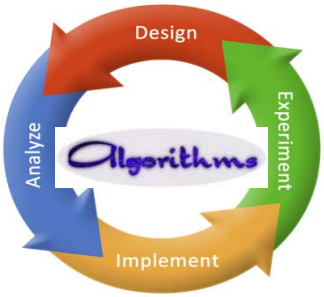
Analysis of Algorithms

V. Adamchik CSCI 570 Spring 2018

Lecture 2 University of Southern California



Plan

Computational Complexity

Topics in Graph Theory

Proofs

Al-Khwarizmi,
Persian astronomer
and mathematician

CS versus IT

Two rewarding careers…

IT:

installing, organizing, maintaining computer systems

CS:

efficient programming using mathematical algorithms

Math versus CS

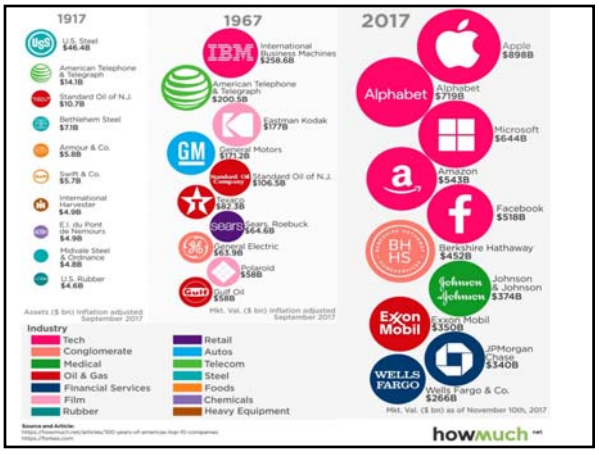
Two rewarding careers…

Math:

- based on abstractions and relation between them
- generally an absolute truth

CScience:

- the basis is empiricism (derived from experience).
- creates a computer model of the natural world



Best Global Universities for Computer Science

#1 Tsinghua University
China Beijing
#64 – Best Global Universities

#18 University of Southern California
United States Los Angeles, CA
#62 – Best Global Universities

#19 Princeton University
United States Princeton, NJ
#9 – Best Global Universities

#18 in the world!!!

Learning tips

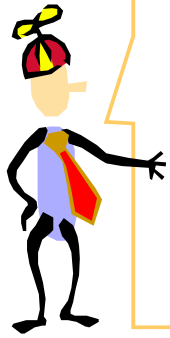
Don't be a *passive* learner

Have your paper and pencil handy

Draw your own pictures

Ask questions

Review my PPT slides frequently ☺



Review

Runtime Complexity

Runtime Complexities

The term analysis of algorithms is used to describe approaches to study the performance of computer programs. In this course we will perform the following types of analysis:

- the worst case complexity
- the best case complexity
- the average case complexity
- the amortized time complexity

We measure the run time of an algorithm using following asymptotic notations: O, Ω, Θ.

Runtime Complexities

Insertion Sort:

- the best case complexity is $O(n)$
- it happens when the input is sorted
- how often does it happen? 2/n! - rarely

Quick Sort:

- the worst case complexity is $O(n^2)$
- the average case complexity is $O(n \log n)$

Upper Bound

For any monotonic functions f, g from the positive integers to the positive integers, we say

$f(n) = O(g(n))$

if

g(n) eventually dominates f(n)

Formally: there exists a constant c such that for all *sufficiently large* n: $f(n) \leq c*g(n)$

Another useful notation: Ω

For any monotonic functions f, g from the positive integers to the positive integers, we say

$f(n) = \Omega(g(n))$

if:

f(n) eventually dominates g(n)

Formally: there exists a constant c such that for all *sufficiently large* n: $f(n) \geq c*g(n)$

More useful notation: Θ

For any monotonic functions f, g from the positive integers to the positive integers, we say

$$f(n) = \Theta(g(n))$$

if:

$$f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))$$

In this class we will be mostly concerned with a big-O notation.

