These notes are intended to give a summary of relevant concepts from the lectures which are helpful to complete the exercises. It is not intended to cover the lectures thoroughly. Learning this content is not a replacement for working through the lecture material and the exercises.

Factor graph — A factor graph represents an arbitrary function in terms of factors and their connections with variables. For example, a factor graph can represent a distribution written as a Gibbs distribution –

\[
p(x) = \frac{1}{Z} \prod_c \phi_c(x_c)
\]

where variables \(x_i \in x\) are represented with variable nodes (circles) and potentials \(\phi_c\) are represented with factor nodes (squares). Edges connect each factor node \(\phi_c\) to all its variable nodes \(x_i \in X_c\).

Variable elimination — Given \(p(X) \propto \prod_c \phi_c(X_c)\), we compute the marginal \(p(X \setminus x^*)\) via the sum rule by exploiting the factorisation by means of the distributive law.

We sum out the variable \(x^*\) by first finding all factors \(\phi_i(X_i)\) such that \(x^* \in X_i\), and forming the compound factor \(\tilde{\phi}^*(\tilde{X}^*) = \sum x^* \phi^*(X^*)\) that does not depend on \(x^*\), i.e. \(\tilde{X}^* = X^* \setminus x^*\).

This is possible as products are commutative, and a sum can be distributed within a product as long as all terms depending on the variable(s) being summed come to the right of the sum.

\[
p(X \setminus x^*) \propto \sum_{x^*} \prod_c \phi_c(X_c) \propto \left[ \prod_{i \in x^* \notin X_i} \phi_i(X_i) \right] \left[ \sum_{x^*} \prod_{i \in x^* \in X_i} \phi_i(X_i) \right]
\]

When eliminating variables, order of elimination matters. However, optimal choice of elimination order is difficult. Picking variables greedily is a common heuristic, where the “best” \(x^*\) is the one that fewest factors \(\phi_c\) depend upon.

Sum-product algorithm — Variable elimination for factor trees reformulated with “messages” which allows for re-use of computations already done. See table on following page.

Max-product algorithm — Same as the sum-product algorithm, but max replaces \(\sum\).

Max-sum algorithm — Max-product algorithm in the log-domain. See table on following page.
Sum-product algorithm

\( \mu_{\phi \rightarrow x}(x) \)  
Factor to variable  
\[ \mu_{\phi \rightarrow x}(x) = \sum_{x_1, \ldots, x_j} \phi(x_1, \ldots, x_j, x) \prod_{i=1}^{j} \mu_{x_i \rightarrow \phi}(x_i) \]
where \( \{x_1, \ldots, x_j\} = \text{ne}(\phi) \setminus \{x\} \)

\( \mu_{x \rightarrow \phi}(x) \)  
Variable to factor  
\[ \mu_{x \rightarrow \phi}(x) = \prod_{i=1}^{j} \mu_{\phi_i \rightarrow x}(x) \]
where \( \{\phi_1, \ldots, \phi_j\} = \text{ne}(x) \setminus \{\phi\} \)

\( \tilde{p}(x) \)  
Univariate marginals – unnormalised  
\[ p(x) \propto \prod_{i=1}^{j} \mu_{\phi_i \rightarrow x}(x) \]
where \( \{\phi_1, \ldots, \phi_j\} = \text{ne}(x) \)

\( \tilde{p}(x_1, \ldots, x_j) \)  
Joint marginals of variables sharing a factor – unnormalised  
\[ p(x_1, \ldots, x_j) \propto \phi(x_1, \ldots, x_j) \prod_{i=1}^{j} \mu_{x_i \rightarrow \phi}(x_i) \]
where \( \{x_1, \ldots, x_j\} = \text{ne}(\phi) \)

Max-sum algorithm

\( \gamma_{\phi \rightarrow x}(x) \)  
Factor to variable  
\[ \gamma_{\phi \rightarrow x}(x) = \max_{x_1, \ldots, x_j} \log \phi(x_1, \ldots, x_j, x) + \sum_{i=1}^{j} \gamma_{x_i \rightarrow \phi}(x_i) \]
\[ \gamma_{\phi \rightarrow x}^*(x) = \arg \max_{x_1, \ldots, x_j} \log \phi(x_1, \ldots, x_j, x) + \sum_{i=1}^{j} \gamma_{x_i \rightarrow \phi}(x_i) \]
where \( \{x_1, \ldots, x_j\} = \text{ne}(\phi) \setminus \{x\} \)

\( \gamma_{x \rightarrow \phi}(x) \)  
Variable to factor  
\[ \gamma_{x \rightarrow \phi}(x) = \sum_{i=1}^{j} \gamma_{\phi_i \rightarrow x}(x) \]
where \( \{\phi_1, \ldots, \phi_j\} = \text{ne}(x) \setminus \{\phi\} \)

log \( \log p_{\text{max}} \)  
Maximum probability  
\[ \log p_{\text{max}} = \max_{x} \gamma^*(x), \quad \gamma^*(x) = - \log Z + \sum_{i=1}^{j} \gamma_{x_i \rightarrow \phi}(x) \]
where \( \{\phi_1, \ldots, \phi_j\} = \text{ne}(x) \)

argmax \( \text{argmax}_{x} \tilde{p}(x) \)  
Maximum probability states – no need for normalisation  
Init: \( \hat{x} = \text{argmax}_{x} \gamma^*(x) = \text{argmax}_{x} \sum_{i=1}^{j} \gamma_{x_i \rightarrow \phi}(x) \)
Backtrack to leaves via \( \gamma_{\phi \rightarrow x}^*(x) \)