

# Channel Routing

## Quick Notes

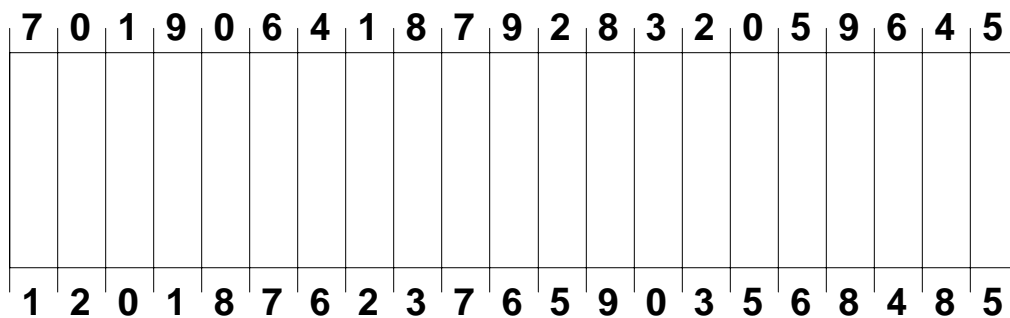
A channel is a horizontal routing area with fixed pins on the top and bottom. There are no pins to the right or left, but certain nets may be designated as route-to-the-edge. Thus we say that there may be “floating pins” to the right or left.

Each pin is designated by a number. All pins with the same number must be routed together. Pins with different numbers must be electrically isolated from one another.

The input to a channel routing problem is two sets of numbers, one that gives the pin numbers at the top of the channel, and the other that gives the pin numbers at the bottom of the channel.

A pin-number of zero designates an empty pin that is never connected to anything.

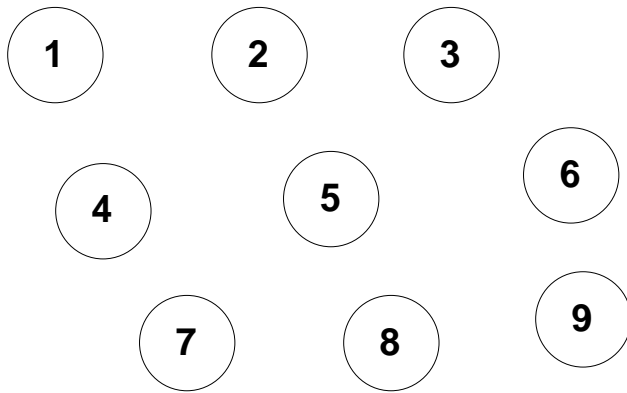
Example of a Channel: (The only important thing is the numbers.)



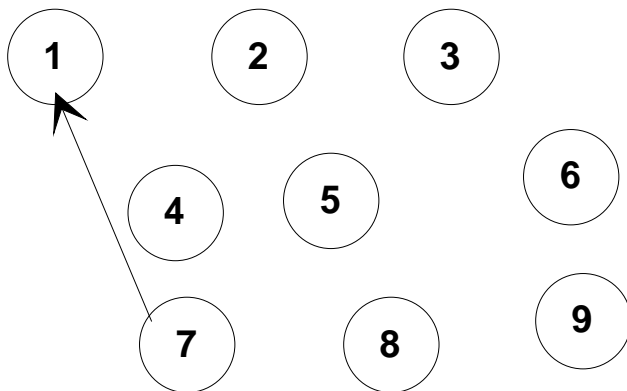
Channels are routed in two layers, one layer for the horizontal wires, one layer for the vertical. Except where contact-cuts are created, the two layers are insulated from one another.

The collection of all pins with the same number is called a NET. In the simplest algorithms, each net is given a single horizontal segment which runs from the first occurrence of a pin to the last occurrence. Vertical segments are added to connect the horizontal segment to the pin. If Net A appears at the top of column x and Net B appears at the bottom of column x, then the horizontal segment for Net A must be routed above the horizontal segment for Net B. If this is not done, the vertical segments of Net A and Net B will short together.