Quickies

1. $n = O(n^2)$? T
2. $n = O(\sqrt{n})$? F
3. $\log n = \Omega(n)$? F
4. $n^2 = \Omega(n \log n)$? T
5. $n^2 \log n = \Theta(n^2)$? F
6. $3n^2 + 4n + 5 = \Theta(n^2)$? T
7. $2^n + 100n^2 + n^{100} = O(n^{101})$? F
8. $(1/3)^n + 100 = O(1)$? T

T(n) = n log n + 1024 n log (log n)

Circle **ALL** answers that apply.
Is $T(n) = O(n^2)$?

$O(n^2)$	$O(n \log n)$	$O(n \log \log n)$	$O(n)$	$O(\log n)$

T(n) = n log n + 1024 n log (log n)

Circle **ALL** answers that apply.
Is $T(n) = \Omega(n^2)$?

$\Omega(n^2)$	$\Omega(n \log n)$	$\Omega(n \log \log n)$	$\Omega(n)$	$\Omega(\log n)$

T(n) = n log n + 1024 n log (log n)

Circle **ALL** answers that apply.
Is $T(n) = \Theta(n^2)$?

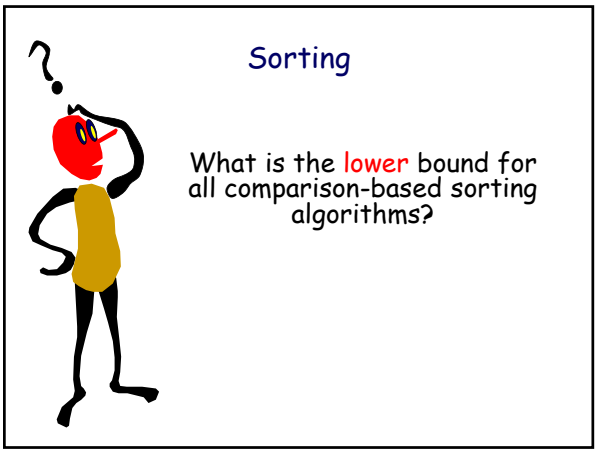
$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log \log n)$	$\Theta(n)$	$\Theta(\log n)$

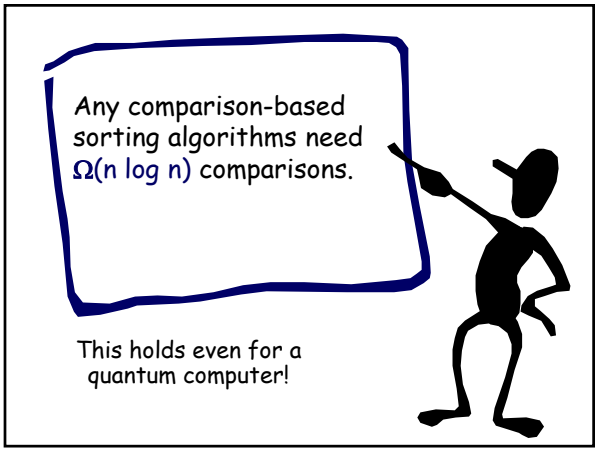
What is the Big-O runtime complexity of the following function?

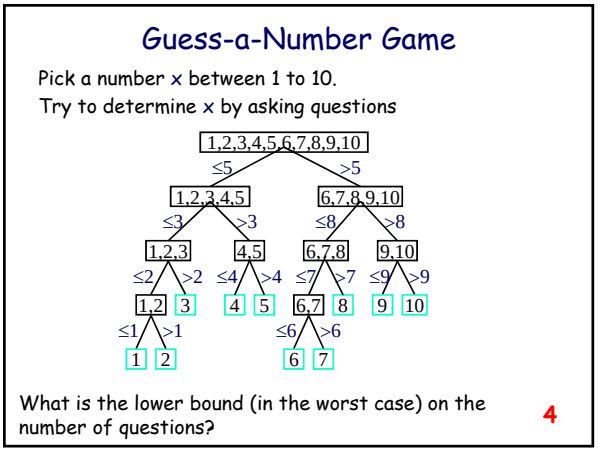
```
void bigOh (int n):
  for i=1 to n
     j=1;
     while j < n
          j = j*2;
```

