# A problem from Technology Review's Puzzle Corner: Dec 2, 2003

## How many different resistances can be obtained by combining 10 one ohm resistors?

A solution by Joel Karnofsky, 2/23/04

Conceptually this problem is easy: Model a resistor network as a graph with two distinguished nodes, where edges correspond to resistors, nodes to junctions and the distinguished nodes to the points across which the resistance is computed. Then, generate all such 10 edge graphs and compute the resistance for each (using Kirchoff's laws) and collect all the unique values. The practical problem is that the enormous number of such graphs makes a brute force approach unrealistic.

Two ideas will be used to reduce the number of graphs considered. First, a unique normal form will be used to represent each isomorphism equivalence class of graphs, so that redundant graphs can be eliminated as new ones are generated recursively by adding edges. Second, since it is easy to compute the resistance of a graph that can be decomposed into either the serial or parallel combination of two smaller graphs, it is only necessary to generate indecomposable ones.

Some terminology. A graph will always have two distinguished nodes. A subgraph of a graph has the same distinguished nodes. The 0-graph is the graph with two nodes and no edges. The 1-graph has two nodes and one edge connecting them. A parallel decomposition means splitting a graph into two subgraphs whose intersection is the 0-graph and union is the entire graph. A serial decomposition means selecting a non-distinguished node so that the graph can be split into two parts, each of which is a graph with the selected node and one of the original distinguished nodes as its new distinguished nodes and where the intersection of these graphs is just the selected node and their union is the original graph. A d-path is a loop-free path between the distinguished nodes. Two d-paths are said to intersect if they have an interior node in common.

We need to show the following: A serial/parallel indecomposable graph with more than one edge: is connected, has no 1-graph subgraphs and contains a subgraph of the form we will call an h-graph. An h-graph consists of exactly two non-intersecting d-paths and a third, loop-free path connecting interior points on the first two. The simplest h-graph has 5 edges.

A sketch of a proof is: First, if the graph is not connected, it is the parallel combination of its connected components, made into subgraphs by including, if necessary, the distinguished nodes. If it contains a 1-subgraph, it is the parallel combination of this subgraph and its complement. To prove the h-graph property we need two facts:

First, a connected, serial indecomposable graph with no 1-subgraphs contains at least two non-intersecting d-paths. To start, by connected, there is at least one d-path and it has some interior nodes since 1-subgraphs are excluded. It is clear that in a serial indecomposable graph all d-paths cannot have some interior node in common (otherwise the graph could be split into serially connected parts at this node). In particular there must be at least one other d-path. Now consider the two d-paths ($p_1$, $p_2$) that have the fewest intersections at interior points over all pairs of d-paths. If they don't intersect, we are done. Otherwise, starting from one distinguished point, let $n_1$ be the first node of intersection of $p_1$ and $p_2$. By serial indecomposability, there must be a third d-path ($p_3$) that does not pass through $n_1$. If $p_3$ intersects $p_1$ or $p_2$ before $n_1$, let $n_2$ be the last such intersection on $p_3$ and, without loss of generality, assume it is on $p_1$. If the portion of $p_3$ following $n_2$ does not intersect $p_1$ or $p_2$, then the paths $p_2$ and $p_1$ up to $n_2$ followed by $p_3$ after $n_2$ do not intersect and we are done. Otherwise, let $n_3$ be the first intersection of $p_3$ with $p_1$ or $p_2$ after $n_2$ (which must be after $n_1$ on $p_1$ and $p_2$. Without loss of generality (by swapping portions of $p_1$ and $p_2$ after $n_1$), we can assume $n_3$ is also on $p_1$. Now, the paths $p_2$ and $p_1$ up to $n_2$ followed by $p_3$ up to $n_3$ followed by $p_1$ after $n_3$ do not intersect at or before $n_1$ on $p_2$ and $n_3$ on $p_1$ and from then on are the same as the original $p_2$ and $p_1$, so the new paths have at least one fewer intersections than the originals. But this contradicts the minimality property of $p_1$ and $p_2$, so they must not intersect at all. The case where $p_3$ does not intersect $p_1$ or $p_2$ until after $n_1$ is similar.

Second, if a connected, parallel indecomposable graph is modified so that each distinguished node is separated into distinct copies for each incoming edge, the resulting graph (without distinguished nodes) is connected. This is clear since the original graph is connected and the connected components of the resulting graph can be rejoined at their copies of the distinguished node and then combined in parallel to make the original graph. In particular, there is a d-path through any edge connected to nodes 1 or 2.