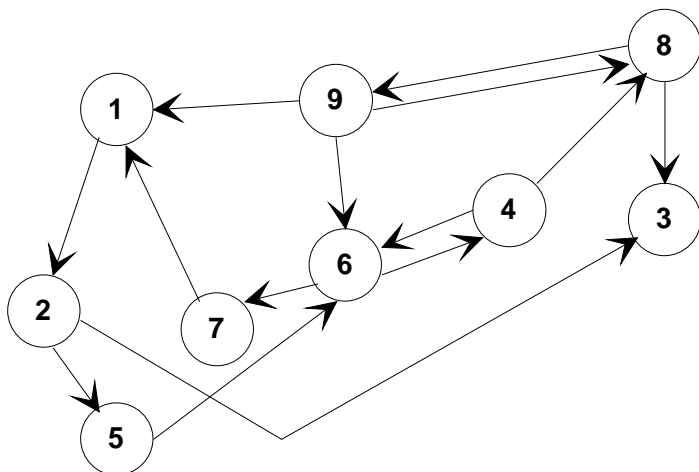
To handle the vertical positioning of nets in a channel, it is necessary to construct an object called a Vertical Constraint Graph (VCG). The VCG has one vertex for each net. The following is the VCG for the above channel before adding any edges.



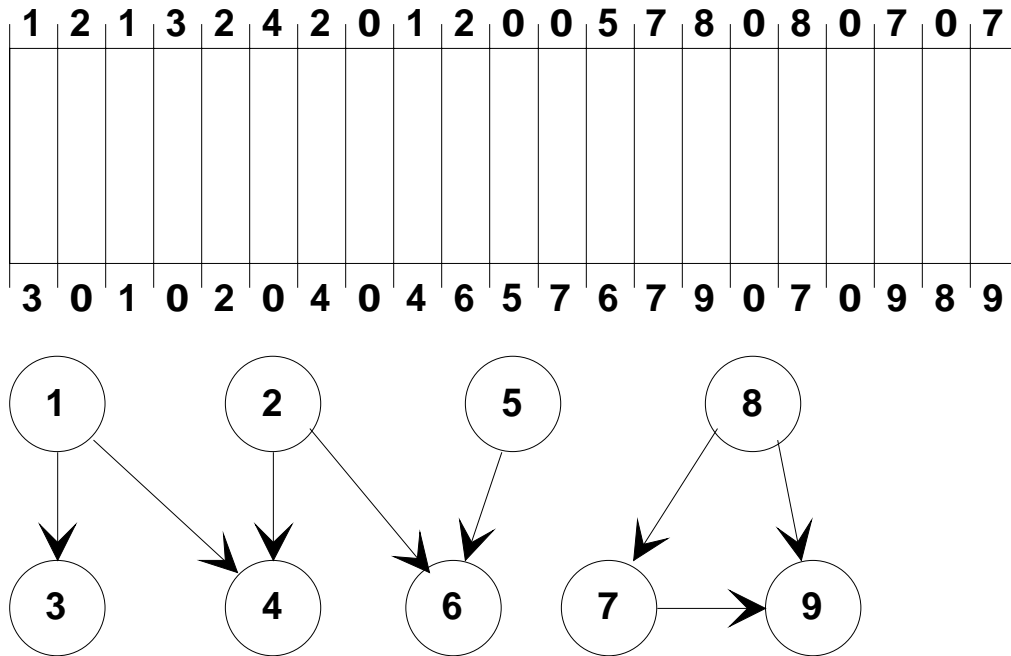
Next, we go through the channel, one column at a time, adding an edge for each column. The edge must be directed, and points from the top net to the bottom net. The following graph shows the edge that is added for the first column of the channel given above.



Completing the edges, we get the following graph:



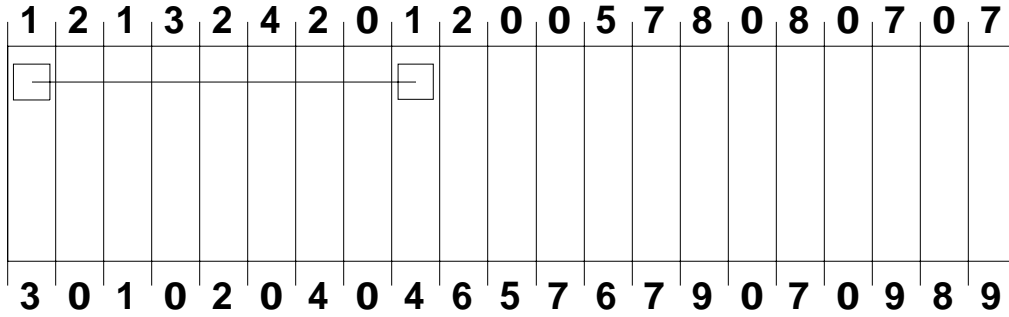
Note that this graph is highly cyclic. In most cases we will concentrate on algorithms that require acyclic VCGs. The following channel produces an acyclic VCG as illustrated.



