Homework 2, CS301, McConnell, Spring '09

Due Tuesday, February 17.

Modified 2/9 to add some problems about inductive definitions.

Readings for Tuesday, 2/10: Sections 2.1 and 2.2; Readings for Thursday, 2/12: Section 2.3

1. (Carried over from Homework 1.) Here is a relation on the *ordered pairs* of integers: $\{((m,n),(i,j))|m+j=n+i\}$. Show that this is an equivalence relation. **The original posting of this problem had a typo that changes the problem.**

2. Let $C(n,k)$ denote n-*choose*-k. Recall from CS161 that this denotes the number of subsets of a set of size n that have size k. For example, $C(4,2)=6$, because the subsets of $\{a,b,c,d\}$ that have size two are $\{a,b\}$, $\{a,c\}$, $\{a,d\}$, $\{b,c\}$, $\{b,d\}$, $\{c,d\}$, and there are six of them. (The choice of elements for the set of size four doesn't affect the argument.)

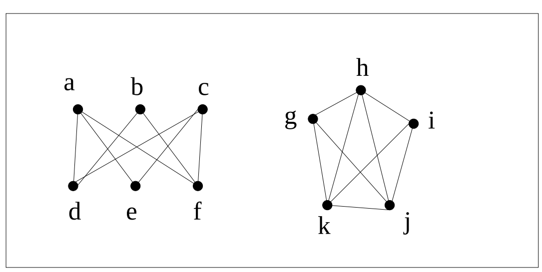
   It was shown in CS161 that $C(n,k)=n!/(k!(n-k)!)$ for $0 \le k \le n$. **If you don't remember the proof, look it up; you will be responsible for knowing it in this class.**

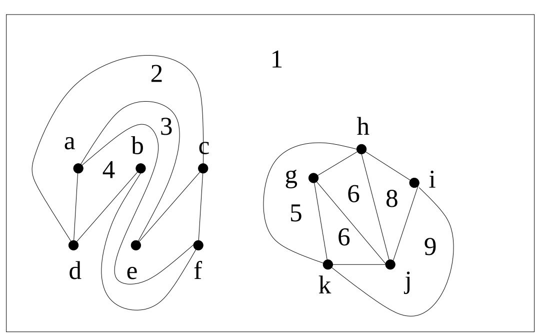
   (a) Argue that $\sum_{i=0}^{n} C(n,i) = 2^n$. *Hint: you don't need to use algebra, just a nice meaning to assign to the lefthand side.*

   (b) Argue that $\sum_{i=0}^{n} C(n,i)C(n,n-i) = C(2n,n)$. *Hint: This is another one that doesn't need algebra. Consider the problem of selecting a group of n people from a set of n men and n women.*

3. A **planar drawing** of a graph is a drawing of it such that no two edges cross and no edge crosses over the top of a vertex.

   For example, the following is a graph and a planar embedding of it.





The **connected components** of an undirected graph are the equivalence classes of Problem 6d from homework 1. The *faces* of a planar drawing are the connected regions of the plane that are left when points lying under vertices and edges are removed. The faces in the drawing are numbered 1 through 9; face 1 is an infinite face.

Show by induction on the number of edges that in any planar drawing, if n is the number of vertices, m is the number of edges, f is the number of faces, and c is the number of connected components, then $n + f = m + c + 1$.

For example, in the drawing, $n = 11$, $m = 17$, $f = 9$ and $c = 2$, and $11+9 = 17+2+1$.

Proceed as follows. Show that the claim holds when $m = 0$, no matter what n is. Then suppose $m > 0$ and adopt the induction hypothesis that the claim holds for planar drawings with $m - 1$ or fewer edges. Given an *arbitrary* planar drawing of a graph with m edges, remove an edge from the drawing to get a planar drawing of a graph with $m - 1$ edges. Consider two cases: the removal of the edge increases the number of connected components or it doesn't.

4. Notice that in the induction proofs above, I started with an arbitrary example of a certain size, and then removed something from it to use the induction hypothesis, then added back the removed element to show that the claim holds for our initial object.

   Some students find this unnatural at first because they are used to starting from an arbitrary example that the induction hypothesis applies to, giving a recipe for constructing a larger instance from it, and showing that the induction hypothesis applies to the larger instance also.

   Here's an example of what can go wrong with this approach.

