Finding Hamiltonian Cycles in the Inner Cube of an n-Cube

Preliminary Draft

Brent M. Dingle

CPSC 626
Fall 1998
Texas A&M University

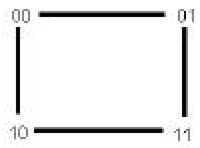
Abstract:
In this paper we will be examining sequential and parallel methods of finding Hamiltonian Cycles in the inner cube of a given n-cube. The intent was to convert a currently existing sequential method into a parallel method for time improvement or to create an entirely new parallel method – most likely specific to this problem. In the end it is concluded there is no significant gain from converting current sequential methods into parallel methods and no obviously implementable parallel solution presents itself specific to this problem.

Regardless, we will actually present solutions derived by the sequential methods discussed for the cases of n = 3, 5 and 7 (and attach the C++ sequential code implementation).

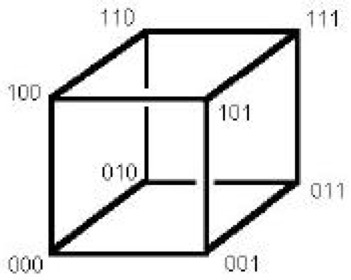
The first order of business is defining what is meant by inner cube of an n-cube.

Definition: An n-cube is constructed by taking all permutations of n binary digits and connecting them by placing edges between those permutations which differ by exactly one digit.

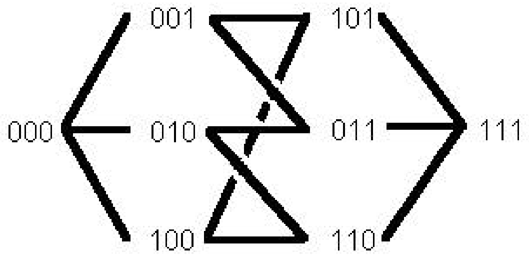
For example the 2-cube:



Or the 3-cube:



Thus for every $n \geq 2$, we create an 'n-dimensional' cube.
It should be noted that we can express these cubes in a slightly different fashion (henceforth referred to as columnar notation). For example consider the 3-cube again:



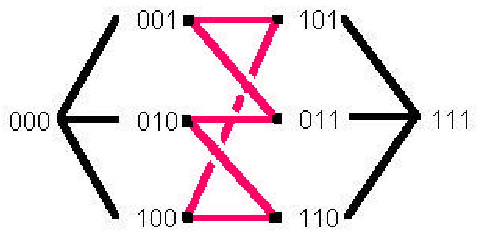
From this we can illustrate what we mean by inner cube.

Definition: The complement of a binary digit is obtained by changing each zero digit to a one and each one digit to a zero. (e.g. the complement of 010 is 101, the complement of 100 is 011, etc).

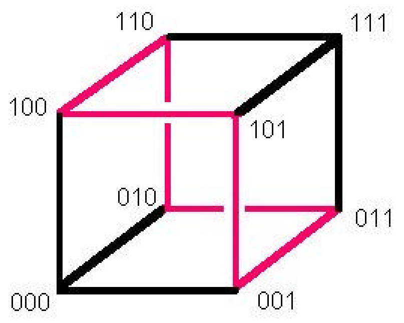
Definition: Given an n-cube with n odd, the inner cube is the graph composed of the set of vertices such that each vertex contains $(n+1)/2$ zeroes (ones) and $(n+1)/2 -1$ ones (zeroes) combined with the complements of such vertices with all edges between the vertices. This will correspond to the two innermost columns as we have drawn the 3-cube above (i.e. vertices: 001, 010, 100, 101, 011 and 110).

All other Graph Theory terms are defined in the commonly used fashions (see reference 6).

So what we are looking for is a Hamiltonian cycle on the inner cube of the n-cube.
In the 3-cube case, using columnar notation, this is easily done:



or using the more common drawing:



The 5-cube case is slightly more difficult, the 7-cube case is much more difficult and the 9-cube case is (currently) unsolved. To understand why this is so we need to examine some properties of the inner cube:

Properties of inner cubes:
Given an n-cube with n odd:
1. In our columnar notation there will always be n+1 columns representing the entire n-cube and the first (n+1)/2 columns will mirror the other side of the graph in a complementary fashion.
2. Our inner cube is made up of the (n+1)/2 column and the (n+1)/2 + 1 column.
3. The (n+1)/2 column will be the complement of the (n+1)/2 + 1 column.
4. There will be (n+1)/2 edges between these columns.
5. For i > 1 and i ≤ (n+1)/2 the number of vertices in the i^{th} column is:

$$Num.Verts(i) = \frac{NumVerts(i-1)}{(i-1)}(n-(i-2))\text{, where } NumVerts(1) = 1.$$