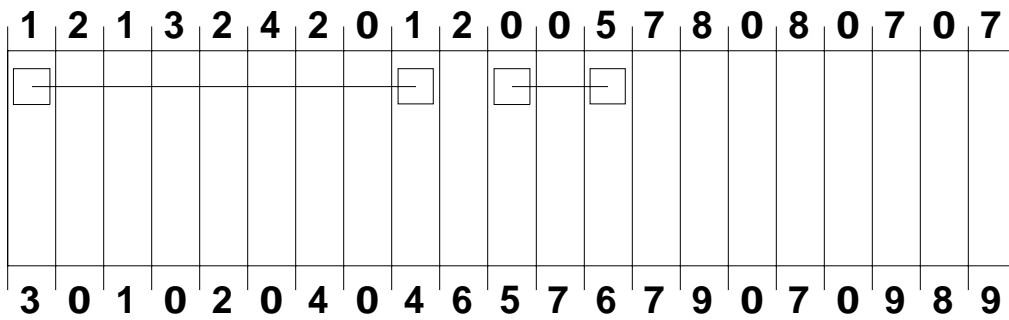
The left-edge algorithm is based on the VCG. One starts with the VCG, and makes a list of the vertices that have no incoming edges. In this case, it would be vertices 1, 2, 5, and 8. These are nets that are routed in the first row of the channel. We take this set of nets and sort it into ascending order by the leftmost column of the net. The following table gives these numbers.

- 1- Col 0
- 2- Col 1
- 5- Col 11
- 8- Col 15

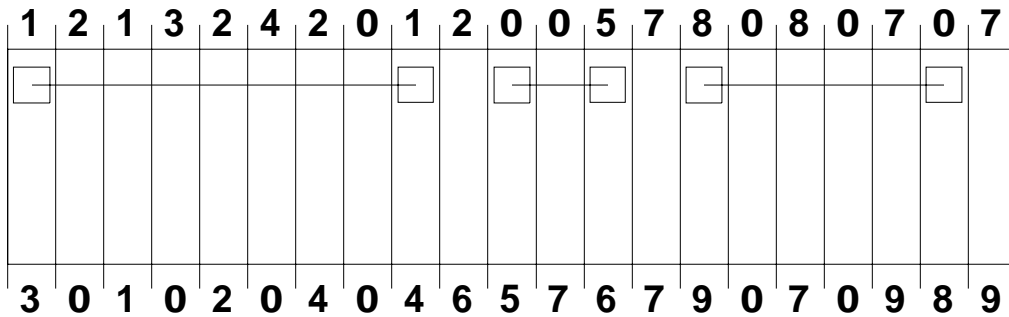
We attempt to put these four nets into the first row in the order given. When we come to a particular net, we check to see if it overlaps horizontally with anything already in the channel, and if it does, we skip that one. The following shows the channel with net 1 added. The contact-cuts at the ends of the net have been added, but not the internal contact cuts.



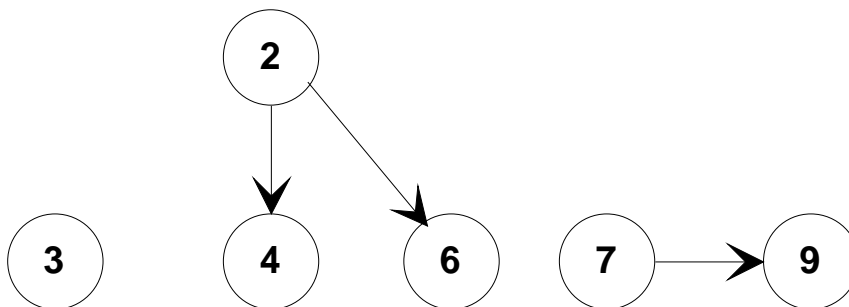
Now, because 2 overlaps with 1, we can't add net 2 to the first row, so we skip it and go on to net 5. Adding net 5 gives the following.