Now we can prove the h-graph property of a serial/parallel indecomposable graph with more than one edge. Consider the modified version of this graph as in the previous paragraph. We know it is connected and we can pick two, nonintersecting paths that are d-paths in the original graph. One at a time, remove edges and nodes from this graph as follows. If there are any loops, remove an edge from one, but not an edge from either of the special paths. When there are no loops, remove a leaf edge and node from the remaining tree, but again not from the special paths. Because this procedure preserves the connectedness of the graph, it clear that when it stops the remainder corresponds to an h-graph subgraph of the original graph. This finishes the sketch of a proof.

Notice that the procedure in the last paragraph can be run in reverse: starting with an h-graph, repeatedly either add an edge connecting two nodes, but not the distinguished ones, or add a node and an edge connecting it to an existing node other than the distinguished ones. We have seen that any serial/parallel indecomposable graph with more than one edge can be built in this way, starting from some h-graph.

Based on computations described below, the number of non-isomorphic, serial/parallel indecomposable graphs with number of edges from five through ten respectively is: 1, 5, 37, 226, 1460, 9235. These relatively modest numbers make the computation of the desired resistances feasible.

What follows is *Mathematica* code that implements the above ideas. (*Mathematica* code is not easy to read. This note should not be considered program documentation.)

The basic data structure looks like this example (of the unique five edge h-graph):

```
      graph[4,{edge[1,3],edge[1,4],edge[2,3],edge[2,4],edge[3,4]}]
```

In *graph[nodes,edges]*, *nodes* is the number of nodes, with the actual nodes named by the numbers from 1 to *nodes* and *edges* is a list of edges, with each edge represented as *edge[node1,node2]*. It is allowed to have repeated copies of the same edge (corresponding to resistors wired in parallel). In the code below, for each edge the node numbers are in order, there is at least one edge ending on every node and nodes 1 and 2 are always the distinguished nodes.

Here is code that computes all non-isomorphic h-graphs with a specified number of edges (>= 5):

```
hGraphs[edges_]:=Flatten[Table[
      graph[edges-1,Join[line[1,3,i,2],line[1,i+1,j,2],line[h1,j+1,edges-1,h2]]],
      {i,3,Floor[(edges+1)/2]},{j,i+1,edges-1},{h1,3,Floor[(i+3)/2]},
       {h2,i+1,If[OddQ[i]&&h1==Floor[(i+3)/2],Floor[(j+i+1)/2],j]}
      ]];

line[first_,second_,nextLast_,last_]/;second>nextLast:={edge[first,last]};
line[first_,second_,nextLast_,last_]:=Join[
    {edge[first,second]},edge@@@Partition[Range[second,nextLast],2,1],{edge[last,next
Last]}
    ];
```

Here is code that takes a graph and adds one edge in all the ways discussed above; first across all pairs of nodes other than 1,2 and then from all nodes other than 1 and 2 to a new node:

```
oneMore[graph[nodes_,edges_]]:=Join[
      normalForms[nodes,Rest@Flatten[
             Table[Append[edges,edge[i,j]],{i,nodes-1},{j,i+1,nodes}],1]],
      normalForms[nodes+1,Table[Append[edges,edge[i,nodes+1]],{i,3,nodes}]]
      ];
```

The above code uses the following to generate graphs in a normal form in which edges are sorted in *Mathematica's* order and the first such graph in sort order is chosen from all isomorphically equivalent graphs, which are generated by taking all permutations of the nodes that do not interchange distinguished and non-distinguished nodes.

```
normalForms[nodes_,edgeLists_]:=With[{
      permRules=With[{list=Range[3,nodes]},
          DeleteCases[Thread[list->#],HoldPattern[x_->x_]]&/@Permutations[list]
          ]
      },
    graph[nodes,normalForm[#/.permRules]]&/@edgeLists
    ];
normalForm[fullList_]:=First@Map[Sort,Join[fullList,fullList/.{1->2,2->1}],{0,-2}];
```

