Analysis of Algorithms

V. Adamchik

Lecture 2

Fall 2023

CSCI 570

University of Southern California

Review

Amortized Cost

Reading: chapters 1 & 2

Ch1: review questions

2. (**T/F**) Any function which is $\Omega(\log n)$ is also $\Omega(\log(\log n))$.

3. (**T/F**) If $f(n) = \Theta(g(n))$ then $g(n) = \Theta(f(n))$.

Ch1: exercises

4. Arrange the following functions

$$4^{\log n},\ \sqrt{\log n},\ n^{\log\log n},\ (\sqrt{2})^{\log n},\ 2^{\sqrt{2\log n}},\ n^{1/\log n},\ (\log n)!$$

in increasing order of growth rate with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$.

$4^{\log n} = n^2$

$n^{\log\log n} > n^{1/\log n}$

$\log\log n \times \log n > 1$

$(\sqrt{2})^{\log n} > 2^{\sqrt{2\log n}}$

log on both sides

$\log n \cdot \log\sqrt{2} > \sqrt{2\log n} \cdot \log 2$

$\log n \cdot \log\sqrt{2} > \sqrt{2\log n}$

Discussion Problem 1

Consider these two statements about a connected undirected graph with V vertices and E edges:

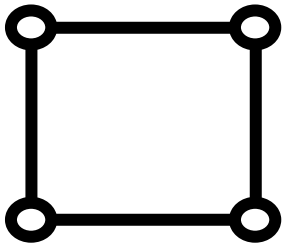
I. $O(V) = O(E)$
II. $O(E) = O(V^2)$

Mark all the correct choices below

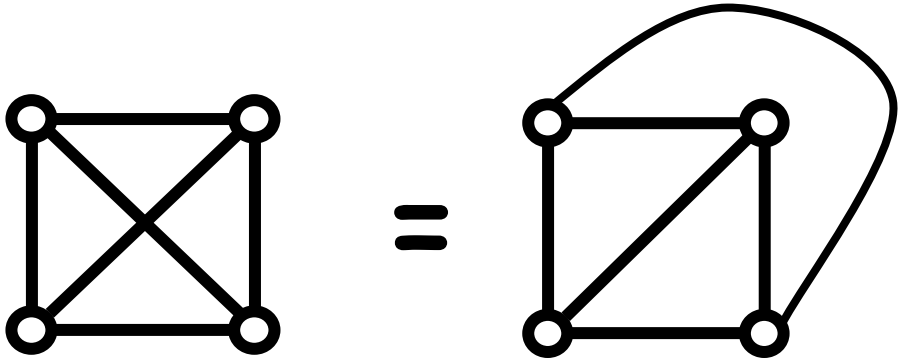
(a) I and II are both false.
(b) Only I is true.
(c) Only II is true.
(d) I and II are both true.

Planar Graphs

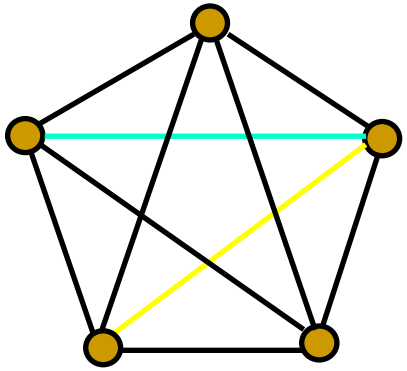
A graph is planar if it can be drawn in the plane without crossing edges

