Massachusetts Institute of Technology Handout 23
6.042J/18.062J: Mathematics for Computer Science October 05, 1995
Professors David Karger and Tom Leighton

Lecture Notes 9
Graphs

Formally speaking, a graph $G(V, E)$ consists of a set of *vertices* V and a set of *edges* E. In this example

- V=A=1,2,3,4
- E=R=(1,2), (1,3), (1,4) (2,4)

Vertices or nodes usually represent objects of interest, such as people, numbers, cities, computers in a network, etc. In this class, we will restrict our attention to finite graphs, in which V is a finite set.

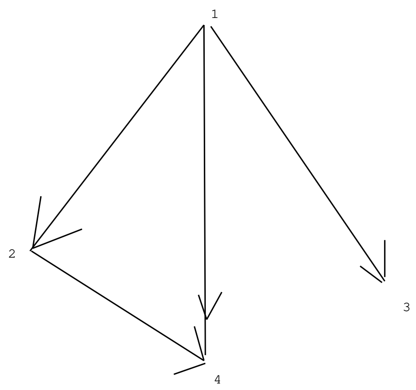

Figure 1: A graph

Edges consist of a pair of vertices. There are two types of edges, directed and undirected edges. If all edges of a graph are directed, then we say that it is a directed graph. Undirected graphs are defined correspondingly. Usually, undirected graphs will represent symetric relations.

Sometimes, graphs will also have special edges, such as loops or multiple edges. Loops will usually represent reflexive relations. In most cases, we will restrict ourselves to graphs without loops and multiple edges, though.

Definition 0.1 *We say that a graph is* simple *if it is undirected and it has neither loops nor multiple edges.*

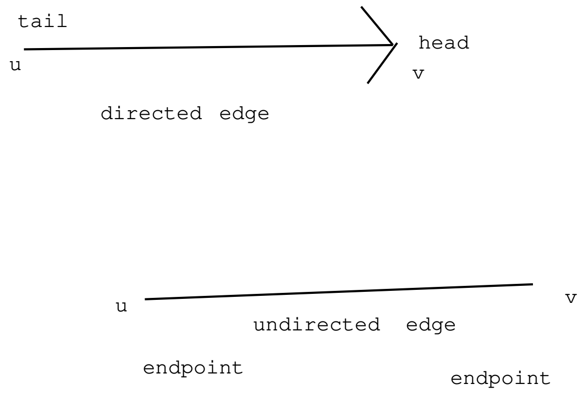

Figure 2: Types of edges

Graphs are used to represent all sorts of computational problems. We have already seen how posets, which are a special type of graphs can be used to represent the task scheduling problem. You will see them much more often in 6.046 and in other Computer Science courses. Today, we'll show you a few simple examples of applications of graphs. We'll also cover some of the most basic techniques from graph theory.

We will start with another scheduling problem that is a bit different from the task scheduling problem from last lecture.

Problem: Given N exams and a set of conflicts, schedule the exams into T time slots without any conflicts, that is, two exams cannot be given at the same time.

The exam scheduling problem can be easily represented in terms of a conflict graph. Nodes represent the exams and two nodes are joined by an edge if the corresponding exams cannot be held at the same time.

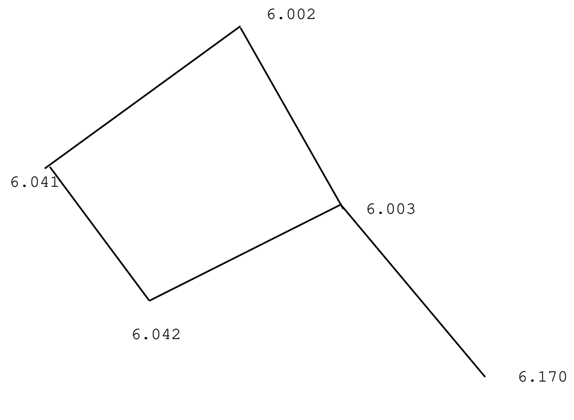

Figure 3: Conflict graph

Graph representation:

- V=Exams
- E=(u,v) — u and v conflict

- T=3 Wednesday 5-7, 7-9, 9-11

In general, trying to figure out if you can color a graph with T colors can takee a long time. It's a classic example of a problem for which no fast optimal algorithms are known. However, in some cases, we can get good upper bounds on teh number of the colors that are needed.