When adding the last edge of interest to a graph, it turns out to be faster to not generate normal forms, since the time to compute them is greater than the time to compute resistances, even allowing for the many isomorphically equivalent graphs whose resistances will be computed. That this is not done in general is because the number of graphs grows much too large if equivalent ones are not eliminated in early steps. Also, there is no need to add edges leading to new nodes in the last step, since they cannot affect the resistance. Here is code used to add the last edge:

```
lastOne[graph[nodes_,edges_]]:=Rest@Flatten[Table[
       graph[nodes,Append[edges,edge[i,j]]],{i,nodes-1},{j,i+1,nodes}],1];
```

Using all the above, here is how the non-isomorphic, serial/parallel indecomposable graphs with more than one and at most 10 nodes and a non-compressed batch with 11 nodes are computed. Starting with 5, the smallest possible number of edges, new graphs are iteratively generated by adding one edge in all possible ways to graphs with one fewer edge and the result combined with any h-graphs with the new number of edges. This takes about 2 hours on a 433 Mhz computer. (Only about 3 minutes if the goal is 10 rather than 11 nodes.)

```
graphs[5]=hGraphs[5];
Do[
    graphs[i]=Union@Flatten[Join[hGraphs[i],oneMore/@graphs[i-1]]],
    {i,6,10}
  ];
graphs[11]=Union@Flatten[Join[hGraphs[11],lastOne/@graphs[10]]];
```

Given a graph, it is now necessary to compute its effective resistance. This is done using Kirchoff's laws to generate a set of linear equations for *Mathematica* to solve. First, all adjacent edges, ones sharing a common node, are found, then all adjacent nodes, ones connected by an edge. If there are any nodes with only one other adjacent (a resistor is dangling), the code stops because this graph will have the same resistance as the smaller one gotten by deleting the node. Next, all d-paths are found by starting at node 1, recursively going from one node to all those adjacent to it and stopping a path at node 2 or when a previously encountered node is seen again.

Now, for each d-path, an equation is generated which sets the sum of variables associated with each edge on the path equal to one. The sign of the variable is set to minus if the edge on the path is traversed from a higher to lower node number. This equation corresponds to thinking of a 1 volt potential applied from node 1 to 2 and sums the voltage drops (using that the resistances are all 1 ohm) along the path. A second set of equations is generated, one for each node other than 1 and 2, which sets the sum of the signed variables associated with the edges adjacent to the node equal to zero. This corresponds to the sum of currents into a node being zero.

In general, these equations are not adequate to deduce all the currents in the graph, but they are enough to get those for all edges adjacent to node 1 (in particular because there is a d-path though each such edge for indecomposable graphs), which is what *Mathematica's Solve* function is told to do. The sum of these currents is the total current flowing out of node 1 and, given the 1 volt assumption, one over this sum is the effective resistance. Here is the code:

```
resistance[graph[nodes_,edges_]]:=Module[{adjacentEdges,firstEdges,paths},

    adjacentEdges=Split[Sort[Join[edges,Reverse/@edges]],First[#1]==First[#2]&];
    adjacentNodes=Union/@Map[Last,adjacentEdges,{2}];
    If[Min@@(Length/@adjacentNodes)==1,Return[1]];

    paths=Reap[nextNode[{1}]][[-1,-1]];
    firstEdges=First[adjacentEdges];
    1/Plus@@(x@@@firstEdges/.First@Solve[
                Join[
                  Thread[Plus@@@Apply[x,Partition[#,2,1]&/@paths,{2}]==1 ],
                  Thread[Plus@@@Apply[x,Drop[adjacentEdges,2],{2}]==0]
                  ],
                x@@@firstEdges,
                x@@@Complement[edges,firstEdges]
                ])
    ];
nextNode[path_]:=Scan[
    If[#==2,Sow[Append[path,2]],If[FreeQ[path,#],nextNode[Append[path,#]]]]&,
    adjacentNodes[[Last[path]]]
    ];

x[i_,j_]/;i>j=-x[j,i];
```

Putting this all together, it is possible to compute the number of unique resistances for graphs with successively increasing numbers of edges. For one edge, the 1-graph has resistance 1 and the other graph with one edge, with three nodes and the edge leading to the non-distinguished node, has infinite resistance. Recursively, the code includes any values from serial/parallel indecomposable graphs with the next number of edges and all serial and parallel combinations of earlier values whose edge counts total to the new number. Here is the code:

```
values[1]={1,Infinity};graphs[_]={};
values[n_]:=values[n]=Union@Flatten[{
            resistance/@graphs[n],
            Table[Outer[serialParallel,values[n-i],values[i]],{i,1,Floor[n/2]}]
            }];
serialParallel[Infinity,Infinity]=Infinity;
serialParallel[v1_,v2_]:={v1+v2,1/(1/v1+1/v2)};
```

To compute the number of resistances for graphs with from one to eleven edges, the above is called with:

```
Table[Length@values[i],{i,11}]
```

and produces (the answer!):

```
    2, 4, 8, 16, 36, 80, 194, 506, 1400, 4039, 12044
```

In particular, the answer for the original 10 resistor problem is 4039.