So for example with n = 5:

Column Number	Number of Vertices
1	**1**
2	**(1 / 1) * (5 – 0) = 5**
3	**(5 / 2) * (5 – 1) = 10**
4	**10 by symmetry to column 3**
5	**5 by symmetry to column 2**
6	**1 by symmetry to column 1 (or by extended definition)**

So our inner cube will have 10 + 10 = 20 vertices and each vertex will have degree = (5+1)/2 = 3.

Continuing on with n = 7:

Column Number	Number of Vertices
1	**1**
2	**(1 / 1) * (7 – 0) = 7**
3	**(7 / 2) * (7 – 1) = 21**
4	**(21 / 3) * (7 – 2) = 35**
5	**35 by symmetry**
6	**21 by symmetry**
7	**7 by symmetry**
8	**1 by symmetry**

So our inner cube will have 35 + 35 = 70 vertices and each vertex will have degree = (7+1)/2 = 4.

And in the case of n = 9:

Column Number	Number of Vertices
1	**1**
2	**(1 / 1) * (9 – 0) = 9**
3	**(9 / 2) * (9 – 1) = 36**
4	**(36 / 3) * (9 – 2) = 84**
5	**(84 / 4) * (9 – 3) = 126**
6	**126 by symmetry**
7	**84 by symmetry**
8	**36 by symmetry**
9	**9 by symmetry**
10	**1 by symmetry**

So our inner cube will have 126 + 126 = 252 vertices and each vertex will have degree = (9+1)/2 = 5.

And with no great surprise the numbers just keep getting larger and the difficulty keeps growing.

Notice our total number of possible solutions is around (degree of each vertex **-** 1)$^{\text{(number of vertices)}}$.

The general difficulties of finding a Hamiltonian cycle in an arbitrary graph are well documented throughout the Computer Science world. In general, the problem of determining if a given graph is Hamiltonian is NP-Complete and actually finding a Hamiltonian cycle is NP-Hard. It is also well known that this problem relates in some very direct ways to the Travelling Salesman Problem.

All of that aside it is also known that a general way to find Hamiltonian cycles sequentially is to use an exhaustive method based on "backtracking."
This solution method goes basically as follows:

1. Create the adjacency matrix of the graph to be used.
2. Pick a vertex to start on.
3. Store all the vertices adjacent to this vertex in a stack.
4. Pick the vertex on top of the stack.
5. Store all the vertices adjacent to this vertex (not already visited) in a stack.
6. Repeat 4 and 5 until we obtain a cycle or no longer have valid adjacent vertices.
7. Remove the top element on the stack and repeat 4, 5 and 6 until no elements remain.

For graphs with many vertices (say more than 30 or 40) this process takes a very long time to complete. However as it tries every possible path it will succeed in finding all existing Hamiltonian cycles.

It should be noticed that the "backtracking" method gains very little if "directly" converted into a parallel algorithm. You might be able to knock out several powers of order but when you are looking at say 4^{256} and comparing it to 4^{250}, while you have something better it does not help much.

Fortunately throughout time many people have attempted to solve this type of problem. Unfortunately, none of them have found a general solution. Most of the techniques of making the problem solvable in short order attempt to reduce the problem size in some way. This requires rather intensive knowledge of the problem and makes the solution method very specific. Some other techniques of solving this type of problem employ various heuristic methods. Yet others use a form of random "guessing" where the Hamiltonian cycle might be. Some of these methods do not even guarantee that they will find the cycle if it is indeed there. There are some methods which try to classify certain types of this problem with criteria about the graph. For instance there has been some focus on random graphs and on bipartite (planar) graphs. While this may lead to some form of "general type" solutions the criteria of the graphs is still rather limiting.

