Tree and Graph Search
A search problem consists of:

- A state space
- A successor function (with actions, costs)
- A start state and a goal test

A solution is a sequence of actions (a plan) which transforms the start state to a goal state.
We construct both on demand – and we construct as little as possible.

Each NODE in in the search tree is an entire PATH in the state space graph.
function Tree-Search(problem, fringe) return a solution, or failure
  fringe ← Insert(make-node(initial-state[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    for child-node in EXPAND(STATE[node], problem) do
      fringe ← INSERT(child-node, fringe)
    end
  end
end
**Graph Search Pseudo-Code**

```plaintext
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                if STATE[child-node] is not in closed then fringe ← INSERT(child-node; fringe)
            end
        end
    end
```
Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

**Cartoon of search tree:**
- $b$ is the branching factor
- $m$ is the maximum depth
- Solutions at various depths

**Number of nodes in entire tree?**
- $1 + b + b^2 + \ldots + b^m = O(b^m)$
Depth-First Search (DFS) Properties

- What nodes DFS expand?
  - Some left prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time $O(b^m)$

- How much space does the fringe take?
  - Only has siblings on path to root, so $O(bm)$

- Is it complete?
  - $m$ could be infinite, so only if we prevent cycles (more later)

- Is it optimal?
  - No, it finds the “leftmost” solution, regardless of depth or cost
Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be $s$
  - Search takes time $O(b^s)$

- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^s)$

- Is it complete?
  - $s$ must be finite if a solution exists, so yes!

- Is it optimal?
  - Only if costs are all 1 (more on costs later)
Uniform Cost Search

Strategy: expand a cheapest node first:
Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
  - Processes all nodes with cost less than cheapest solution!
  - If that solution costs $C^*$ and arcs cost at least $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
  - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)

- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$

- Is it complete?
  - Assuming best solution has a finite cost and minimum arc cost is positive, yes!

- Is it optimal?
  - Yes! (Proof next lecture via A*)
Uniform Cost Search

- Strategy: expand lowest path cost

- The good: UCS is complete and optimal!

- The bad:
  - Explores options in every “direction”
  - No information about goal location
Greedy Search

- **Strategy:** expand a node that you think is closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state

- **A common case:**
  - Best-first takes you straight to the (wrong) goal

- **Worst-case:** like a badly-guided DFS
A-star: Combining UCS and Greedy

- **Uniform-cost** orders by path cost, or *backward cost* $g(n)$
- **Greedy** orders by goal proximity, or *forward cost* $h(n)$

- **A* Search** orders by the sum: $f(n) = g(n) + h(n)$

Example: Teg Grenager
Admissible Heuristics

- A heuristic $h$ is **admissible** (optimistic) if:

$$0 \leq h(n) \leq h^*(n)$$

where $h^*(n)$ is the true cost to a nearest goal

- Examples:

- Coming up with admissible heuristics is most of what’s involved in using A* in practice.
Consistency of Heuristics

- **Main idea:** estimated heuristic costs ≤ actual costs
  - **Admissibility:** heuristic cost ≤ actual cost to goal
    \[ h(A) \leq \text{actual cost from A to G} \]
  - **Consistency:** heuristic “arc” cost ≤ actual cost for each arc
    \[ h(A) - h(C) \leq \text{cost}(A \text{ to } C) \]

- **Consequences of consistency:**
  - The \( f \) value along a path never decreases
    \[ h(A) \leq \text{cost}(A \text{ to } C) + h(C) \]
  - A* graph search is optimal
Optimality

- **Tree search:**
  - A* is optimal if heuristic is admissible
  - UCS is a special case (h = 0)

- **Graph search:**
  - A* optimal if heuristic is consistent
  - UCS optimal (h = 0 is consistent)

- Consistency implies admissibility

- In general, most natural admissible heuristics tend to be consistent, especially if from relaxed problems
Adversarial Search
Minimax Values

States Under Agent’s Control:
\[ V(s) = \max_{s' \in \text{successors}(s)} V(s') \]

States Under Opponent’s Control:
\[ V(s') = \min_{s \in \text{successors}(s')} V(s) \]