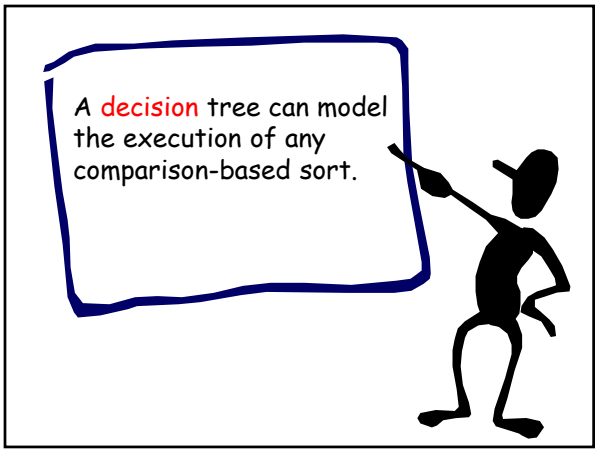
O(n log n)

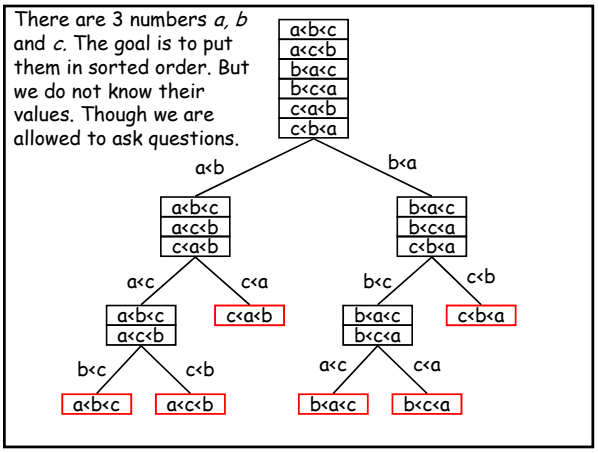

Review: Sorting


Sorting
What is the lower bound for all comparison-based sorting algorithms?


Any comparison-based sorting algorithms need Ω(n log n) comparisons.
This holds even for a quantum computer!


Guess-a-Number Game
Pick a number x between 1 to 10.
Try to determine x by asking questions
1,2,3,4,5,6,7,8,9,10
≤5
>5
1,2,3,4,5
6,7,8,9,10
≤3
>3
≤8
>8
1,2,3
4,5
6,7,8
9,10
≤2
>2
≤4
>4
≤7
>7
≤9
>9
1,2
3
4
5
6,7
8
9
10
≤1
>1
≤6
>6
1
2
6
7
What is the lower bound (in the worst case) on the number of questions?
4

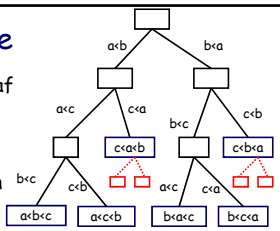

A decision tree can model the execution of any comparison-based sort.


There are 3 numbers a, b and c. The goal is to put them in sorted order. But we do not know their values. Though we are allowed to ask questions.
a<b<c
a<c<b
b<a<c
b<c<a
c<a<b
c<b<a
a<b
b<a
a<b<c
a<c<b
c<a<b
b<a<c
b<c<a
c<b<a
a<c
c<a
b<c
c<b
a<b<c
a<c<b
c<a<b
b<a<c
b<c<a
c<b<a
b<c
c<b
a<c
c<a
a<b<c
a<c<b
b<a<c
b<c<a

Decision Tree

In the decision tree each leaf represents a permutation of the n numbers.

Hence, there are n! leaves in this tree.



Now make this tree complete. Assume the height is h.

The number of leaves in it is 2^h. Thus, $2^h \geq n!$.

But h is the number of comparisons, so h = T(n) is runtime.