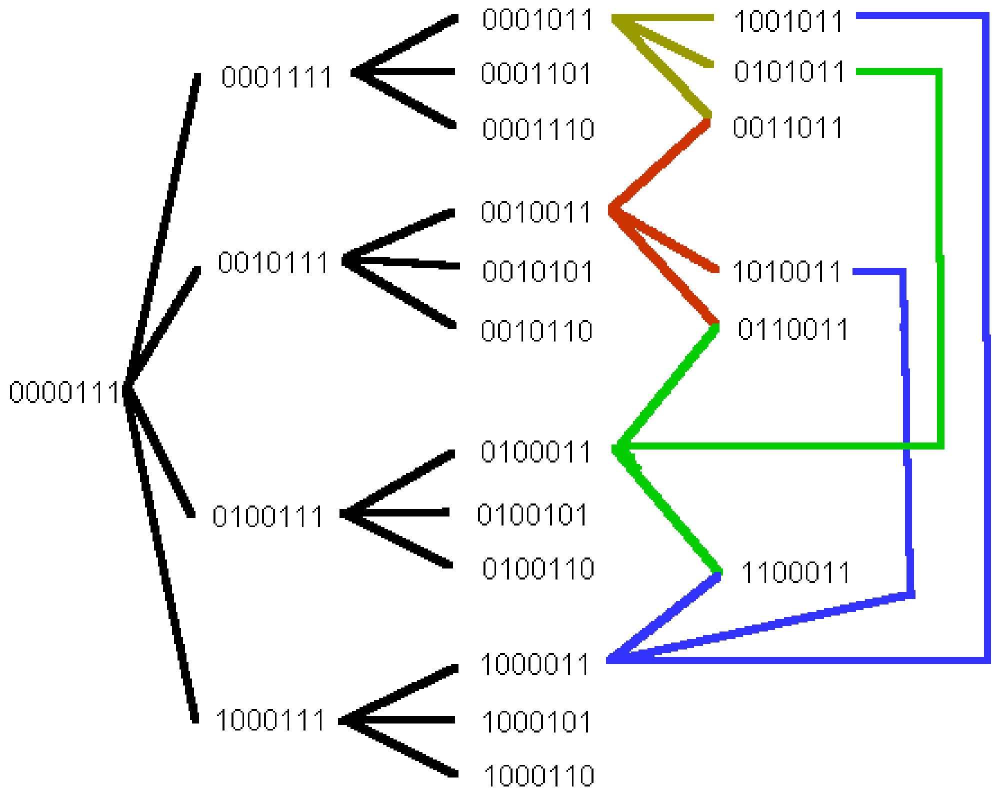
Generally this is discouraging. In my research I have not yet found any "general type" solutions which would apply to inner cubes – though the research on bipartite graphs using parallel processing may or may not be adapted in some way to be applicable (I am still in the process of 'digesting' the material at hand).

With this thought in mind I began focussing my research more towards parallel algorithms (some of what I had uncovered was already parallel related). Unfortunately, I found much the same results there as with the sequential cases. The solutions to this type of problem are mostly case specific, with very few general techniques. There is, however, a large number of insinuations in the literature that parallel machines should make these problems easier, but I noticed a severe lack of information beyond theoretical guessing.

Yet all is not lost, as it happens the backtracking method with the speed of computers today is capable of finding the solutions for n=3 and n=5 in a very short amount of time (in the manner of seconds and then in a manner of minutes correspondingly). The n=7 case takes slightly longer, running into the days and weeks (if not years) to come to full completion. Currently I have been running the program for two days on a Pentium 266 processor operating under Windows 95 and have located 1920 cycles. I have not yet analyzed the results thoroughly to say that they are distinct. Given that the first 24 vertices all match, I suspect the cycles are more or less the same just running through various combinations of smaller cycles. And that brings us to how we might optimize our solution process.

Since my research has turned up very little direct help on solving the problem, it has become obvious that the only way to solve it will be to analyze the problem structure itself and formulate some form of specific solution technique. The complication being that we cannot be too specific as it would be very nice if the solution were to work for any n-cube (i.e. not just for a 7-cube or 9-cube, but any n-cube). To this end we will examine the inner cube's structure:

Suppose we wish to parallelize our attack on the problem by preselecting 12 unique paths and giving each path to a single processor, to have it start from there:
So let n = 7 and consider the vertex 0000111 and its neighbors:



It was my original belief that because there was so much duplication of neighbors that this would somehow allow us to decrease the total number of paths being searched.
For example, assume we picked the paths:

A = 0000111 – 001111 – 0001011

and

B = 0000111 – 1000111 – 1000011

as two of our starting paths. Further assume we send path A to processor 1 and path B to processor 2. Then it would seem that if processor 1 searched the path A – 1001011 – whatever, then processor 2 should <u>not</u> have to search path B – 1001011 – whatever. Unfortunately, while I believe this to be true, I cannot prove it to be. However, if it is true it should greatly decrease the time needed to arrive at completion by reducing duplicated efforts. In the above picture it would reduce the path choices from 12 to 6 which would cascade to a much greater savings.

