21

I know Vector in C++ and Java, it's like dynamic Array, but I can't find any general definition of Vector data structure. So what is Vector? Is Vector a general data structure(like arrray, stack, queue, tree,...) or it just a data type depending on language?

The vector data structure (say, the design of a vector class) DOES need to store the size, so at a minimum, there would be a starting point (the base of an array, or some address in memory) and a distance from that point indicating size.

That's really "RAM" oriented, though, in description, because there's one more point not yet described which must be part of the data describing the vector - the notion of element size. If a vector represents bytes, and memory storage is typically measured in bytes, an address and a distance (or size) would represent a vector of bytes, but nothing else - and that's a very machine level thinking. A higher thought, that of some structure, has it's own size - say, the size of a float or double, or of a structure or class in C++. Whatever the element size is, the memory required to

A nice discussion of Vectors on Stack Overflow:
stackoverflow.com/questions/32541194/what-is-vector-data-structure



Mathematical vectors.

`Vector` in `structure`

GNU nano 5.8 Vector.java

```
// An implementation of extensible arrays.
// (c) 1998, 2001 duane a. bailey

package structure;
import java.util.Iterator;
import java.util.Collection;

/**
 * An implemention of extensible arrays, similar to that of java.util.Vector.
 *
 * This vector class implements a basic extensible array.  It does not implement
 * any of the additional features of the Sun class, including list-like operations.
 * Those operations are available in other implementors of {@link List} in this
 * package.
 * <p>
 * Example usage:
```

Let's look at the implementation of a Vector in the `structure` package.

- File location: `~/cs136/js/src/structure/Vector.java`

Using `Vector` for our Application: Pros, Cons, Questions

Pros

- The class supports our basic requirements (i.e. `add`).
- Array doubling is implemented, so our program will be much simpler.
- The iterator can be used to replay the gameplay from the beginning.

Cons

- There will be lag every time the array is resized. Moreover, the lag time will increase the longer the game is played.
- The iterator cannot be used to replay the gameplay from any given frame.
  - We can create a derived class that inherits from `Vector`, and adds a new iterator.

Questions

- The `Vector` class has more functions than we need. Is this a problem?
- What other applications might use the `Vector` class?
- Can we formally analyze the efficiency of our "doubling" approach?
- If you were designing a Vector data structure, then would you make any different choices?

The design and use of data structures is an important part of your development as computer scientists.