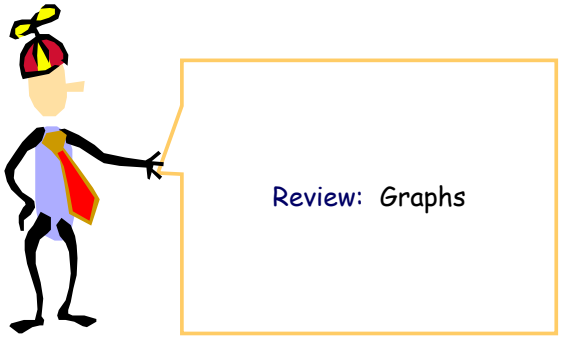
Hence, $2^{T(n)} \geq n!$. Or, $T(n) \geq \log(n!)$.

It follows (see the next slide), $T(n) = \Omega(n \log n)$

Worst-Case Running Time

$$\begin{aligned}\log n! &= \log[n(n-1)(n-2)\ldots 1] \\ &\geq \log[n(n-1)(n-2)\ldots(n-n/2)] \\ &\geq \log[(n/2)^{n/2}] \\ &= \frac{n}{2}\log\frac{n}{2}\end{aligned}$$

$$\log n! = \Omega(n \log n)$$



Review: Graphs

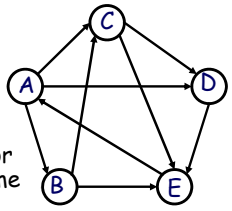
Definition

A graph G is a pair (V,E) where V is a set of vertices (or nodes) E is a set of edges connecting the vertices.

A self-loop is an edge that connects to the same vertex twice.

A multi-edge is a set of two or more edges that have the same two vertices.

A graph is simple if it has no multi-edges or self-loops.



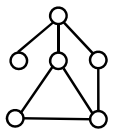
More terms

Directed: an edge is an ordered pair of vertices
Undirected: edge is unordered pair of vertices
Weighted: (a cost associated with an edge)
Path: (is a sequence of vertices)
Simple Path: (no vertices repetition)
Cycle: (the start and end vertices are the same)
Acyclic: (no cycles)
Connected or Disconnected: ()
The degree of a vertex (in an undirected graph is the number of edges associated with it.)

The handshaking theorem

Let G=(V,E) be an undirected graph with V vertices and E edges. Then

$$2E = \sum_{x \in V} \deg(x)$$



In a directed graph:

$$E = \sum_{x \in V} \text{indeg}(x) = \sum_{x \in V} \text{outdeg}(x)$$

Exercise

Given a graph with 7 vertices; 3 of them of degree two and 4 of degree one. Is this graph is connected?

$$2E = \sum_{x \in V} \deg(x)$$

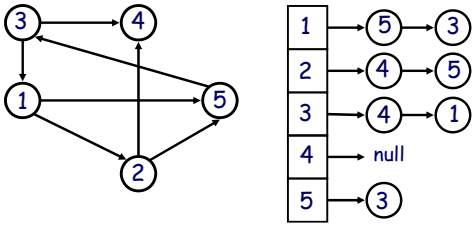
No, the graph has only 5 edges.

Representing Graphs

Adjacency List
or
Adjacency Matrix

Vertex X is *adjacent* to vertex Y if and only if there is an edge (X, Y) between them.

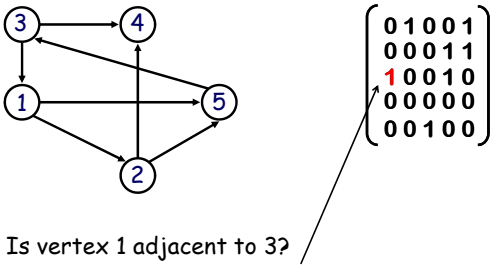
Adjacency List Representation



Is vertex 1 adjacent to 3?

It takes linear time to figure it out.

Adjacency Matrix Representation



Is vertex 1 adjacent to 3?

It takes constant time to figure it out.

Exercise

What is the maximum number of edges in a simple undirected graph with V vertices?

$$\frac{V(V-1)}{2}$$

Such graph is called complete.

Representing Graphs

Adjacency List Representation is used for representation of the sparse ($E = O(V)$) graphs.

Adjacency Matrix Representation is used for representation of the dense ($E = \Omega(V^2)$) graphs.