For example, if d is the maximum degree of any vertex, than we can easily find a coloring with only d+1 colors.

Definition 0.2 *The degree of a vertex v in an undirected graph is the number of edges incident to v.*

Thus in our example the degree of 6.042 was 3.

Theorem 0.3 *Any graph with maximum node-degree d is (d+1)-colorable.*

Proof. By induction on the number n of nodes.

- Inductive Hypothesis: P(n)= Any n-vertex graph with maximum node-degree d is (d+1)-colorable.
- P(1) is true as the empty graph is 1-colorable.
- Show that for all n, $P(n) \Rightarrow P(n+1)$ Assume P(n) and prove P(n+1). Let G be any (n+1)-node graph with maximum degree d. Select any node u and remove it to create G'=G-u. Now G' has n nodes and maximum degree at most d. Thus, by P(n), it is (d+1)- colorable. All we need to show to complete our proof is that we can find a color for u, but this is obvious, as the degree of u is at most d, so by reverse pigeon-hole principle, there must be a color we didn't use when we colored the neighbors of u. Thus G is (d+1)-colorable, so P(n+1) holds and the proof is complete.

■ Sometimes (d+1) colors is the best you can do. This bound is actually optimal for the complete graph K_n shown in the figure, as all of its nodes must have different colors. In some other cases, this bound is very far from the optimum, as in the star-graph shown below:

- Coloring problems come up in all sorts of applications. For example, if you have several chemicals to store, you may want to put jars with incompatible chemicals on different shelves. You want to know how many shelves you will need. In this case, the nodes will be the jars, edges will represent the incompatibility relations and colors with represent the shelves.
- Another application is map coloring. Here nodes represent the countries of the map and edges are drown between the nodes of adjacent countries. Graphs of this type turn out to be colorable in small number of colors. We'll talk more about this in recitation.

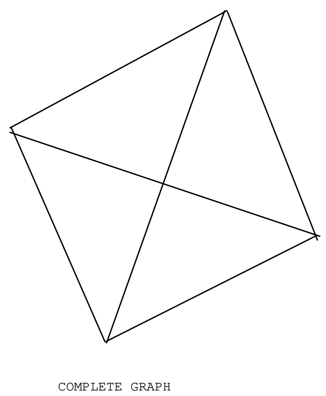

Figure 4: The Complete graph K_4.

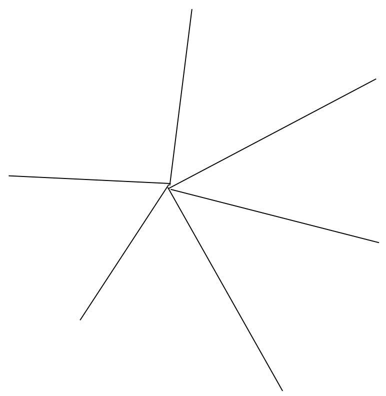

Figure 5: Star

Graphs are often represented by adjacency matrices. For undirected graps, this matrix is symetric. If you have loops, then the diagonal entries are 1.

Adjacency matrices have several uses. One of their nice properties is that by taking powers of the adjacency matrix, you get information about paths in the graph.

For example, the square of the adjacency matrix tells you how many paths of length two exist between any pair of nodes.

This holds true for all graphs. Moreover, if we take the k-th power of the adjacency matrix, we get the number of paths of length k from node i to node j in the entry (i,j) of the matrix.

Theorem 0.4 *For all graphs G and all nodes $i, j \in G$, the (i,j) entry of A^k is the number of paths of length k from node i to node j, where A is the adjacency matrix of G.*

Proof. By induction on k.