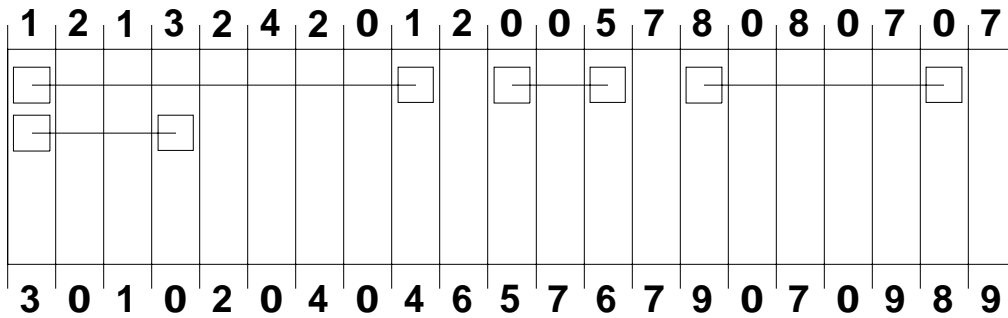
And finally, we add net 8.



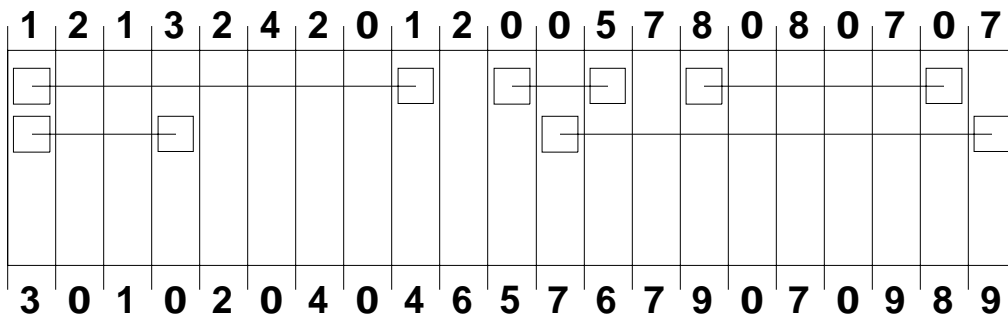
Now we increment the row-number to row 2, and delete nets 1, 5 and 8 from the VCG as follows.



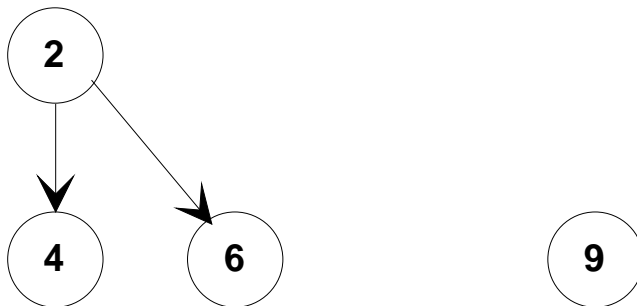
We can route 2, 3, and 7 in row 2. The leftmost order is 3, 2, 7, so we start with net 3, giving the following.



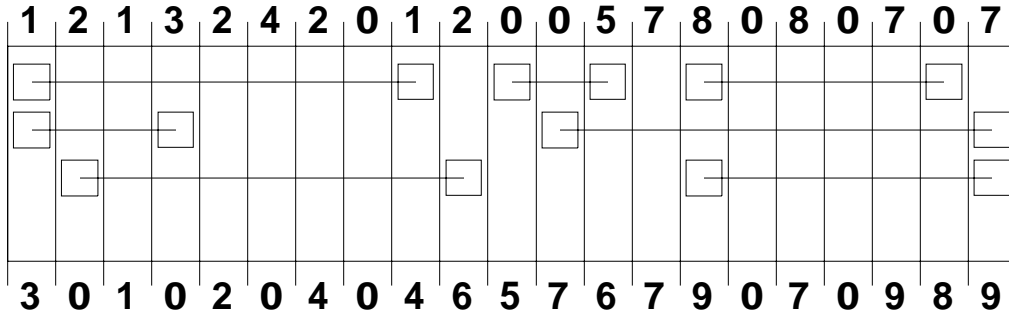
Since 2 overlaps with 3, we skip it, and add net 7 giving the following.



