Junction tree algorithm.

We start from Variable Elimination.

Example:

\[ P(x) = \frac{1}{Z} \prod_{i=1}^{n} \psi_{A_i} \psi_{A_iB_i} \psi_{B_iC_i} \psi_{C_i} \psi_{C_iD_i} \psi_{D_iE_i} \psi_{E_iF_i} \]

induced graph.

If we want marginal distribution \( P(A) \):

\[ P(A) \propto \sum_{B} \psi_{AB} \sum_{C} \psi_{AC} \]

\[ \sum_{D} \psi_{BD} \sum_{E} \psi_{CE} \]

\[ \sum_{F} \psi_{CF} \psi_{C} \]

\[ \psi_{BC} \]

\[ P(F) : \quad E \rightarrow D \rightarrow B \rightarrow A \rightarrow C. \]

\[ \sum_{C} \psi_{CF} \sum_{A} \psi_{AC} \sum_{B} \psi_{AB} \sum_{D} \psi_{BD} \sum_{E} \psi_{CE} \psi_{DE} \]

\[ \sum_{F} \psi_{CF} \psi_{C} \]

\[ \psi_{BC} \]

Junction tree: data structure that can store partial computations, which can be reused to make computation of multiple queries efficient. → no need to redo elimination each time.

Once we get a clique tree, simply run sum-product on it.

\[ [s = cnc', s'' = cnc''] \quad m_{c' \rightarrow c} (s) = \sum_{c''} \psi_{c} \prod_{c' \in N(c) \setminus c'} m_{c'' \rightarrow c} (s'') \]

marginals: \[ p(c) \propto \psi_{c} \prod_{c' \in N(c)} m_{c' \rightarrow c} (s) \]
There could be multiple clique trees, not all of them junction trees.

![Clique Trees Diagram]

Separator sets.

☆ Junction tree property / Running intersection property.

For any clique nodes \( c, c' \) in a clique tree, \( S = c \cap c' \) must be contained in every node on the path connecting \( c \) and \( c' \).

**Def** Junction tree: clique trees that have this property.

**why useful?**

![Separator Sets Diagram]

\( R = c \setminus S \) only appears in \( c \).

So guarantees safe sum over \( R \).

**How to get a junction tree?**  
- Maximum weight spanning tree.
Specifying an initial elimination order is also unnecessary.

More generally, the operation of adding all edges to get the induced graph is called "triangulation".

→ adding edges so that any cycle of length ≥ 4 will have at least a chord.
→ after this, the only possible chordless cycles are triangles
→ elimination guarantees a triangulated graph.

Examples:

![Graphs](image)

Summary:

1. triangulation, find maximal cliques.
2. build junction tree.
3. initialize clique potentials.
4. run sum-product on junction tree.

Complexity:

1. 2. 3. only need to do once for a given graph, can be reused for any distribution that factorizes according to this graph.
4. complexity at most $O(NE^{w+1})$.

N: # cliques, 
K: # states for each variable, 
w: tree-width.
Example: HMM:

\[
P(X, Y) = P(x_1) \prod_{i=2}^{n} P(x_i | x_{i-1}) \prod_{i=1}^{n} P(y_i | x_i)
\]

\[
\psi_{i-1}(x_{i-1}, x_i) \bigg/ \psi_i(x_i, y_i)
\]

\[
\psi_{i-1}(x_{i-1}, x_i) = P(x_i | x_{i-1}) \quad i \geq 2
\]

Assume condition on \( Y \), we want marginals of \( P(X | Y) \) → simply clamp \( Y \) to the specified value and everything else the same.

"Forward pass":

\[
m_{x_i, y_i} \rightarrow x_i x_{i+1} (x_i) = \psi_i (x_i, y_i) = P(y_i | x_i)
\]

\[
m_{x_i y_i} \rightarrow x_{i+1} x_{i+2} (x_i) = \psi_{i+1} (x_{i+1}, y_{i+1}) = P(y_{i+1} | x_{i+1})
\]

\[
m_{x_1 x_2} \rightarrow x_2 x_3 (x_2) = \sum_{x_1} \psi_{x_1 x_2} (x_1, x_2) \cdot m_{x_1 y_1} \rightarrow x_1 x_2 (x_1) \cdot m_{x_2 y_2} \rightarrow x_2 x_3 (x_2)
\]

\[
= \sum_{x_1} P(x_1) P(x_2 | x_1) P(y_1 | x_1) P(y_2 | x_2)
\]

\[
= \sum_{x_1} P(x_1, x_2, y_1, y_2) \propto P(x_2 | y_1, y_2) P(x_2 | y_1, y_2)
\]

\[
m_{x_{i-1} x_i} \rightarrow x_i x_{i+1} (x_i) = \sum_{x_{i-1}} \psi_{x_{i-1} x_i} (x_{i-1}, x_i) \cdot m_{x_{i-2} x_{i-1}} \rightarrow x_{i-1} x_i (x_{i-1}) \cdot m_{x_i y_i} \rightarrow x_i x_{i+1} (x_i)
\]

\[
= \sum_{x_{i-1}} P(x_{i-1} | x_i) P(x_{i-1} | y_{i-1}, y_{i-2}) P(y_i | x_i)
\]

\[
= \sum_{x_{i-1}} P(x_{i-1} | x_i) P(x_{i-1} | y_{i-1}, y_{i-2}) \propto P(x_i | y_{i-1}, y_{i-2})
\]
Other variants and more discussions on junction trees can be found in


M. Jordan & C. Bishop. A Introduction to Graphical Models (draft).