   Let us show that in a planar drawing of a graph, if n is the number of vertices, m is the number of edges, f is the number of faces, and c is the number of connected components, then $n + f = m + c + 1$. We know this is true from the above exercise, but let's consider a modification of the proof.

   As a base case, let each connected component of the graph be either an isolated vertex or a simple cycle. The number of edges on a cycle is equal to the number of vertices on the cycle. Let i be the number of isolated vertices and k be the number of cycles. There are $i+k$ connected components, so $c = i+k$. There are $n-i$ vertices on cycles, hence $m = n - i$, and there is the infinite face plus one face enclosed by each of the k cycles, so $f = k + 1$. Substituting into $n + f = m + c + 1$, we get $n + (k + 1)$ on the left and $(n - i) + (i + k) + 1$, which are easily seen to be equal using algebra.

   For the induction step, suppose the claim is true for planar drawings of graphs with at most m edges. Now let's show it for graphs with $m + 1$ edges. Consider a planar drawing of a graph with $m - 2$ edges, n vertices, f faces, and c components. Then $n + f = (m - 2) + c + 1$.

   Pick a face, and pick two edges e_1 and e_2 on the face. Insert a new vertex in the middle of e_1, a new vertex in the middle of e_2, and connect these two vertices with a new edge across the face. This adds two to the number of vertices, three to the number of edges, and one to the number of faces, yielding $n' = n + 2$, $m' = m + 1$, $c' = c$, and $f' = f$. Using this, the fact that $n + f = (m - 2) + c + 1$, and algebra, it is easily established that $n' + f' = m' + c' + 1$. Since $m' = m + 1$, the claim follows for $m + 1$.

I claim that this proof fails to prove the result. What's wrong with it? What step was taken in the first proof to avoid any danger of falling into this trap?

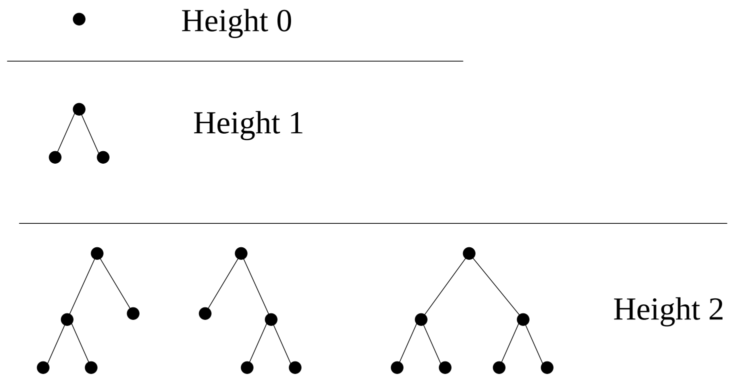
Recall our Version 1, Version 2, and Version 3 abstract models of a computer from lecture on Thursday, February 5. Chapter 2 focuses on finite automata (Version 1). This comes in two sub-versions: deterministic finite automata (Section 2.1) and nondeterministic finite automata (Section 2.2). Though it is not obvious these sub-versions are equivalent to each other, in the sense that neither can compute anything that the other can't also compute. We will prove this, filling in our understanding of the relationships among abstract models of computers.

In addition to their use in understanding the nature of computation, finite automata have many practical uses in compilers and pattern matching in texts.

- Exercise 2.1, p. 54.
- Exercise 2.2, p. 54.
- Exercise 2.3, p. 55.
- Exercise 2.4, p. 55.
- Exercise 2.5, p. 55. For a hint, consider the automaton of Figure 2.7. A number is even if it is 0 (mod 2), and odd if it is 1 (mod 2). How can you turn it into one that recognizes strings where the number a's and b's are both 0 (mod 2), the number of a's and b's are both 1 (mod 2), or the number of a's is 1 (mod 2) and the number of b's is 0 (mod 2).
- Exercise 2.7, p. 55.
- Exercise 2.9, p. 57.
- Exercise 2.11, p. 58.
- **An inductive definition** – full binary trees.

  Let us define a *full binary tree* by induction by height, as follows. As a base case, a full binary tree of height 0 is just a single tree node. For the induction step, a full binary tree of height k consists of a root that has one subtree that is a full binary tree of height $k - 1$ and another subtree that is a full binary tree of height *at most* $k - 1$.

  The following are the full binary trees of heights 0, 1, and 2 that can be constructed using this definition. Notice that the ones of height 2 are obtained by selecting different combinations of ones of lesser height that satisfy the definition. If we have already drawn the trees of height 0 and 1, it is easy to draw those of height 2.