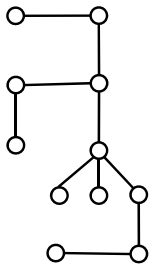
Explain-!

Is Facebook social graph sparse or dense?

We can say a connected graph is maximally sparse if it is a tree.

We can say a graph is maximally dense if it is complete.

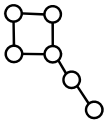
Trees



A tree is a connected simple graph without cycles.

Not Tree

Not Tree



Theorem: Let G be a graph with V nodes and E edges.

The following are equivalent:

1. G is a tree.
2. Every two nodes of G are joined by a unique path.
3. G is connected and $V = E + 1$.
4. G is acyclic and $V = E + 1$.
5. G is acyclic and if any two non-adjacent nodes are joined by an edge, the resulting graph has exactly one cycle.

To prove this, it suffices to show

$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 1$

We'll just show

$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4$

and leave the rest to the reader

$1 \Rightarrow 2$ 1. G is a tree.

2. Every two nodes of G are joined by a unique path.

Proof: (by contradiction)

Assume G is a tree that has two nodes connected by two different paths:

Then there exists a cycle!

$2 \Rightarrow 3$ 2. Every two nodes of G are joined by a unique path.

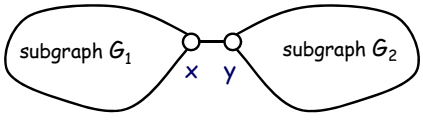
3. G is connected and $V = E + 1$.

Proof $V = E + 1$: (by strong induction)

Base Case: $V = 2$.

Assume true for every graph with $< V$ vertices.

Let G have V nodes and let x and y be adjacent.



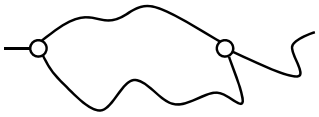
Then by IH: $V = V_1 + V_2 = E_1 + E_2 + 2 = E + 1$

$3 \Rightarrow 4$ 3. G is connected and $V = E + 1$.

4. G is acyclic and $V = E + 1$.

Proof: (by contradiction)

Assume, G has a cycle with k vertices in it.



Start counting nodes and edges in the whole graph. Number of edges in the graph will be at least V, since the cycle has k vertices and k edges.

Corollary: Every nontrivial tree has at least two vertices of degree 1.

Proof (by contradiction):

Assume all but one of the vertices in the tree have degree at least 2.

In any graph, sum of the degrees = 2E.

Under our assumption $2E = \sum deg_i \geq 2(V-1)+1=2V-1$.

Then the total number of edges in the tree is at least $E \geq (2V-1)/2 = V - 1/2 > V - 1$.

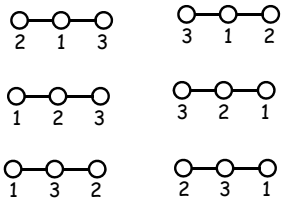
Contradiction, since in a tree $E = V - 1$.

Cayley's Formula

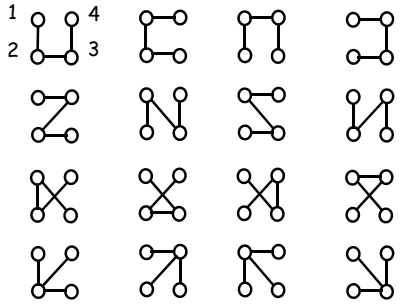
Cayley's formula tells us how many different trees we can construct on n vertices.

How many labeled trees are there with three vertices?

Two labeled trees with the same set of labels are isomorphic iff they have the same adjacency matrix.

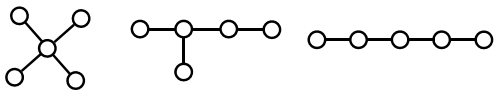


How many labeled trees are there with four nodes?



These are called spanning trees, for a complete graph of 4 vertices.

How many labeled trees are there with five vertices?



5 labelings

5 x 4 x 3 labelings

5!/2 labelings

125 labeled trees

Cayley's Formula (1889)

The number of labeled trees on n vertices is n^{n-2}

Put another way, it counts the number of spanning trees of a complete graph.

