- Hamiltonian Path to TSP cycle
  - Similarity: visit all vertices
  - Differences 1. unweighted graph vs. weighted graph
    2. Path vs. cycle. Set all edges to have weight 1
    (Q: how to ensure the edge from t to s is selected?)

- (1) Given a Hamiltonian Path instance, add a vertex x, add edges (t, x), (x, s), set all edges to have weight 1, set L = n + 1, this will be the TSP cycle instance.
- (2) For any Hamiltonian path P in original graph, P + (t, x) + (x, s) form a TSP cycle of length n + 1.
- (3) For any cycle of length n + 1 that visits every vertex, it must visit every vertex exactly once.
  Cut the cycle when it visits vertex x (the only way to visit x is by (t, x), (x, s)), we get a path from s to t that visits every
vertex in original graph exactly once.

- **3-SAT** to quadratic programming

  - \( (x_1 \lor x_2 \lor \overline{x_3}) \land \ldots \land (x_2 \lor \overline{x_3} \lor \overline{x_4}) \land \ldots \)
  - clause: or of 3 “literals”
  - literal: \( x \) or \( \overline{x} \)
  - formula: and of many clauses
  - goal: is there an assignment to the variables such that all clauses are satisfied?

- quadratic programming

  - variables \( x_1, \ldots, x_n \in \mathbb{R} \)
  - constraints:
    \[
    \begin{align*}
    x_1^2 \leq 3 & \quad x_1^2 + 2x_1x_2 - x_2^2 \geq 1 \\
    x_2^2 - 2x_2 \geq x_3^2 - x_2x_3 \\
    \end{align*}
    \]
  - goal: is there an assignment to the variables such that all constraints are satisfied?

- reduction

  1. boolean var. vs. real var.
  2. 3-SAT \( \lor \) clause vs. quadratic constraint.

  Q: can we somehow constrain a real var to have only two values?

  \[
  x_1^2 - x_1 = 0 \quad x_1 = 0 \text{ or } 1
  \]

  \[
  (x_1 \lor x_2 \lor \overline{x_3}) \quad x_1 + x_2 + (1-x_3) \geq 1
  \]

  # of literals satisfied in clause

  Given a 3-SAT instance, for every variable \( x_i \) in 3-SAT:
  create a variable \( y_i \) in QP, add constraint \( y_i^2 - y_i = 0 \)
  for every clause in 3-SAT \( (x_a \lor x_b \lor \overline{x_c}) \):
  create a constraint \( y_a + y_b + (1-y_c) \geq 1 \)
TRIPARTITE MATCHING to SUBSET VECTORS

- Similarity: T.M. select hyperedges, S.V. select vectors
- Difference: hype. vs. vectors
  - exactly n hyperedges vs. select any number of vectors.

- Idea: encode hyperedges as vectors.

3n vertices, a hyperedge: 2 of the 3n vertices

\[(U_1, V_2, W_3)\]

Create vectors of 3n dimensions, each dimension corresponds to

a vertex

\[u, u_2 \ldots u_n, v_1, v_2 \ldots v_n, w_1, w_2 \ldots w_n\]

\[(u_1, v_2, w_3) \rightarrow \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 1 \cdots & 0 \end{pmatrix}\]

\[(u_2, v_1, w_n) \rightarrow \begin{pmatrix} 0 & 1 & \cdots & 0 & 1 & 0 \cdots & 0 & 0 \end{pmatrix}\]

in T.M. select n hyper-edges s.t. every vertex is adjacent to exactly 1 hyper-edge.

Sum of these hyper-edges should be equal to

\[U = (1 1 1 \ldots 1)\]

SUBSET VECTORS to SUBSET SUM

Key observation: a number can be viewed as a vector if we take its base B representation.

\[
\begin{array}{cccccccc}
0 & B & 1 & 0 & B & 1 & 0 & B & 1 \\
0 & B & 1 & 0 & B & 1 & 0 & B & 1 \\
\end{array}
\]

want: sum of numbers behave like sum of vectors.

true when there is no carry operation

\[1001 + 110 = 1111\]
Choose \( B \) to be very large, so when we take sum, there will be no carry operation.

- SUBSET SUM, DP runs in time \( O(nm) \)

input length = \( \lceil n \log_2 m \rceil \)

\( nm \) is not a polynomial of \( n \lceil \log_2 m \rceil \)