Search:

1. BFS: The BFS search strategy explores the shallowest node in the search tree.
queue = collections.deque(); queue.popleft()
2. DFS: explores the deepest node in the search tree
queue.pop()
3. UCS: explores the cheapest node first
[(2, 'C'), (7, 'B'), (6, 'A')] heappop(frontier) -> (2, 'C')
4. Heuristic Function: h(n)
○ Cost (estimate) of the cheapest path from the state at node n to a goal state
○ If n is a goal node h(n) = 0
5. A*: orders by backward cost + forward cost
f(n) = g(n) + h(n):
g: add values on all paths to n, h: the value on n
6. A heuristic h(n) is admissible (optimistic) if
0 <= h(n) <= h*(n),
where h*(n) is the true cost to the nearest goal from n
7. A* is optimal if an admissible heuristic is used
8. Heuristic cost ≤ actual cost for each arc:
h(a)-h(c) ≤ cost(a to c)
9. Consequence of consistency: The f value along a path never decreases, h(a) ≤ cost(a to c) + h(c)
10. Constraint Satisfaction Problems (CSP):
    (1) A set of variables, X = {X1, ... , Xn}
    (2) A set of domains, D = {D1, ... , Dn}, where Di = {v1, ... , vk} for each variable Xi
    (3) A set of constraints C which specify allowable combinations of values
To solve a CSP we need to define a state space:
Each state is defined by an assignment of values to some or all variables {Xi = vi, Xj = vj, ...}
11. Minimax: Always starts from MAX

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(\\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

12. Depth-Limit Search (DLS): Search only to a limited depth in the tree
13. α-β Pruning Algorithm:Min version
Consider Min's value at some node n
n will decrease (or stay constant) while the descendants of n are examined
Let m be the best value that Max can get at any choice point along the current path from the root: If n becomes worse (<) than m, Max will avoid it, and Stop considering n's other children
14. Expectimax Search: take weighted average of children

Markov Decision Processes:

1. Discounted rewards:

$$U([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

$$U([s_0, s_1, s_2, \dots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \le \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max}/(1-\gamma)$$

2. Value Iteration:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

Reinforcing Learning:

1. Temporal Difference Learning:

Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha) sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

2. Q-Learning:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right]$$

3. Now is evaluating <u>Q-value</u>, not value.
4. Exploration Functions: Takes a value estimate u and a visit count n, and returns an optimistic utility

$$Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

$$Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$$

5. Approximate Q-Learning:
transaction = (s, a, r, s')

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s',a')\right] - Q(s,a)$$

$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}]$ Exact Q's

$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a)$ Approximate Q's

Markov Models:

1. Marginal Distributions: P(t) = $\sum_s P(t,s)$
2. Conditional Distributions: P(a|b) = P(a, b) / P(b)
3. The product rule: P(y) * P(x|y) = P(x, y)
4. The chain rule:

$$P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3$$

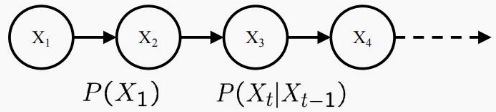
5. Bayes' rule:

$$P(x|y) = \frac{P(y|x)}{P(y)} P(x)$$

6. Independence: P(X, Y) = P(X) * P(Y)

$$X \perp\!\!\!\perp Y$$

7. Conditional Independence:
X is conditionally independent of Y given Z iff any x, y, z
    (1) P(x, y|z) = P(x|z) * P(y|z)
    (2) P(x|z, y) = P(x|z)
8. Joint Distribution of a MM:



Joint distribution:

$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)$$

From the chain rule, every joint distribution over X1, X2, X3, X4 can be written as:

$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(X_4|X_1, X_2, X_3)$$

Assuming: $X_3 \perp\!\!\!\perp X_1 \mid X_2$ and $X_4 \perp\!\!\!\perp X_1, X_2 \mid X_3$

$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)$$

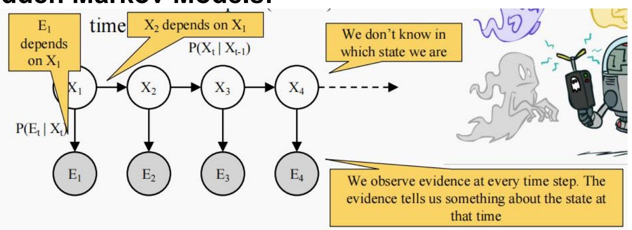
9. Mini-Forward Algorithm:

$$P(x_t) = \sum_{x_{t-1}} P(x_{t-1}, x_t) = \sum_{x_{t-1}} P(x_t \mid x_{t-1}) P(x_{t-1})$$