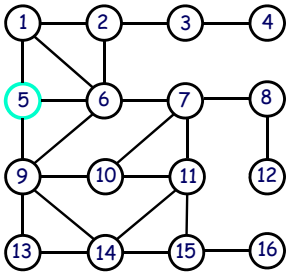
Arthur Cayley (1821-1895)

Graph Traversals

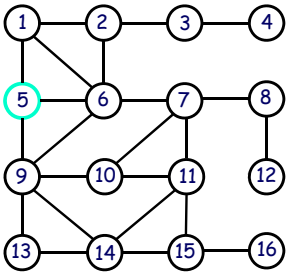
Depth-First-Search (DFS)
Breadth-First-Search (BFS)

DFS uses a stack for bookkeeping.
BFS uses a queue for bookkeeping.

Perform a DFS on the following graph



Perform a BFS on the following graph



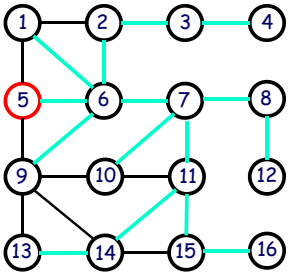
Properties of Traversals

Property 1

They visit all the vertices in the connected component

Property 2

The result of traversal is a spanning tree of the connected component



Exercise

The complete graph on n vertices, denoted K_n, is a simple graph in which there is an edge between every pair of distinct vertices.

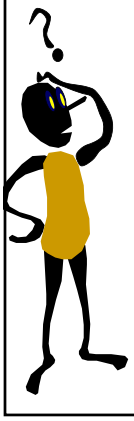
What is the height of the DFS tree for the complete graph K_n?

n-1

What is the height of the BFS tree for the complete graph K_n?

1

Exercise



What is the visual difference between BFS and DFS trees?

DFS trees are tall and skinny.

BFS trees are short and bushy.

Graph Search Algorithms

Depth-First-Search (DFS)
Breadth-First-Search (BFS)
Best-First-Search
Dijkstra's Algorithm
A*-Search
Ant Colony Optimization

Best-First uses a priority queue for bookkeeping.
The algorithm favors vertices that are close to the goal.
Dijkstra's Algorithm favors vertices that are close to the starting point.
A*-Search combines both searches.

Ant Colony Optimization

It's a search technique which is inspired by ants.

- Ants find shortest routes between nest and food.
- They do not use vision.
- They lay pheromone (chemical) on the ground.
- If an ant decides to follow the pheromone trail, it itself marks it with more pheromone.
- The more ants follow the trail, the stronger the pheromone.
- Pheromone decays over time faster on longer routes.

Shortest Path

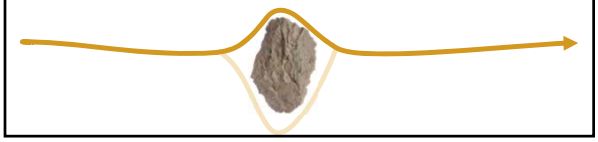
Assume that ants are moving on a straight line from nest to food

An obstacle appears on the path.

Ants have to choose between turning right or left. We can expect half of them will choose to turn right and the other half to turn left.

Shortest Path

Those ants which choose the shorter path will more rapidly recover the interrupted pheromone trail compared to the longer path. Thus, the shorter path will receive a greater amount of pheromone per time unit.



Finding a shortest Hamiltonian cycle

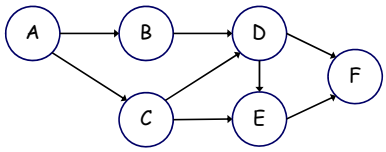
A Hamiltonian cycle is a cycle that visits each vertex exactly once.

- m ants are started at random nodes.
- They traverse the graph randomly, biased to the amount of edge-pheromone and edge-cost.
- An iteration ends when all the ants visit all nodes.
- After each iteration, pheromone trails are updated.
- Solution gets better and better as the number of iterations increase.

Topological Sort for DAG

Suppose each vertex represents a task that must be completed, and an edge (u, v) indicates that task u depends on task v. That is v must be completed before u. The topological ordering of the vertices is a valid order in which you can complete the tasks.