As for the existence of duplicated efforts, that should be obvious. The first evidence of this is simply noting that once you have found one cycle you can take the last element and make it the second and you will reverse the cycle (and an exhaustive method will have to calculate both these cases independently).

Also consider the below cycles which are solutions to the 7-cube scenario (presented by vertex number and then in binary):

<u>Cycle 1:</u>

1	*51*	*23*	*63*	*33*	*69*	*35*	*70*	*34*	*66*	*26*
60	*30*	*59*	*28*	*68*	*32*	*65*	*25*	*54*	*4*	*44*
16	*50*	*20*	*48*	*17*	*67*	*31*	*64*	*24*	*58*	*8*
39	*10*	*40*	*9*	*38*	**5**	**55**	**27**	**56**	**29**	**57**
7	**37**	**6**	**46**	**19**	**47**	**13**	**41**	**12**	**62**	**22**
53	**3**	**43**	**14**	**42**	**15**	**49**	**18**	**45**	**11**	**61**
21	**52**	**2**	**36**							

0000111	*1000111*	*1000110*	*1100110*	*1100100*
1110100	*1110000*	*1111000*	*1101000*	*1101100*
1001100	*1011100*	*1011000*	*1011010*	*1010010*
1110010	*1100010*	*1101010*	*1001010*	*1001110*
0001110	*0101110*	*0101100*	*0111100*	*0111000*
0111001	*0110001*	*1110001*	*1100001*	*1101001*
1001001	*1011001*	*0011001*	*0011101*	*0011100*
0011110	*0011010*	*0011011*	**0010011**	**1010011**
1010001	**1010101**	**1010100**	**1010110**	**0010110**
0010111	**0010101**	**0110101**	**0110100**	**0110110**
0100110	**0100111**	**0100101**	**1100101**	**1000101**
1001101	**0001101**	**0101101**	**0101001**	**0101011**
0101010	**0111010**	**0110010**	**0110011**	**0100011**
1100011	**1000011**	**1001011**	**0001011**	**0001111**

<u>Cycle 2:</u>

1	*51*	*23*	*63*	*33*	*69*	*35*	*70*	*34*	*66*	*26*
60	*30*	*59*	*28*	*68*	*32*	*65*	*25*	*54*	*4*	*44*
16	*50*	*20*	*48*	*17*	*67*	*31*	*64*	*24*	*58*	*8*
39	*10*	*40*	*9*	*38*	**2**	**52**	**21**	**61**	**11**	**41**
13	**47**	**19**	**46**	**6**	**37**	**7**	**57**	**29**	**56**	**27**
55	**5**	**45**	**18**	**49**	**15**	**42**	**14**	**43**	**12**	**62**
22	**53**	**3**	**36**							

0000111	*1000111*	*1000110*	*1100110*	*1100100*
1110100	*1110000*	*1111000*	*1101000*	*1101100*
1001100	*1011100*	*1011000*	*1011010*	*1010010*
1110010	*1100010*	*1101010*	*1001010*	*1001110*
0001110	*0101110*	*0101100*	*0111100*	*0111000*
0111001	*0110001*	*1110001*	*1100001*	*1101001*
1001001	*1011001*	*0011001*	*0011101*	*0011100*
0011110	*0011010*	*0011011*	**0001011**	**1001011**
1000011	**1100011**	**0100011**	**0100111**	**0100110**
0110110	**0110100**	**0110101**	**0010101**	**0010111**
0010110	**1010110**	**1010100**	**1010101**	**1010001**
1010011	**0010011**	**0110011**	**0110010**	**0111010**
0101010	**0101011**	**0101001**	**0101101**	**0100101**
1100101	**1000101**	**1001101**	**0001101**	**0001111**

Notice that the first 38 terms are identical, these terms are in italics. The variance occurs in the remaining 32 terms. Further notice the sequence in cycle 1: 5, 55, 27, 56, 29, 57, 7, 37, 6, 46, 19, 47, 13, 41. Now locate 5 in cycle 2. Now walk backwards in cycle 2 and you will obtain the same sequence. Look closer and you will see even more similarities. It almost appears as though some form of mobius twist was performed. This is only one instance of many that has turned up in looking at the solutions of the 7-cube case.

So clearly there is an enormous amount of duplication in the process as demonstrated by the results of the 7-cube case. Even as I type this paper, the program is still churning away trying to solve for all the cycles. It has currently found cycle number 1920 and still the first 24 vertices remain unchanged as compared to the first cycle it found *(As printing all those cycles would occupy over 200 pages I will not be doing so)*.