Terminal States:
\[ V(s) = \text{known} \]
Minimax Implementation

```
def min_value(state):
    initialize v = +\infty
    for each successor of state:
        v = min(v, max_value(successor))
    return v
```

```
def max_value(state):
    initialize v = -\infty
    for each successor of state:
        v = max(v, min_value(successor))
    return v
```

\[ V(s) = \max_{s' \in \text{successors}(s)} V(s') \]

\[ V(s') = \min_{s \in \text{successors}(s')} V(s) \]
Minimax Implementation (Dispatch)

```python
def value(state):
    if the state is a terminal state: return the state’s utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```python
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```python
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```
def min-value(state, α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ β return v
    β = min(β, v)
    return v

def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ α return v
    α = max(α, v)
    return v

α: MAX’s best option on path to root
β: MIN’s best option on path to root
Alpha-Beta Quiz
Uncertain Search
def value(state):
    if the state is a terminal state: return the state’s utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)

def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v

def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v

v = (1/2) (8) + (1/3) (24) + (1/6) (-12) = 10
Preferences

- An agent must have preferences among:
  - Prizes: $A$, $B$, etc.
  - Lotteries: situations with uncertain prizes

$$L = [p, A; (1-p), B]$$

$$p \cdot A + (1-p) \cdot B$$

- Notation:
  - Preference: $A \succ B$
  - Indifference: $A \sim B$
Expectimax Example
Expectimax Pruning?

We can’t prune unless we have bounds on the values of the leaves.
Multi-Agent Utilities

- What if the game is not zero-sum, or has multiple players?

- Generalization of minimax:
  - Terminals have utility tuples
  - Node values are also utility tuples
  - Each player maximizes its own component
  - Can give rise to cooperation and competition dynamically...

Each player only cares about their utility. If they cared about other players’ utilities that would already be included in their utility.
An agent must have preferences among:

- Prizes: $A$, $B$, etc.
- Lotteries: situations with uncertain prizes

$$L = [p, A; (1 - p), B]$$

Notation:

- Preference: $A \succ B$
- Indifference: $A \sim B$
### Rational Preferences

**The Axioms of Rationality**

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Orderability</strong></td>
<td>((A \succeq B) \lor (B \succeq A) \lor (A \sim B))</td>
</tr>
<tr>
<td><strong>Transitivity</strong></td>
<td>((A \succ B) \land (B \succ C) \Rightarrow (A \succ C))</td>
</tr>
<tr>
<td><strong>Continuity</strong></td>
<td>(A \succ B \succ C \Rightarrow \exists p \ [p, A; \ 1 - p, C] \sim B)</td>
</tr>
<tr>
<td><strong>Substitutability</strong></td>
<td>(A \sim B \Rightarrow [p, A; \ 1 - p, C] \sim [p, B; \ 1 - p, C])</td>
</tr>
<tr>
<td><strong>Monotonicity</strong></td>
<td>(A \succ B \Rightarrow (p \geq q \iff [p, A; \ 1 - p, B] \succeq [q, A; \ 1 - q, B]))</td>
</tr>
</tbody>
</table>

Theorem: Rational preferences imply behavior describable as maximization of expected utility.
Probability Distributions

- Unobserved random variables have distributions

\[
\begin{array}{c|c}
T & P \\
\hline
\text{hot} & 0.5 \\
\text{cold} & 0.5 \\
\end{array}
\quad
\begin{array}{c|c}
W & P \\
\hline
\text{sun} & 0.6 \\
\text{rain} & 0.1 \\
\text{fog} & 0.3 \\
\text{meteor} & 0.0 \\
\end{array}
\]

- A distribution is a TABLE of probabilities of values

- A probability (lower case value) is a single number

\[P(W = \text{rain}) = 0.1\]

- Must have: \( \forall x \ P(X = x) \geq 0 \) and \( \sum_x P(X = x) = 1 \)