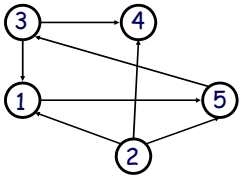
Simple Algorithm



1. Select a vertex that has in-degree zero.
2. Add the vertex to the output.
3. Delete this vertex and all its outgoing edges.
4. Repeat.

A, B, C, D, E, F or A, C, B, D, E, F

Finding a vertex with zero in-degree

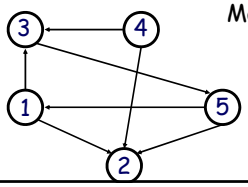


G =

$$\begin{pmatrix} 0&0&0&0&1 \\ 1&0&0&1&1 \\ 1&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&1&0&0 \end{pmatrix}$$

sink

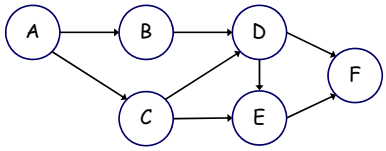
Matrix transposition:



G^T =

$$\begin{pmatrix} 0&1&1&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&1&1&0&0 \\ 1&1&0&0&0 \end{pmatrix}$$

Better Algorithm



- Select a vertex that has in-degree zero.
- Run DFS and return vertices that has no undiscovered leaving edges

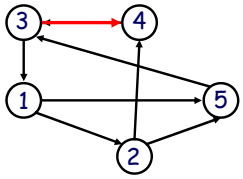
We get vertices in reverse order.

Strongly Connected Graphs

Given a directed graph. It's strongly connected if each vertex is reachable from any other vertex.

If we reverse the edge (3,4), it won't be strongly connected.

It's called a weakly connected graph.



How do you test if a graph is strongly connected?

Strongly Connected Graphs

Brute Force Algorithm: run DFS/BFS from each vertex.

Complexity-? O(V (V+E))

Better Algorithm:

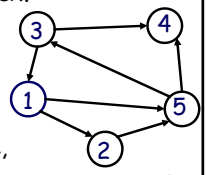
Pick a vertex and run DFS from it.

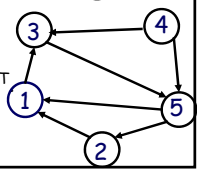
If some vertices are not reachable, stop.

Construct G^T.

Run DFS from the same vertex in G^T

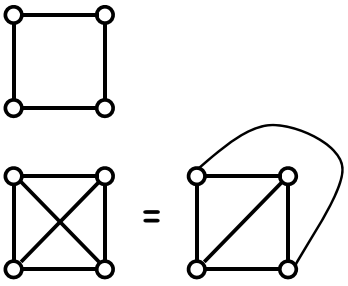
If some vertices are not reachable, stop.





Planar Graphs

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

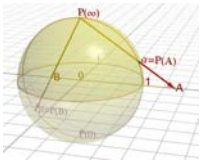


Planar Graphs

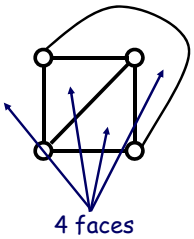
A graph is planar if it can be drawn in the plane without crossing edges

A graph is planar if and only if it can be embedded in a sphere. This is useful because often a sphere is more convenient to work with.

A sphere can be 1-1 mapped (except 1 point) to the plane and vice-versa. E.g. the stereographic projection:



Faces



4 faces

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

Euler's Formula

If G is a connected planar graph with V vertices, E edges and F faces, then

V – E + F = 2

Generalized for any polyhedron.

For a cube:

v=8
e=12
f=6

Proof of Euler's Formula

For connected arbitrary planar graphs V – E + F = 2

The proof is by *induction* on edges.

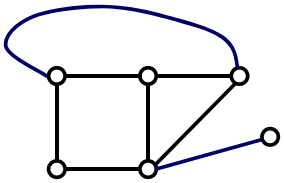
Start with a single edge and 2 vertices: V=2, E=1, F=1. Check.

Add the edges in an order so that a new graph is connected.

There are two cases to consider.