With this in mind I would propose a solution method such that:

1. Determine some finite number of short paths, say 12.Choose these paths in such a way as, based on the duplicate neighbor criteria above, that the total number of path possibilities is minimized.
2. Implement a "backtracking" method algorithm which takes a "short" path as its initial input and from there calculates any/all Hamiltonian cycles.
3. Having completed 1 and 2 above using a single processor run the modified "backtracking" algorithm 12 times on a single processor or use a parallel environment with 12 processors.

It should be noted that this method does require pre-processing to determine what neighbors are shared by which vertices. Once a method for determining how to do this algorithmically is completed it would seem that in a parallel environment the processors might "share" some of their solutions. For example knowing that you can convert the sequence 5, 55, 27, 56, 29, 57, 7, 37, 6, 46, 19, 47, 13, 41, 12, 62, 22, 53, 3, 43, 14, 42, 15, 49, 18, 45, 11, 61, 21, 52, 2, 36 into 2, 52, 21, 61, 11, 41, 13, 47, 19, 46, 6, 37, 7, 57, 29, 56, 27, 55, 5, 45, 18, 49, 15, 42, 14, 43, 12, 62, 22, 53, 3, 36 would be helpful. Along the same lines it might even be wise to calculate these equivalent sequences in advance.

In conclusion it should be obvious that this problem does – somewhere – have the properties necessary to reduce it to being solvable in a much shorter time frame. It should also be obvious that the time could be further reduced by sending specific "short" starter paths to different processors (or likewise using only one processor and multiple runs with different starting paths). However, it is with much regret, that in the past four weeks I have been unable to exploit these properties to the extent of actually implementing such a process.

- BMD

Attached should be the entire solution to the 5-cube case along with the C++ code used to derive the solution. The above cited 7-cube case solutions were also calculated by the attached code.

Bibliography

1. Bang-Jensen, J., M El Haddad, Y. Manoussakis and T. M. Przytycka, *Parallel Algorithms for the Hamiltonian Cycle and Hamiltonian Path Problems in Semicomplete Bipartite DiGraphs* from Algorithmica, 17: 67-87, Springer-Verlag New York Inc., 1997.
2. Bellman, Richard. Algorithms Graphs and Computers, Academic Press, New York, 1970. This book is Volume 62 in *Mathematics in Science and Engineering*.
3. Chambers, Lance. Practical Handbook of Genetic Algorithms: Applications, Volume I, CRC Press, New York, 1995.
4. Evans, James R. Optimization algorithms for networks and graphs – 2nd ed., rev. and expanded, Mercel Dekker, Inc., New York, 1992.
5. Hu, T. C. Combinatorial Algorithms, Addison Wesley Publishing Company, 1982.
6. Laufer, Henry B. Discrete mathematics and Applied Modern Algebra, Prindle, Weber and Schmidt (PWS), Boston, 1984.
7. Mackenzie, Philip D. and Quentin F. Stout, *Optimal parallel Construction of Hamiltonian Cycles and Spanning Tree in Random Graphs (Preliminary Version)*, from Proc. 5th ACM Symp. on Parallel Algorithms and Architectures, pp. 224-229, 1993.
8. Mott, Joe L. Discrete Mathematics for Computer Scientists and Mathematicians, Prentice-Hall, New Jersey, 1986.
9. Nijenhuis, Albert and Herbert S. Wilf. Combinatorial Algorithms, Academic Press, New York, 1975.
10. Reinelt, Gerard. Lecture Notes in Computer Science 840: The Traveling Salesman, Computation Solutions for TSP Applications, Springer-Verlag Berlin Heidelberg 1994.
11. Rulan, Kevin Scott. *Polyhedral Solution to the Pickup and Delivery Problem*, Dissertation presented to the Sever Institute of Washington University at Saint Louis for partial fulfillment for the degree of Doctor of Science, August, 1995.
12. Vandegriend, Basil. *Finding Hamiltonian Cycles: Algorithms, Graphs and Performance*, Thesis submitted to the University of Alberta, Spring 1998.