- P(k)= For all graphs G and all nodes $i, j \in G$, the (i,j) entry of A^k is the number of paths of length k from node i to node j.
- P(1) is true by the definition of the adjacency matrix.
- $P(k-1) \Rightarrow P(k)$ for all k. Note that $A^k = A^{k-1} \cdot A$. Therefore, by the definiton of matrix multiplication we have:

  $$a_{i,j}^{(k)} = \sum_{m=1}^{n} a_{im}^{(k-1)} a_{mj}$$

  Note that $a_{mj} = 1$ if and only if $(m, j) \in E$, otherwise it is 0. On the other hand, by P(k), $a_{(im)}$ is the number of (i,m) paths of length (k-1). Therefore, the product $a_{im}^{(k-1)} a_{mj}$ equals the number of (i,j) paths of length k whose next-to-last node is m. Thus the number of *all* (i,j)-paths of length k is obtained by taking the sum of these products for all n, which completes the proof of the theorem.

■

This fact is used to compute the transitive closure of any graph.

Definition 0.5 *The transitive closure G^* of a graph G is G with loops added, as well as all edges implied by transitivity.*

Transitive closures are useful when you want to know all possible implications. For example, if nodes are propositions and edges are implications, then G* gives all other deductions you can drow.

The transitive closure is also useful in reachability problems. For example, given a network where the edges describe the plane flights, you can decide whether one city can be reached from another one. (Of course, here the cities are the nodes, the flights are the edges and the transitive closure is the reachability graph.

The following is an algorithm to compute the transitive closure of the n-node graph G:

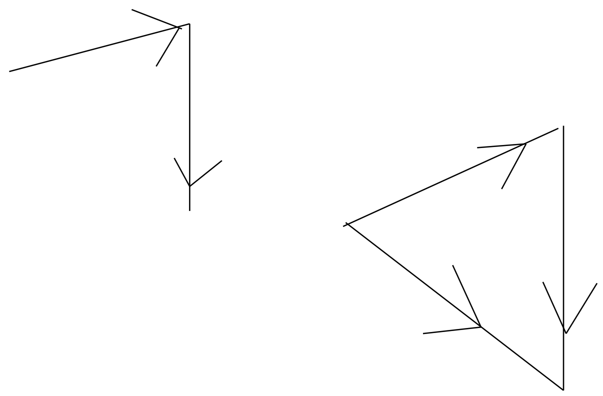

Figure 6: Graph and its closure.

- Insert a loop at every node.
- Compute A^n
- $(i,j) \in E^*$ if and only if $a^n_{i,j} > 0$.

The reason for which we took the n-th power of A is that if there is any path from i to j, then there is one which is not longer than n steps. Indeed, take the shortest (i,j)-path. If that were longer than n steps, then by pigeon-hole principle it would visit some node u twice. But then we could omit the part of the path which lies between the two visit of u, thus we could get a shorter (i,j)-path, contradicting our hypothesis.

Thus indeed, the above algorithm gives us the transitive closure.

To compute the transitive closure, we need to do about $2log(n)$ matrix multiplication (and not n-1) using the fast exponentiation algorithm seen before.

Let us take a look at transitive closures of undirected graphs. We see that that of some of them is the complete graph, that of some others is not.

Definition 0.6 *An undirected graph G is said to be connected if for all pair of nodes (a,b) there is a path connecting a to b.*

Fact: The transitive closure of a connected undirected graph is a complete graph.

When an undirected graph is not connected, then the pieces that are connected are called connected components.

Definition 0.7 *Two nodes u and v are in the same connected components if there is a path between u and v.*

When we take the transitive closure of an undirected graph, the connected components turn into complete subgraphs.

Definition 0.8 *A connected undirected graph with no cycles is called a tree.*

Theorem 0.9 *A tree on n nodes has n-1 edges.*

Proof. BASE CASE: The theorem is true for $n = 2$.

INDUCTIVE STEP: Assuming the theorem true for n, consider the number $n + 1$. Take a tree on (n+1) nodes. It is clear that it must have a node whose degree is one. (Indeed, start a path from some node, then after n steps at most you will reach such a node as the graph has only n+1 nodes and it has no cycles, so you cannot visit any node twice. Thus the path has to stop somewhere and that can only happen in a node of degree 1.) Delete this degree-1 node from the graph, you are left with a cycle-free, connected graph (prove!) on n vertices. This is a tree, so by inductive hypothesis, it has n-1 edges. Therefore, the original graph had (n+1) edges, completing the proof. ∎