10. Stationary distribution:

$$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x) P_\infty(x)$$

Hidden Markov Models:



1. Joint distribution of an HMM:

$$P(X_1, E_1, X_2, E_2, X_3, E_3) = P(X_1)P(E_1|X_1)P(X_2|X_1)P(E_2|X_2)P(X_3|X_2)P(E_3|X_3)$$

2. Conditional Independence: Current observation

independent of all else given current state. eg:

$$E_1 \perp\!\!\!\perp X_2, E_2, X_3, E_3 \mid X_1$$

3. Filtering / Monitoring:

$$B_t(X) = P(X_t \mid e_1, \ldots, e_t)$$

4. Passage of time:
(1) Assume we have current belief:
B(Xt) = P(Xt | e1:t)
(2) Then after one time step passes:

P(Xt+1 | e1:t) = $$\sum_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t})$$

(3) Or: $$B'(X_{t+1}) = \sum_{x_t} P(X'|x_t)B(x_t)$$

(4) Beliefs get "pushed" through the transitions
5. Observations:
(1) Assume we have B'(Xt+1)
(2) After evidence comes in:

P(Xt+1 | e1:t+1) = $$P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})$$

(3) $$B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1}|X_{t+1})B'(X_{t+1})$$

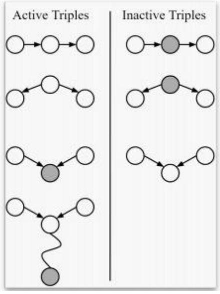
6. The forward algorithm:
We are given evidence at each time and wanna know Bt(X) = P(Xt | e1:t)

$$= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}, e_{1:t-1})$$

Bayes Nets:
1. Arcs: encode conditional independence
2. Probabilities in BNs:

$$P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i))$$

3. A path is active if each triple is active:
○ Causal chain A → B → C where B is unobserved (either direction)
○ Common cause A ← B → C where B is unobserved
○ Common effect (aka v-structure) A → B ← C where B or one of its descendents is observed
4. Maximum Expected Utility (MEU): maximize utilities



5. D-Separation:
Check all (undirected!) paths between and Xi and Xj
● If one or more active, then independence not guaranteed
● Otherwise (i.e. if all paths are inactive), then independence is guaranteed
6. Preferences:
(1) A > B, prefer A to B, U(A) > U(B)
(2) A ~ B: indifferent, U(A) = U(B)
(3) U(p1, S1; p2, S2; … ; pn, Sn) = ∑ pi * U(Si)

NLP:
1. Word2Vec: Go through each position t in the text, which has a center word c and context words o
2. Use the similarity of the word vectors for c and o to calculate the probability of o given c, i.e., P(o | c)
3. maximize L:

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j}|w_t;\theta)$$

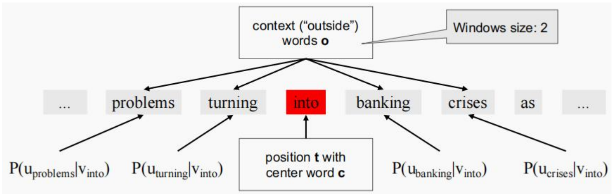
4. Or minimize J, which is average negative log likelihood:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j}|w_t;\theta)$$

5. Calculate P(Wt+j | Wt; θ):
(1) Use **two vectors** per word w:
○ vw when w is a center word
○ uw when w is a context ("outside") word
(2) Average both at the end to obtain a single vector representation per word

(3) Then for a center word c and a outside word o:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$



6. Parameters:
(1) Recall that all word vectors are squeezed into θ
(2) Assume we have d-dimensional vectors and V many words
(3) Every word has two vectors!

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_{abacus} \\ \vdots \\ v_{zebra} \\ v_{zyzzyva} \\ u_{aardvark} \\ u_{abacus} \\ \vdots \\ u_{zebra} \\ u_{zyzzyva} \end{bmatrix} \in \mathbb{R}^{2dV}$$

7. Optimization: Gradient Descent
To train the model, we gradually adjust parameters to minimize J
○ We change the parameters by walking down the gradient (aka slope in 1D)
■ We are going to have to take the (partial) derivative!
Steps:
(1) In gradient descent, we considers the rate of change of J with respect to θ, i.e. $\frac{\partial J}{\partial \theta}$
(2) Then we take steps proportional to the negative of the gradient