## Discussion forum questions

1. We are feeling experimental and want to create a new dish. There are various ingredients we can choose from and we'd like to use as many of them as possible, but some ingredients don't go well with others. If there are n possible ingredients (numbered 1 to n), we write down an n × n binary matrix where the (i, j) entry is 1 if i and j can go together and 0 otherwise. Notice that this matrix is necessarily symmetric; and that the diagonal entries are always 0.
We wish to solve the following problem:

**EXPERIMENTAL CUISINE:**

*input:* n, the number of ingredients to choose from; B,the n × n binary matrix that encodes which items go well together
*output:* the maximum number of ingredients which can be selected together

Show that if EXPERIMENTAL CUISINE can be solved in polynomial time, then P=NP.
**Answer:** By reduction to INDEPENDENT SET: we can use an algorithm that solves EXPERIMENTAL CUISINE to solve INDEPENDENT SET.
For INDEPENDENT SET, we wish to know if a graph G has a set of k vertices such that no two vertices are adjacent. To solve this, we create an instance of the EXPERIMENTAL CUISINE problem. For each vertex of the graph, we create an ingredient. The matrix B is the complement of the adjacency matrix for G: B(i, j) is 0 if there is an edge between vertices i and j in G (in this case we would not be allowed to select these ingredients). If the algorithm for EXPERIMENTAL CUISINE finds an answer ≥ k, then the solution to INDEPENDENT SET is true and otherwise false.
So, if we can solve EXPERIMENTAL CUISINE in poly-time, we can solve INDEPENDENT SET in poly-time.
Since INDEPENDENT SET is NP-complete, solving INDEPENDENT SET in poly-time would imply that P=NP.

2. Question 8.10 from DPV
**Answer:**

(a) This is a generalization of the clique problem:
Suppose that we could solve the subgraph isomorphism problem. Then we could let G (the input subgraph) be a clique of any size and we could see if G is contained in H. This would solve the clique problem which we know to be NP-complete.

(b) This is a generalization of the Rudata path problem:
Suppose that we could solve the longest path problem. We could take our input graph to be the input to the Rudrata path problem G, and the integer will be given by $|V| - 1$. Ifwecan find a path in G of length $|V| - 1$ then this is a Rudrata path. We know that the Rudrata path problem is NP-complete.

(c) This is a generalization of vertex cover: Definition of set cover: Given a set E and subsets $S_1, S_2, \dots, S_m \subseteq E$ and a budget b select b of these subsets so that their union is E. Suppose that we could solve the set cover problem. We could then take our set E to be the edges of the graph and a subset $S_i$ for each vertex containing the edges adjacent to that vertex. We know that the vertex cover problem is NP-complete.