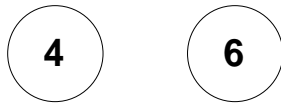
We advance to row 3 and delete 3 and 7 from the VCG, giving the following.



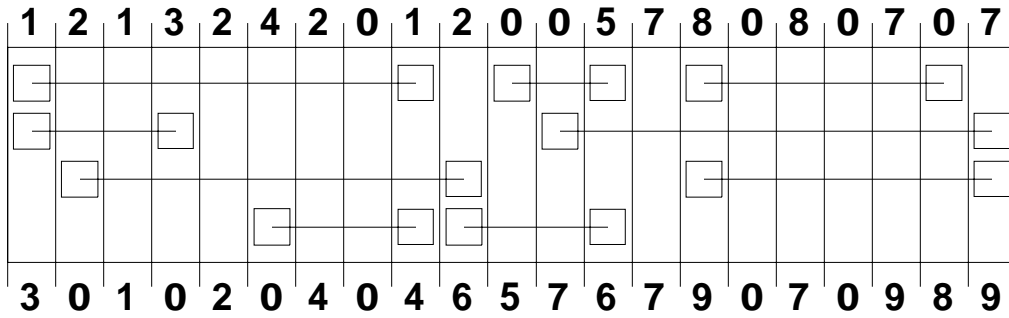
We can route net 2 and net 9 in row 3. They don't overlap so we can put both of them in giving the following.



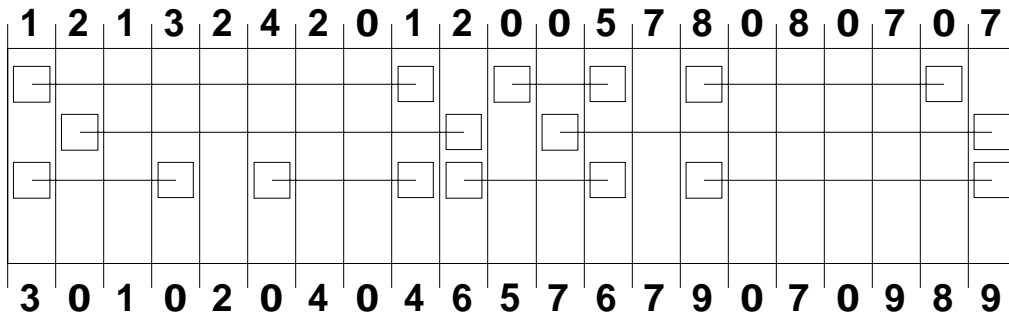
We advance to row 4 and delete nets 2 and 9 from the VCG as follows.



We can route nets 4 and 6 in row 4 as follows.



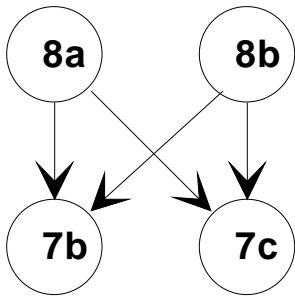
Of course, this routing is not optimal, as the following diagram shows.



## Dog-Leg Routing

Dog-leg routing is essentially the same thing as left-edge routing, but we are allowed to use more than one horizontal segment per net. Thus in the preceding example, we would break net 1 into net 1a, and net 1b. The break would occur at column 2 (the third column from the left). Net 2 has two internal pins, so it is broken into three pieces 2a, 2b,

and 2c. There are two “tricky” things to remember. First, a net is permitted to overlap with itself. Even though net 1a overlaps with net 1b in column 2, net 1a can be routed in the same row as net 1b. Second, both segments of a net exist in the column where the net is broken. Observe the fifth column from the left in the previous example. Net 8 is above net 7. This position is an internal pin for both nets 7 and 8. It is the dividing point between net 8a and net 8b, and it is also the dividing point between net 7b and net 7c. The VCG edges created for this position would look as follows.



Other than these two things, once the nets have been broken into segments, dogleg routing is essentially the same thing as left-edge routing.