Here is the perfect code cryptosystem, played out by Alice and Bob:

(1) Bob creates a graph G (his public key) that contains a perfect code P (his private key). He keeps P to himself and shares G with everyone.

(2) Alice wants to send an integer N to Bob. To do this, she chooses one integer number n_i for each vertex v_i in Bob's graph G, the only restriction on the n_i's being $n_1 + n_2 + \dots + n_k = N$. (They sum to N)

(3) Having labelled all the vertices of Bob's graph G with numbers n_i, Alice hides her message as follows:

For each vertex v, replace the label of v as follows: If v is adjacent to $v_1, v_2, \dots, v_m$, the new label of v is $n_v + n_1 + n_2 + \dots + n_m$. (I.e: add up the labels of the vertices adjacent to v and v's label). (Clumping) She sends the newly labeled graph to Bob.

(4) Bob adds up the labels of the vertices in P. The result is N.

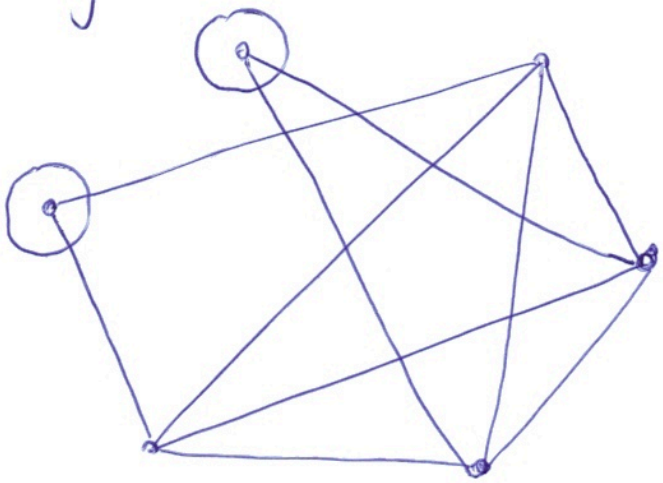
Why does Bob get N?

(upon adding)

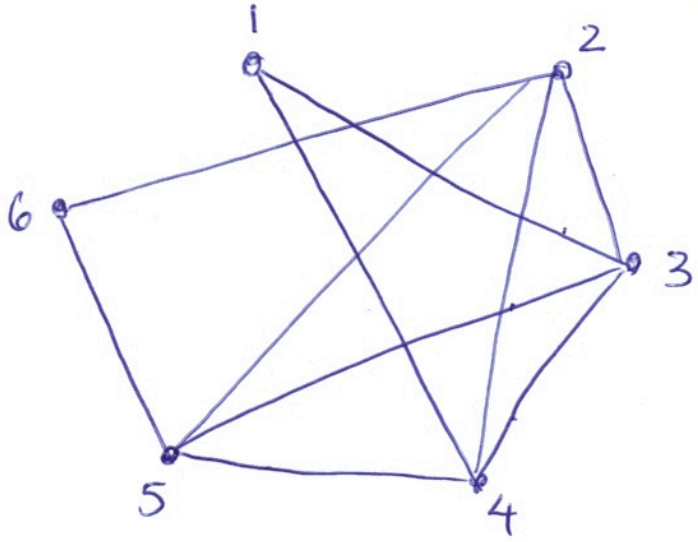
Every vertex in G is connected to exactly one vertex in P. So when you sum the clumped labels of P, you are exactly summing all of the original labels of all vertices in G!

Example: We will do a graph with only a few vertices, which of course is not secure. Bob would do better to make his graph more complicated.

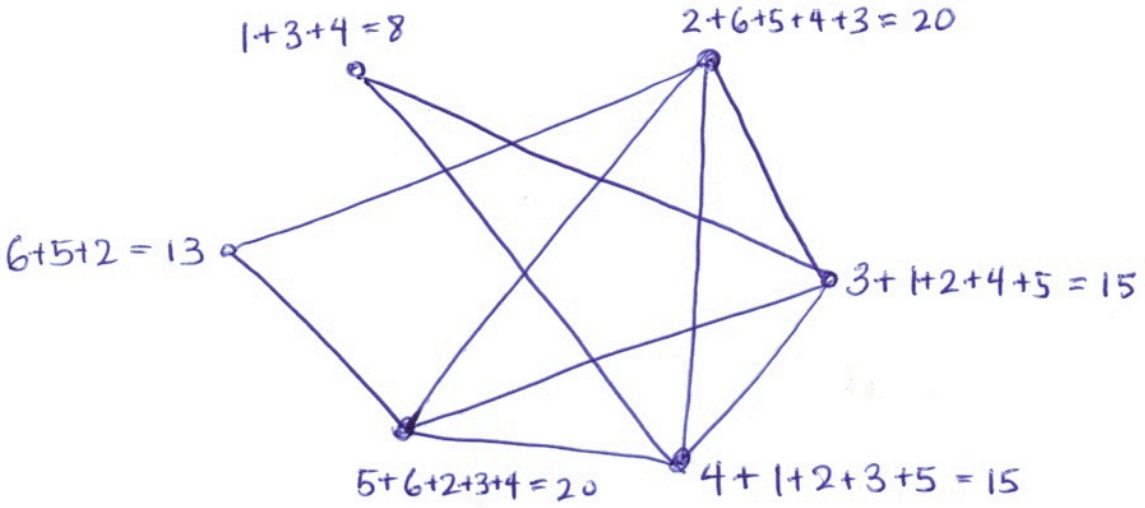
(1) Bob shares this graph, the circled vertices are his private key:



(2) Alice chooses a number: $N = 21$ She labels:



(3) Alice clumps to make new labels:



(4) Bob sums the clumped labels on the vertices in his perfect code:

$$8 + 13 = 21.$$

He got the message!

Remark: Note that the security of the system depends upon the assumption that finding a perfect code in a graph is hard. I.e., if we had an easy way of looking at Bob's graph and deducing the perfect code that he's keeping secret, then we could read all the messages being sent to Bob.

Q: How do we know it's hard to find a perfect code?

Ans: We don't know for certain.