CSCI 136: Data Structures and Advanced Programming

Lecture 12

Asymptotic analysis, part 3

Instructor: Dan Barowy

Williams

Outline

Study tip

Proof: doubling is good strategy

Interfaces

Inheritance

Announcements

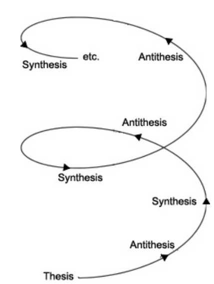
Feedback: How should I study for midterm?

Grades will be determined as follows:

Final exam:	20%
Midterm exam:	20%
Programs/Labs:	35%
Code reviews:	10%
Engagement:	15%

Life skill #9

Experimentation



1. A **thesis** is an intellectual proposition (i.e., a **T/F** statement).
2. An **antithesis** is an alternative proposition using the same facts.
3. **Synthesis** reconciles the two hypotheses by gathering new facts and proposing a new thesis.

Life skill #9

Thesis: **You tell me**. How should you study?

Life skill #9

Antithesis:

1. Revisit your **glossary** of terms.
2. Revisit **homework**. What did you get wrong?
3. What **activities** have we done in class?
   1. Expect to have to **code on paper**.
   2. Expect to have to **prove inductively**.
4. **Practice problems** in book (you have solutions!)

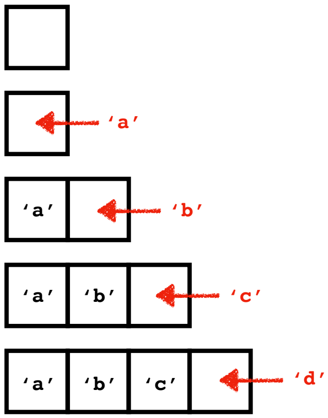
Life skill #9

Synthesis: How should you study?

From last class

Why is the **array doubling** strategy for Vector **better** than expanding the array **one element at a time**?

One-at-a-time expansion



Initial array.

Insert element.

New array; **copy** previous; **insert** element.

New array; **copy** previous; **insert** element.

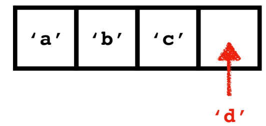
New array; **copy** previous; **insert** element.

...

...

Insertion into an array

How much does **array insertion** cost?



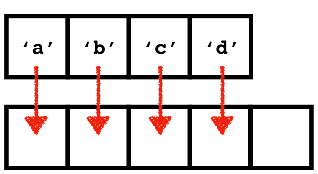
It costs **O(1)**.

In fact, lookup and insertion both cost **O(1)**.

Tradeoff: arrays are fixed size.

Copying an array

How much does an **array copy** cost?

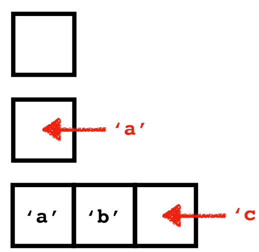


It costs **O(1)** × **m**, where **m** is the size of the original array.

≈ **O(m)**

One-at-a-time expansion costs?

(in the worst case, each time)



Initial array.

Insert element.

New array; **copy** previous; **insert** element.

O(m) + **O(1)** ≈ **O(m)**, where **m** is the size of the original array.

Cost is **dominated by the size of the array** being copied.

How many copies?

of copies for one-at-a-time expansion:

`add()`	1	+	2	+	3	+	…	+ (n-1)
	2nd elem.		3rd elem.		4th elem.			nth elem.

Recall theorem: $1 + 2 + 3 + \ldots + k = k(k+1)/2$

Sub n-1 for k: $(n-1)((n-1)+1)/2 = n(n-1)/2$

$= n^2/2-n/2$

One-at-a-time expansion costs ≈ **O(n²)**

How many copies?

of copies for doubling expansion:

`add()`	1	+	2	+	4	+	…	+ (n/2)
	up to 2nd elem.		up to 4th elem.		up to 8th elem.			up to nth elem.

Neat theorem: $1 + 2 + 4 + \ldots + 2^{k-1} = 2^k-1$

Suppose $n = 2^k$.

Then $1 + \ldots + n/2 = 1 + \ldots + 2^k/2$

$= 1 + \ldots + 2^{k-1} = 2^k-1 = n-1$

Doubling expansion costs ≈ **O(n)**

Which is faster?

💩 One-at-a-time expansion costs ≈ **O(n²)** 💩

😎 Doubling expansion costs ≈ **O(n)** 😎

Doubling is Vin Diesel-approved.

Interfaces

Interface

An **interface** defines boundary between two systems across which they share information. An interface is a **contract**: calling a method defined in an interface returns the data as promised.

A key principle of object-oriented design is to **deny access** to all data (i.e., to make `private`) by default, allowing access only through methods specified by the interface.

(code)

Inheritance

Inheritance

Inheritance is a **mechanism** for defining a class in terms of another class. It is a labor-saving device employed to reduce **code duplication**. Inheritance allows programmers to specify a new implementation while :

1. **maintaining the same behavior**,
2. **reusing code**, and
3. **extending the functionality** of existing software.

(code)

Recap & Next Class

Today we learned:

Sample Big-O analysis

Interfaces

Inheritance

Next class:

Sorting