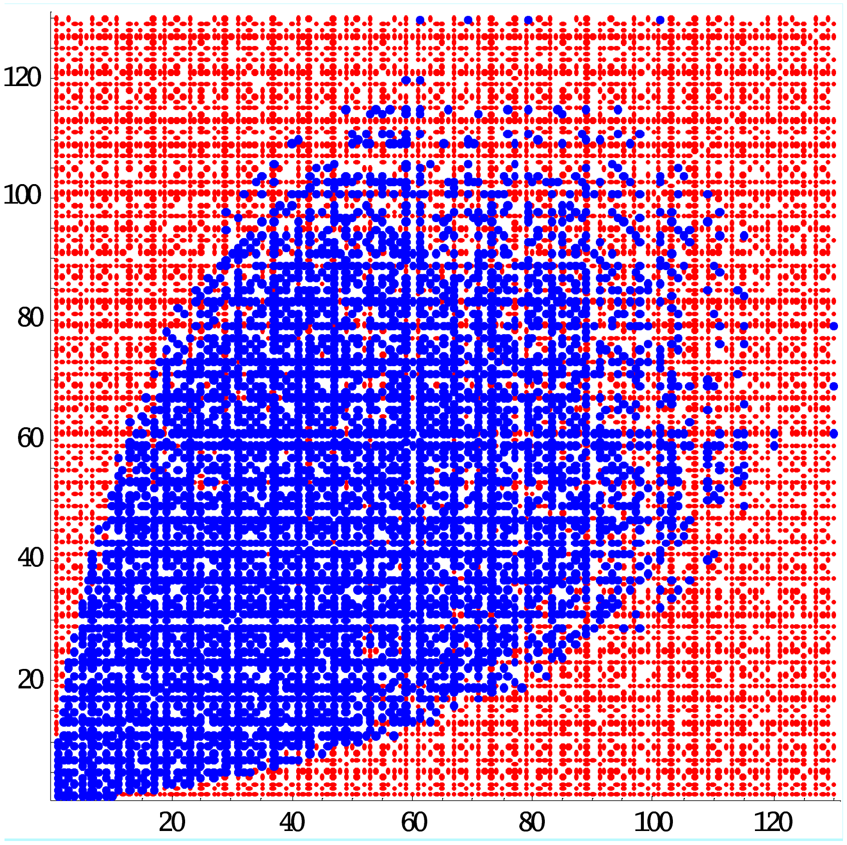
Given all the computation, it seems worthwhile to say a little more about the results.

First, an empirical formula for estimating the number of resistances for *n* resistors is $2^{n + .04(n-3)^2}$. The ratios of the estimated to true value for *n* = 1,...,11 are:

```
1.117, 1.028, 1., 1.028, 0.9931, 1.027, 1.028, 1.012, 0.9923, 0.9864, 1.003
```

but there is no reason the believe the formula extrapolates well.

Next, considering the 4039 resistance values for 10 resistors, the following plot has a blue dot for each value, with x,y coordinates from the denominator and numerator respectively. There is a red dot corresponding to any lowest terms fraction not in the computed data. A point is left white if the corresponding fraction is not in lowest terms. The axis upper limits (both 130) are the largest values for the computed resistances.



In addition to the blue points being so densely clustered, another notable thing about the plot is how symmetrical it is about the x = y line, This implies that if some circuit has resistance a/b then some other circuit likely has b/a. In fact, for 9 or fewer resistors, this symmetry is perfect. However, for 10 resistors the following values are achieved, but not their inverses:

$$\frac{95}{106}, \frac{101}{109}, \frac{98}{103}, \frac{97}{98}, \frac{103}{101}, \frac{97}{86}, \frac{110}{91}, \frac{103}{83}, \frac{130}{101}, \frac{103}{80}, \frac{115}{89}, \frac{106}{77}, \frac{109}{77}, \frac{98}{67}, \frac{101}{67}$$