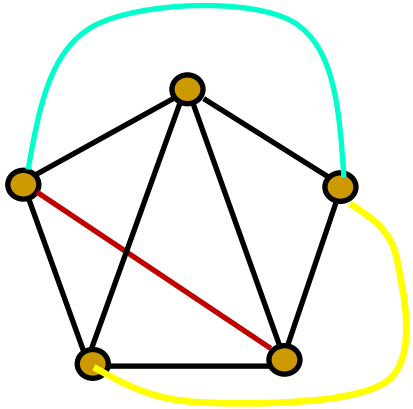


K_4 is planar



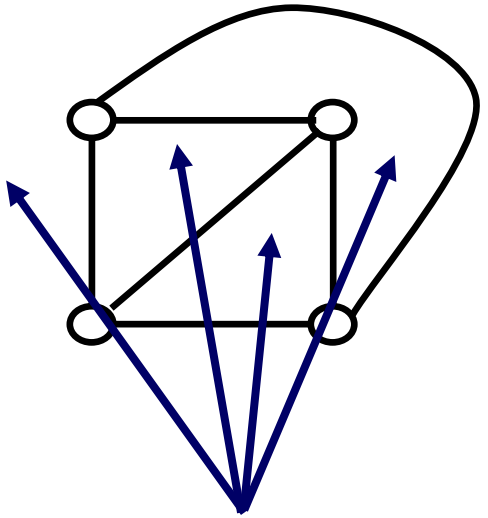
K_5 is not planar





Euler's Formula

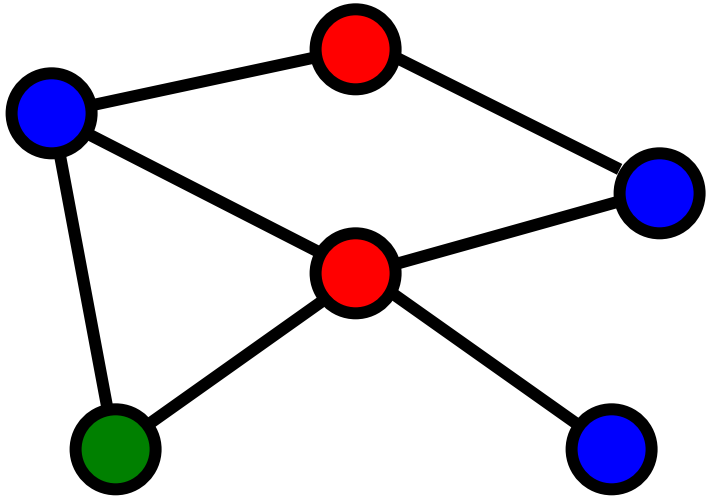
Theorem. If G is a connected planar graph with V vertices, E edges and F faces, then $V - E + F = 2$.



A planar graph when drawn in the plane, splits the plane into disjoint faces.

Coloring Planar Graphs

A coloring of a graph is an assignment of a color to each vertex such that no neighboring vertices have the same color



4 Color Theorem (1976)

Theorem: *Any simple planar graph can be colored with less than or equal to 4 colors.*

It was proven in 1976 by K. Appel and W. Haken. They used a special-purpose computer program.

Since that time computer scientists have been working on developing a formal program proof of correctness. The idea is to write code that describes not only what the machine should do, but also why it should be doing it.

If a graph is NOT planar, the coloring problem is hard (NP-hard)

Amortized Analysis

In a sequence of operations, the worst-case does not occur often in each operation - some operations may be cheap, some may be expensive.

Therefore, a traditional worst-case per operation analysis can give overly *pessimistic* bound.

When same operation takes different times, how can we accurately calculate the runtime complexity?

Unbounded Array

Consider insertions into an array of size n:

some operations take O(n), others – O(1)

If the current array is full, the cost of insertion is linear;
if it is not full, insertion takes a constant time.

Amortized analysis is an alternative to the traditional worst-case analysis. Namely, we perform a worst-case analysis on a sequence of operations.

We will create a new data structure – unbounded array: when we need to insert into a full array, we allocate a new array twice as large and copy the elements we already have to the new array.

The Aggregate Method

The aggregate method computes the upper bound $T(n)$ on the total cost of n operations.

The amortized cost of an operation is given by $\frac{T(n)}{n}$

In this method each operation will get the same amortized cost, even if there are several types of operations in the sequence.

Unbounded Array

Insert	Old size	New size	Copy
1	1	—	—
2	1	2	1
3	2	4	2
4	4	—	—
5	4	8	4
6	8	—	—
7	8	—	—
8	8	—	—
9	8	16	8

Unbounded Array

Unbounded Array: summary

The amortized cost of an operation is given by $\frac{T(n)}{n}$, where $T(n)$ is the upper bound on the total cost of *n* operations.

We considered unbounded array with a doubling-up resizing policy

Insertions: 1, 2, 3, 4, 5, 6, 7, 8, 9, …, 2^n+1
Insertion Cost: 1, 1, 1, 1, 1, 1, 1, 1, 1, …, 1
Copy Cost: 0, 1, 2, 0, 4, 0, 0, 0, 8, … , 2^n

We computed the average cost per insert: O(1)

It is important to realize that we achieve a great amortized cost just because we have implemented a clever resizing policy!

Binary Counter

Given a binary number n with $\log(n)$ bits, stored as an array, where each entry $A[i]$ stores the i-th bit.

The cost of incrementing a binary number (binary addition) is the number of bits flipped.

	# of flips
000	1
001	2
010	1
011	3
100	1
101	2
110	1
111	3

Binary Counter

Discussion Problem 2

Another Binary Counter. Let us assume that the cost of a flip is 2^k to flip k-th bit. Flipping the lowest-order bit costs $2^0 = 1$, the next bit costs $2^1 = 2$, and so on. What is the amortized cost per increment? Use the aggregate method.

Discussion Problem 3

Another Yet Binary Counter. Let us assume that the cost of a flip is (k+1) to flip k-th bit. Flipping the lowest-order bit costs 0 + 1 = 1, the next bit costs 1 + 1 = 2, the next bit costs 2 + 1 = 3, and so on. What is the amortized cost per operation for a sequence of *n* increments, starting from zero? What is the amortized cost per increment? Use the aggregate method.

The Accounting Method

The accounting method (or the banker's method) computes the individual cost of each operation.

We assign different charges to each operation; some operations may charge more or less than they actually cost.

The amount we charge an operation is called its amortized cost.

Discussion Problem 4

You have a stack data type, and you need to implement a FIFO queue. The stack has the usual POP and PUSH operations, and the cost of each operation is 1. The FIFO has two operations: ENQUEUE and DEQUEUE.

We can implement a FIFO queue using two stacks. What is the amortized cost of ENQUEUE and DEQUEUE operations?