Shorthand notation:

\[
P(\text{hot}) = P(T = \text{hot}),
\]

\[
P(\text{cold}) = P(T = \text{cold}),
\]

\[
P(\text{rain}) = P(W = \text{rain}),
\]

... OK if all domain entries are unique
Joint Distributions

- A joint distribution over a set of random variables: $X_1, X_2, \ldots X_n$ specifies a real number for each assignment (or outcome):

  \[ P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n) \]

  \[ P(x_1, x_2, \ldots x_n) \]

- Must obey:

  \[ P(x_1, x_2, \ldots x_n) \geq 0 \]

  \[ \sum_{(x_1, x_2, \ldots x_n)} P(x_1, x_2, \ldots x_n) = 1 \]

- Size of distribution if n variables with domain sizes d?

  - For all but the smallest distributions, impractical to write out!
Quiz: Events

- $P(+x, +y)$?
- $P(+x)$?
- $P(-y \ OR \ +x)$?

$P(X, Y)$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$Y$</td>
<td>$P$</td>
</tr>
<tr>
<td>$+x$</td>
<td>$+y$</td>
<td>0.2</td>
</tr>
<tr>
<td>$+x$</td>
<td>$-y$</td>
<td>0.3</td>
</tr>
<tr>
<td>$-x$</td>
<td>$+y$</td>
<td>0.4</td>
</tr>
<tr>
<td>$-x$</td>
<td>$-y$</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Marginal Distributions

- Marginal distributions are sub-tables which eliminate variables
- Marginalization (summing out): Combine collapsed rows by adding

\[
P(T, W)
\]

<table>
<thead>
<tr>
<th>T</th>
<th>W</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>sun</td>
<td>0.4</td>
</tr>
<tr>
<td>hot</td>
<td>rain</td>
<td>0.1</td>
</tr>
<tr>
<td>cold</td>
<td>sun</td>
<td>0.2</td>
</tr>
<tr>
<td>cold</td>
<td>rain</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[
P(T) = \sum_s P(t, s)
\]

\[
P(W) = \sum_t P(t, s)
\]

\[
P(X_1 = x_1) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2)
\]
Quiz: Marginal Distributions

\[ P(X, Y) \]

\[
\begin{array}{|c|c|c|}
\hline
X & Y & P \\
\hline
+x & +y & 0.2 \\
+x & -y & 0.3 \\
-x & +y & 0.4 \\
-x & -y & 0.1 \\
\hline
\end{array}
\]

\[
P(x) = \sum_y P(x, y)
\]

\[
P(y) = \sum_x P(x, y)
\]

\[
\begin{array}{|c|c|}
\hline
X & P \\
\hline
+x & \\
-x & \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
Y & P \\
\hline
+y & \\
-y & \\
\hline
\end{array}
\]
Conditional Probabilities

- A simple relation between joint and conditional probabilities
  - In fact, this is taken as the definition of a conditional probability

\[ P(a|b) = \frac{P(a, b)}{P(b)} \]

<table>
<thead>
<tr>
<th></th>
<th>W</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hot</td>
<td>sun</td>
<td>0.4</td>
</tr>
<tr>
<td>hot</td>
<td>rain</td>
<td>0.1</td>
</tr>
<tr>
<td>cold</td>
<td>sun</td>
<td>0.2</td>
</tr>
<tr>
<td>cold</td>
<td>rain</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[
P(W = s | T = c) = \frac{P(W = s, T = c)}{P(T = c)} = \frac{0.2}{0.5} = 0.4
\]

\[
= P(W = s, T = c) + P(W = r, T = c) = 0.2 + 0.3 = 0.5
\]
Quiz: Conditional Probabilities

\[ P(X, Y) \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>+x</td>
<td>+y</td>
<td>0.2</td>
</tr>
<tr>
<td>+x</td>
<td>-y</td>
<td>0.3</td>
</tr>
<tr>
<td>-x</td>
<td>+y</td>
<td>0.4</td>
</tr>
<tr>
<td>-x</td>
<td>-y</td>
<td>0.1</td>
</tr>
</tbody>
</table>

- \( P(+x \mid +y) ? \)
- \( P(-x \mid +y) ? \)
- \( P(-y \mid +x) ? \)

\[
P(-y \mid +x) = \frac{0.3}{0.5}
\]

\[
P(-y, +x) = \frac{P(-y, +x)}{P(+x)} = \frac{0.3}{0.5}
\]
Normalization Trick

\[ P(W = s | T = c) = \frac{P(W = s, T = c)}{P(T = c)} \]
\[ = \frac{P(W = s, T = c)}{P(W = s, T = c) + P(W = r, T = c)} \]
\[ = \frac{0.2}{0.2 + 0.3} = 0.4 \]

**SELECT** the joint probabilities matching the evidence

**NORMALIZE** the selection (make it sum to one)

\[ P(W | T = c) \]

<table>
<thead>
<tr>
<th>T</th>
<th>W</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>sun</td>
<td>0.4</td>
</tr>
<tr>
<td>hot</td>
<td>rain</td>
<td>0.1</td>
</tr>
<tr>
<td>cold</td>