Program Output for the 5-cube:

```
HamTrack Program for finding Hamiltonian Cycles in
the inner cube of an n-cube.

Written by Brent Dingle, December 1998.
Program for CPSC 626, Texas A&M University.
Thanks to Dr. Amato (CPSC) and Dr. Hobbs (MATH).

Please enter the number of digits (odd integer < 16).
--> 5
For 5 digits we will be interested in
columns 3 and 4. We note there will be
10 vertices in each of these columns.

Thus if a Hamiltonian cycle exists it will
contain 20 vertices.

There will also be 3 edges from a vertex in
column 3 to a vertex in column 4.

Creating Adjacency Matrix...
Beginning to look for Ham. Cycles...

C1:
      1     15    8     19    10    20    9     17    3     11    2
      16    7     18    4     13    6     14    5     12

      00011 10011 10010 11010 11000 11100 10100 10110 00110 00111 00101
      10101 10001 11001 01001 01101 01100 01110 01010 01011

C2:
      1     15    8     19    10    18    7     16    2     13    4
      12    5     14    6     20    9     17    3     11

      00011 10011 10010 11010 11000 11001 10001 10101 00101 01101 01001
      01011 01010 01110 01100 11100 10100 10110 00110 00111

C3:
      1     15    8     19    5     14    6     20    10    18    7
      16    9     17    3     11    2     13    4     12

      00011 10011 10010 11010 01010 01110 01100 11100 11000 11001 10001
      10101 10100 10110 00110 00111 00101 01101 01001 01011

C4:
      1     15    8     19    5     12    4     13    6     14    3
      17    9     20    10    18    7     16    2     11

      00011 10011 10010 11010 01010 01011 01001 01101 01100 01110 00110
      10110 10100 11100 11000 11001 10001 10101 00101 00111
```

```
C5:
      1     15    8     17    9     20    10    19    5     12    4
      18    7     16    2     13    6     14    3     11

      00011 10011 10010 10110 10100 11100 11000 11010 01010 01011 01001
      11001 10001 10101 00101 01101 01100 01110 00110 00111

C6:
      1     15    8     17    9     16    7     18    4     13    2
      11    3     14    6     20    10    19    5     12

      00011 10011 10010 10110 10100 10101 10001 11001 01001 01101 00101
      00111 00110 01110 01100 11100 11000 11010 01010 01011

C7:
      1     15    8     17    3     14    6     20    9     16    7
      18    10    19    5     12    4     13    2     11

      00011 10011 10010 10110 00110 01110 01100 11100 10100 10101 10001
      11001 11000 11010 01010 01011 01001 01101 00101 00111

C8:
      1     15    8     17    3     11    2     13    6     14    5
      19    10    20    9     16    7     18    4     12

      00011 10011 10010 10110 00110 00111 00101 01101 01100 01110 01010
      11010 11000 11100 10100 10101 10001 11001 01001 01011

C9:
      1     15    7     18    10    20    9     16    2     11    3
      17    8     19    5     14    6     13    4     12

      00011 10011 10001 11001 11000 11100 10100 10101 00101 00111 00110
      10110 10010 11010 01010 01110 01100 01101 01001 01011

C10:
      1     15    7     18    10    19    8     17    3     14    5
      12    4     13    6     20    9     16    2     11

      00011 10011 10001 11001 11000 11010 10010 10110 00110 01110 01010
      01011 01001 01101 01100 11100 10100 10101 00101 00111

C11:
      1     15    7     18    4     13    6     20    10    19    8
      17    9     16    2     11    3     14    5     12

      00011 10011 10001 11001 01001 01101 01100 11100 11000 11010 10010
      10110 10100 10101 00101 00111 00110 01110 01010 01011

C12:
      1     15    7     18    4     12    5     14    6     13    2
      16    9     20    10    19    8     17    3     11

      00011 10011 10001 11001 01001 01011 01010 01110 01100 01101 00101
      10101 10100 11100 11000 11010 10010 10110 00110 00111
```

```
C13:
      1     15    7     16    9     20    10    18    4     12    5
      19    8     17    3     14    6     13    2     11

      00011 10011 10001 10101 10100 11100 11000 11001 01001 01011 01010
      11010 10010 10110 00110 01110 01100 01101 00101 00111


C14:
      1     15    7     16    9     17    8     19    5     14    3
      11    2     13    6     20    10    18    4     12

      00011 10011 10001 10101 10100 10110 10010 11010 01010 01110 00110
      00111 00101 01101 01100 11100 11000 11001 01001 01011

C15:
      1     15    7     16    2     13    6     20    9     17    8
      19    10    18    4     12    5     14    3     11

      00011 10011 10001 10101 00101 01101 01100 11100 10100 10110 10010
      11010 11000 11001 01001 01011 01010 01110 00110 00111

C16:
      1     15    7     16    2     11    3     14    6     13    4
      18    10    20    9     17    8     19    5     12