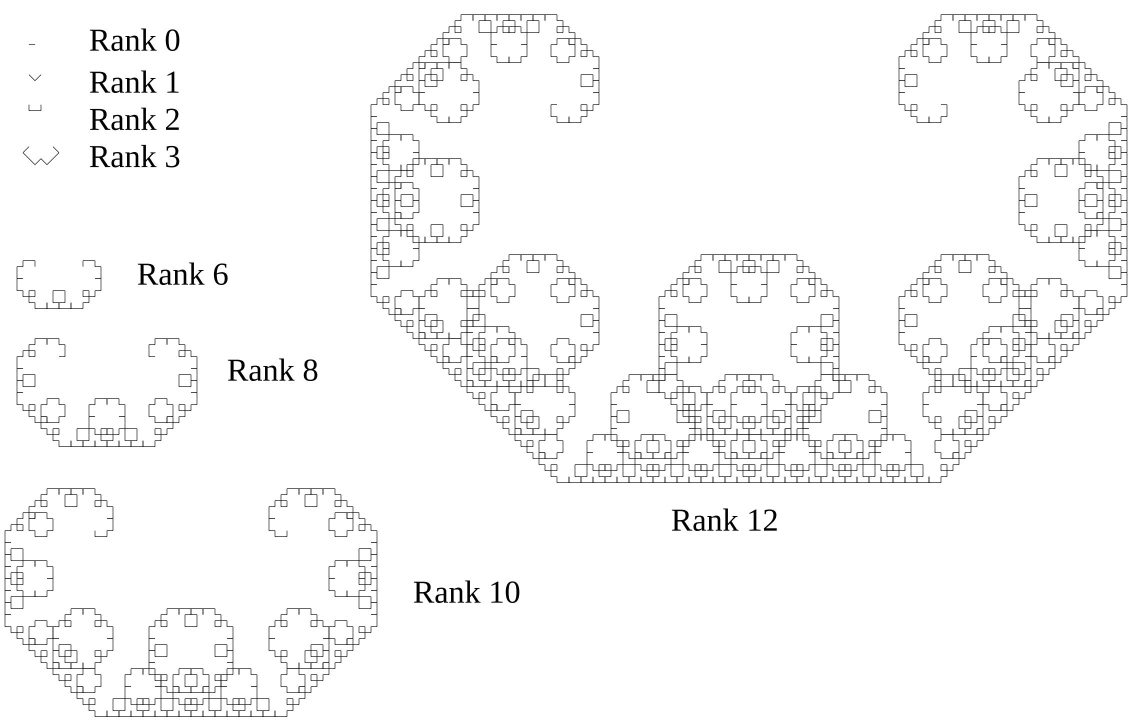
Draw all full binary trees of height 3.

- **An inductive definition** - C-curve.

  A C-curve is a convoluted path that your pencil can follow.

  A C-curve of *rank* 0 is a horizontal line of unit length. A C-curve of *rank* k is obtained by drawing a C-curve of rank $k-1$ rotated clockwise by 45 degrees, and *without lifting your pencil*, drawing another of rank $k-1$ rotated counter-clockwise by 45 degrees.

  The following depicts C-curves of different ranks. Draw the C-curves of ranks 4 and 5.



- **An inductive definition** – Strings of balanced parentheses.

  Base case: The empty string is a string of balanced parentheses. Induction step: A string of balanced parentheses is obtained by one of the following rules:

  1. Concatenate two strings of balanced parentheses;

2. Surround a string of balanced parentheses on either side by two balanced parentheses.

Here are some examples:

1. () – apply rule 2 to the empty string.
2. (()) – apply rule 2 to string 1.
3. (())() – Apply rule 1 to strings 2 and 1, in that order.
4. ((())()) – Apply rule 2 to string 3.
5. ((())())(()) – Apply rule 1 to strings 4 and 2, in that order.

Write out all strings of balanced parentheses of lengths 2, then all strings of balanced parentheses of length 4, then all of length 6, then all of length 8.

Some helpful observations are that all strings of length 8 are obtained by applying rule 2 to a string of length 6, concatenating two strings of length 4, concatenating a string of length 2 and a string of length 6, in that order, or concatenating a string of length 6 and a string of length 2, in that order. It is therefore easy to construct all the examples of length 8 if you have already written out all examples of lengths 2, 4, and 6.