<td>sun</td>
<td>0.2</td>
</tr>
<tr>
<td>cold</td>
<td>rain</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[ P(W = r | T = c) = \frac{P(W = r, T = c)}{P(T = c)} \]
\[ = \frac{P(W = r, T = c)}{P(W = r, T = c) + P(W = r, T = c)} \]
\[ = \frac{0.3}{0.2 + 0.3} = 0.6 \]
Inference by Enumeration

- **General case:**
  - Evidence variables: $E_1 \ldots E_k = e_1 \ldots e_k$
  - Query* variable: $Q$
  - Hidden variables: $H_1 \ldots H_r$

- We want:
  - $P(Q | e_1 \ldots e_k)$

- Step 1: Select the entries consistent with the evidence

- Step 2: Sum out $H$ to get joint of Query and evidence

- Step 3: Normalize

$$Z = \sum_q P(Q, e_1 \ldots e_k)$$

$$P(Q | e_1 \ldots e_k) = \frac{1}{Z} P(Q, e_1 \ldots e_k)$$

* Works fine with multiple query variables, too
Inference by Enumeration

- \( P(W) \)?
- \( P(W \mid \text{winter}) \)?
- \( P(W \mid \text{winter, hot}) \)?

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>T</th>
<th>W</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>summer</td>
<td>hot</td>
<td>sun</td>
<td>0.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hot</td>
<td>rain</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cold</td>
<td>sun</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cold</td>
<td>rain</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>winter</td>
<td>hot</td>
<td>sun</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hot</td>
<td>rain</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cold</td>
<td>sun</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cold</td>
<td>rain</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>
The Product Rule

- Sometimes have conditional distributions but want the joint

\[ P(y)P(x|y) = P(x, y) \quad \iff \quad P(x|y) = \frac{P(x, y)}{P(y)} \]
The Product Rule

\[ P(y) P(x|y) = P(x, y) \]

Example:

\[
\begin{array}{c|c|c}
   & D & W \\
---&---&---
R & wet & sun & 0.1 \\
   & dry & sun & 0.9 \\
   & wet & rain & 0.7 \\
   & dry & rain & 0.3 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
   & D & W \\
---&---&---
   & wet & sun \\
   & dry & sun \\
   & wet & rain \\
   & dry & rain \\
\end{array}
\]
The Chain Rule

- More generally, can always write any joint distribution as an incremental product of conditional distributions

\[ P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \]

\[ P(x_1, x_2, \ldots x_n) = \prod_{i} P(x_i|x_1 \ldots x_{i-1}) \]

- You can pick any order.

- Why is the Chain Rule always true?
Bayes’ Rule

- Two ways to factor a joint distribution over two variables:

\[ P(x, y) = P(x|y)P(y) = P(y|x)P(x) \]

- Dividing, we get:

\[ P(x|y) = \frac{P(y|x)P(x)}{P(y)} \]

- Why is this at all helpful?
  - Lets us build one conditional from its reverse
  - Often one conditional is tricky but the other one is simple
  - Foundation of many systems (e.g. ASR, MT, IRL)

- In the running for most important AI equation!
Inference with Bayes’ Rule

- Example: Diagnostic probability from causal probability:

\[ P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})P(\text{cause})}{P(\text{effect})} = \frac{P(\text{effect}|\text{cause})}{P(\text{cause})} P(\text{cause}) \]

- Example:
  - M: meningitis, S: stiff neck
  - \( P(+m) = 0.0001 \)
  - \( P(+s|+m) = 0.8 \)
  - \( P(+s|-m) = 0.01 \)

\[
P(+m|+s) = \frac{P(+s|+m)P(+m)}{P(+s)} = \frac{P(+s|+m)P(+m)}{P(+s|+m)P(+m) + P(+s|-m)P(-m)} = \frac{0.8 \times 0.0001}{0.8 \times 0.0001 + 0.01 \times 0.999}
\]

- Note: posterior probability of meningitis still very small
- Note: you should still get stiff necks checked out! Why?
Quiz: Bayes’ Rule

- Given:

  \[
  \begin{array}{c|c|c}
  \text{D} & \text{W} & \text{P} \\
  \hline
  \text{wet} & \text{sun} & 0.1 \\
  \text{dry} & \text{sun} & 0.9 \\