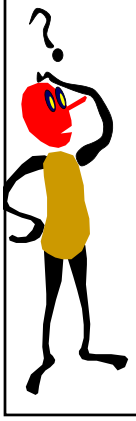
(1) The edge connects two vertices already there.
(2) The edge connects the current graph to a new vertex

In case (1) we add a new edge (E++) and we split one face in two (F++). So V-E+F is preserved.



In case (2) we add a new vertex (V++) and a new edge (E++). So again V-E+F is preserved.

Exercise



What do you think, a planar graph is sparse or dense?

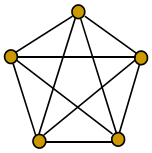
In any connected planar graph with at least 3 vertices: $E \leq 3V - 6$.

Theorem: In any connected planar graph with at least 3 vertices:

$E \leq 3V - 6$

$E = O(V)$

By means of this theorem we can prove, for example, that a complete graph K_5 is not planar



K_5 has 5 vertices and 10 edges, thus

$E = 10 \leq 3 \times 5 - 6 = 9$

which is clearly false

Theorem: In any connected planar graph with at least 3 vertices:

$$E \le 3V - 6$$

Proof.

1. If the graph has no cycles,

$$E = V-1 \le V \le V +(2V-6) = 3V - 6,$$

since $V \ge 3$, and therefore $2V-6 \ge 0$,

Theorem: In any connected planar graph with at least 3 vertices:

$$E \le 3V - 6$$

Proof (cont.)

2. If the graph has a cycle. We will count the number of pairs (edge, face),i.e. $\sum$(edge, face)

Each face is bounded by at least 3 edges:

$$\sum(\text{edge, face}) \ge 3F$$

Each edge is associated with at most 2 faces:

$$\sum(\text{edge, face}) \le 2E$$

It follows, $3F \le 2E$

Theorem: In any connected planar graph with at least 3 vertices:

$$E \le 3V - 6$$

Proof (cont.) We found, $3F \le 2E$

By Euler's theorem :

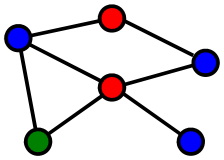
$$2 = V - E + F$$

$$6 = 3V - 3E + 3F \le 3V - 3E + 2E = 3V - E$$

QED

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.

In 2005 such a proof has been developed by Gonthier, using the Coq proof system.

Graph Coloring

Theorem: Any simple planar graph can be colored with 6 colors.

Proof. (by induction on the number of vertices).

If G has six or less vertices, then the result is obvious. Suppose that all such graphs with V-1 vertices are 6-colorable.

Take a graph with V vertices and omit a vertex of degree less than 6. Use IH to color all other vertices.

Now color the omitted vertex, since it has at most 5 adjacent vertices, we have enough colors. QED

Any planar graph contains a vertex of degree less than 6.

$\sum deg(v) = 2E \le 2(3V-6) < 6V$. Now, compute the average degree per vertex: $\sum deg(v) / V < 6$.

Bipartite Matching

A graph is bipartite if the vertices can be partitioned into two disjoint (also called independent) sets V_1 and V_2 such that all edges go only between V_1 and V_2 (no edges go from V_1 to V_1 or from V_2 to V_2)

Personnel Problem. You are the boss of a company. The company has M workers and N jobs. Each worker is qualified to do some jobs, but not others. How will you assign jobs to each worker?

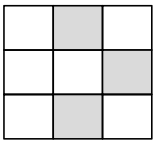
Rook Attack

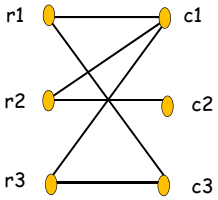
This problem asks us to place a maximum number of rooks (they move horizontally and vertically) on a chessboard with some squares cut out (forbidden positions). Reduce it to bipartite matching.

Rook Attack

Partitions are rows and columns.

We put an edge between r and c if the intersection is not forbidden.





The number of non-attacking rooks equals the number of edges in a matching.

Rook Attack

Assume that we get the matching below in red.

Then we can place rooks in the following squares:
1st col-2nd row and 3rd col-3rd row