      00011 10011 10001 10101 00101 00111 00110 01110 01100 01101 01001
      11001 11000 11100 10100 10110 10010 11010 01010 01011

C17:
      1     12    5     19    10    20    6     14    3     11    2
      13    4     18    7     16    9     17    8     15

      00011 01011 01010 11010 11000 11100 01100 01110 00110 00111 00101
      01101 01001 11001 10001 10101 10100 10110 10010 10011

C18:
      1     12    5     19    10    18    4     13    2     16    7
      15    8     17    9     20    6     14    3     11

      00011 01011 01010 11010 11000 11001 01001 01101 00101 10101 10001
      10011 10010 10110 10100 11100 01100 01110 00110 00111

C19:
      1     12    5     19    8     17    9     20    10    18    4
      13    6     14    3     11    2     16    7     15

      00011 01011 01010 11010 10010 10110 10100 11100 11000 11001 01001
      01101 01100 01110 00110 00111 00101 10101 10001 10011

C20:
      1     12    5     19    8     15    7     16    9     17    3
      14    6     20    10    18    4     13    2     11

      00011 01011 01010 11010 10010 10011 10001 10101 10100 10110 00110
      01110 01100 11100 11000 11001 01001 01101 00101 00111
```

```
C21:
        1       12      5       14      6       20      10      19      8       15      7
        18      4       13      2       16      9       17      3       11

        00011 01011 01010 01110 01100 11100 11000 11010 10010 10011 10001
        11001 01001 01101 00101 10101 10100 10110 00110 00111

C22:
        1       12      5       14      6       13      4       18      7       16      2
        11      3       17      9       20      10      19      8       15

        00011 01011 01010 01110 01100 01101 01001 11001 10001 10101 00101
        00111 00110 10110 10100 11100 11000 11010 10010 10011

C23:
        1       12      5       14      3       17      9       20      6       13      4
        18      10      19      8       15      7       16      2       11

        00011 01011 01010 01110 00110 10110 10100 11100 01100 01101 01001
        11001 11000 11010 10010 10011 10001 10101 00101 00111

C24:
        1       12      5       14      3       11      2       16      9       17      8
        19      10      20      6       13      4       18      7       15

        00011 01011 01010 01110 00110 00111 00101 10101 10100 10110 10010
        11010 11000 11100 01100 01101 01001 11001 10001 10011

C25:
        1       12      4       18      10      20      6       13      2       11      3
        14      5       19      8       17      9       16      7       15

        00011 01011 01001 11001 11000 11100 01100 01101 00101 00111 00110
        01110 01010 11010 10010 10110 10100 10101 10001 10011

C26:
        1       12      4       18      10      19      5       14      3       17      8
        15      7       16      9       20      6       13      2       11

        00011 01011 01001 11001 11000 11010 01010 01110 00110 10110 10010
        10011 10001 10101 10100 11100 01100 01101 00101 00111

C27:
        1       12      4       18      7       16      9       20      10      19      5
        14      6       13      2       11      3       17      8       15

        00011 01011 01001 11001 10001 10101 10100 11100 11000 11010 01010
        01110 01100 01101 00101 00111 00110 10110 10010 10011

C28:
        1       12      4       18      7       15      8       17      9       16      2
        13      6       20      10      19      5       14      3       11

        00011 01011 01001 11001 10001 10011 10010 10110 10100 10101 00101
        01101 01100 11100 11000 11010 01010 01110 00110 00111
```

```
C29:
      1     12    4     13    6     20    10    18    7     15    8
      19    5     14    3     17    9     16    2     11

      00011 01011 01001 01101 01100 11100 11000 11001 10001 10011 10010
      11010 01010 01110 00110 10110 10100 10101 00101 00111

C30:
      1     12    4     13    6     14    5     19    8     17    3
      11    2     16    9     20    10    18    7     15

      00011 01011 01001 01101 01100 01110 01010 11010 10010 10110 00110
      00111 00101 10101 10100 11100 11000 11001 10001 10011

C31:
      1     12    4     13    2     16    9     20    6     14    5
      19    10    18    7     15    8     17    3     11

      00011 01011 01001 01101 00101 10101 10100 11100 01100 01110 01010
      11010 11000 11001 10001 10011 10010 10110 00110 00111

C32:
      1     12    4     13    2     11    3     17    9     16    7
      18    10    20    6     14    5     19    8     15

      00011 01011 01001 01101 00101 00111 00110 10110 10100 10101 10001
      11001 11000 11100 01100 01110 01010 11010 10010 10011

C33:
      1     11    3     17    9     20    6     14    5     12    4
      13    2     16    7     18    10    19    8     15

      00011 00111 00110 10110 10100 11100 01100 01110 01010 01011 01001
      01101 00101 10101 10001 11001 11000 11010 10010 10011

C34:
      1     11    3     17    9     16    2     13    4     18    7
      15    8     19    10    20    6     14    5     12

      00011 00111 00110 10110 10100 10101 00101 01101 01001 11001 10001
      10011 10010 11010 11000 11100 01100 01110 01010 01011

C35:
      1     11    3     17    8     19    10    20    9     16    2
      13    6     14    5     12    4     18    7     15

      00011 00111 00110 10110 10010 11010 11000 11100 10100 10101 00101
      01101 01100 01110 01010 01011 01001 11001 10001 10011

C36:
      1     11    3     17    8     15    7     18    10    19    5
      14    6     20    9     16    2     13    4     12

      00011 00111 00110 10110 10010 10011 10001 11001 11000 11010 01010
      01110 01100 11100 10100 10101 00101 01101 01001 01011
```

```
C37:
        1      11     3      14     6      20     9      17     8      15     7
        16     2      13     4      18     10     19     5      12

        00011 00111 00110 01110 01100 11100 10100 10110 10010 10011 10001
        10101 00101 01101 01001 11001 11000 11010 01010 01011

C38:
        1      11     3      14     6      13     2      16     7      18     4
        12     5      19     10     20     9      17     8      15

        00011 00111 00110 01110 01100 01101 00101 10101 10001 11001 01001
        01011 01010 11010 11000 11100 10100 10110 10010 10011

C39:
        1      11     3      14     5      19     10     20     6      13     2
        16     9      17     8      15     7      18     4      12

        00011 00111 00110 01110 01010 11010 11000 11100 01100 01101 00101
        10101 10100 10110 10010 10011 10001 11001 01001 01011

C40:
        1      11     3      14     5      12     4      18     10     19     8
        17     9      20     6      13     2      16     7      15

        00011 00111 00110 01110 01010 01011 01001 11001 11000 11010 10010
        10110 10100 11100 01100 01101 00101 10101 10001 10011

C41:
        1      11     2      16     9      20     6      13     4      12     5
        14     3      17     8      19     10     18     7      15

        00011 00111 00101 10101 10100 11100 01100 01101 01001 01011 01010
        01110 00110 10110 10010 11010 11000 11001 10001 10011

C42:
        1      11     2      16     9      17     3      14     5      19     8
        15     7      18     10     20     6      13     4      12

        00011 00111 00101 10101 10100 10110 00110 01110 01010 11010 10010
        10011 10001 11001 11000 11100 01100 01101 01001 01011

C43:
        1      11     2      16     7      18     10     20     9      17     3
        14     6      13     4      12     5      19     8      15

        00011 00111 00101 10101 10001 11001 11000 11100 10100 10110 00110
        01110 01100 01101 01001 01011 01010 11010 10010 10011

C44:
        1      11     2      16     7      15     8      19     10     18     4
        13     6      20     9      17     3      14     5      12

        00011 00111 00101 10101 10001 10011 10010 11010 11000 11001 01001
        01101 01100 11100 10100 10110 00110 01110 01010 01011
```

```
C45:
        1       11      2       13      6       20      9       16      7       15      8
        17      3       14      5       19      10      18      4       12

        00011 00111 00101 01101 01100 11100 10100 10101 10001 10011 10010
        10110 00110 01110 01010 11010 11000 11001 01001 01011



C46:
        1       11      2       13      6       14      3       17      8       19      5
        12      4       18      10      20      9       16      7       15

        00011 00111 00101 01101 01100 01110 00110 10110 10010 11010 01010
        01011 01001 11001 11000 11100 10100 10101 10001 10011

C47:
        1       11      2       13      4       18      10      20      6       14      3
        17      9       16      7       15      8       19      5       12

        00011 00111 00101 01101 01001 11001 11000 11100 01100 01110 00110
        10110 10100 10101 10001 10011 10010 11010 01010 01011

C48:
        1       11      2       13      4       12      5       19      10      18      7
        16      9       20      6       14      3       17      8       15

        00011 00111 00101 01101 01001 01011 01010 11010 11000 11001 10001
        10101 10100 11100 01100 01110 00110 10110 10010 10011


Total number of Hamiltonian cycles found: 48
Note: depending on how the graph was set up there might
be only half that number.
```