  \text{wet} & \text{rain} & 0.7 \\
  \text{dry} & \text{rain} & 0.3 \\
  \end{array}
  \]

- What is \( P(W \mid \text{dry}) \)?
Independence

- Two variables are independent in a joint distribution if:

\[ P(X, Y) = P(X)P(Y) \]
\[ \forall x, y \ P(x, y) = P(x)P(y) \]

- Says the joint distribution factors into a product of two simple ones
- Usually variables aren’t independent!

- Can use independence as a modeling assumption
  - Independence can be a simplifying assumption
  - Empirical joint distributions: at best “close” to independent
  - What could we assume for \{Weather, Traffic, Cavity\}? 

- Independence is like something from CSPs: what?
Example: Independence?

\[ P_1(T, W) \]

<table>
<thead>
<tr>
<th>T</th>
<th>W</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>sun</td>
<td>0.4</td>
</tr>
<tr>
<td>hot</td>
<td>rain</td>
<td>0.1</td>
</tr>
<tr>
<td>cold</td>
<td>sun</td>
<td>0.2</td>
</tr>
<tr>
<td>cold</td>
<td>rain</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[ P(T) \]

<table>
<thead>
<tr>
<th>T</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>0.5</td>
</tr>
<tr>
<td>cold</td>
<td>0.5</td>
</tr>
</tbody>
</table>

\[ P_2(T, W) = P(T)P(W) \]

\[ P(W) \]

<table>
<thead>
<tr>
<th>W</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>0.6</td>
</tr>
<tr>
<td>rain</td>
<td>0.4</td>
</tr>
</tbody>
</table>

\[ P(T) \]

<table>
<thead>
<tr>
<th>T</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>0.3</td>
</tr>
<tr>
<td>hot</td>
<td>0.2</td>
</tr>
<tr>
<td>cold</td>
<td>0.3</td>
</tr>
<tr>
<td>cold</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Conditional Independence

- Unconditional (absolute) independence very rare (why?)

- *Conditional independence* is our most basic and robust form of knowledge about uncertain environments.

- X is conditionally independent of Y given Z if and only if:

\[ P(x, y|z) = P(x|z)P(y|z) \]

or, equivalently, if and only if

\[ \forall x, y, z : P(x|z, y) = P(x|z) \]
Probability Recap

- **Conditional probability**
  \[ P(x|y) = \frac{P(x, y)}{P(y)} \]

- **Product rule**
  \[ P(x, y) = P(x|y)P(y) \]
  \[ = P(x_2)P(x_3|x_2)P(x_4|x_2, x_3) \]

- **Chain rule**
  \[ P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots \]
  \[ = \prod_{i=1}^{n} P(X_i|X_1, \ldots, X_{i-1}) \]

- **X, Y independent if and only if:**
  \[ \forall x, y : P(x, y) = P(x)P(y) \]

- **X and Y are conditionally independent given Z if and only if:**
  \[ \forall x, y, z : P(x, y|z) = P(x|z)P(y|z) \]
Queue

\[(S, 0)\]
\[(S, A, 1)\] \[(S, B, 1)\]
\[(S, 5, 1)\] \[(5, A - C, 2)\] \[(S, A, 2, 2)\] \[(S, A - C - G, 3)\]

\[h(B) - h(D) \geq \text{cost}(B)\]

\[3\]
Fill in the following table with the joint distribution $P(I, M, R)$:

<table>
<thead>
<tr>
<th>$I$</th>
<th>$P(I)$</th>
<th>$R$</th>
<th>$P(R)$</th>
<th>$P(M \mid I, R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+i$</td>
<td>0.8</td>
<td>$+r$</td>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>$-i$</td>
<td>0.2</td>
<td>$-r$</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>$+i$</td>
<td>$+r$</td>
<td>$+m$</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>$-i$</td>
<td>$+r$</td>
<td>$-m$</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>$-i$</td>
<td>$-r$</td>
<td>$+m$</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>$-i$</td>
<td>$-r$</td>
<td>$-m$</td>
<td>0.8</td>
<td>0</td>
</tr>
</tbody>
</table>

$$P(I, R, M) = P(I) \cdot P(R) \cdot P(M \mid I, R)$$

- $P(I, R, M)$ for $I = +i, R = +r, M = +m$:
  - $P(I) = 0.8$
  - $P(R) = 0.4$
  - $P(M \mid I, R) = 0$
  - $P(I, R, M) = 0.8 \cdot 0.4 \cdot 0 = 0$

- $P(I, R, M)$ for $I = +i, R = +r, M = -m$:
  - $P(I) = 0.8$
  - $P(R) = 0.4$
  - $P(M \mid I, R) = 1$
  - $P(I, R, M) = 0.8 \cdot 0.4 \cdot 1 = 0.32$