COMS 4701: Artificial Intelligence

Final Sample Solutions

Problem 1

1. All algorithms lose their guarantee of completeness. Given a start-goal pair for which no solution exists, search would never end. **If an answer only addresses cases where solutions do exist, then BFS, UCS, and A* are still complete, as they will eventually expand the solution. The lack of a reached set can cause nodes to be repetitively expanded, but that does not prevent the solution from being reached.**

2. DFS and BFS were not optimal to begin with and still not guaranteed to be optimal since they do not consider true costs. UCS and A* are still guaranteed to be optimal. Without a reached table, the frontier would contain many more nodes than before, but nodes are still popped from the frontier in order of increasing cost.

3. Still admissible; this has nothing to do with the implementation of the underlying best-first search.

Problem 2

$$v_0 \leq \sum_{i=1}^{k} p_i v_i + v_{\max} \sum_{i=k+1}^{n} p_i = \sum_{i=1}^{k} p_i v_i + v_{\max} \left(1 - \sum_{i=1}^{k} p_i\right)$$

Sample solution:

```
v = 0
p_so_far = 0
for a,p in actions(state):
    if v+(1-p_so_far)*V_MAX <= alpha: return v
    v += p*X_value(result(state,a))
    p_so_far += p
return v
```

Points we were looking for:

- Looping over action-probability pairs
- Comparison of `v` so far to determine pruning condition
- Correct update of `v` so far
- Correct update of `p` so far OR usage of "future" probabilities

Problem 3

1. The equations are

$$\begin{aligned} r_1 + \gamma V^\pi(s_2) - V^\pi(s_1) &= 0 \\ r_2 + \gamma V^\pi(s_3) - V^\pi(s_2) &= 0 \\ &\vdots \\ r_{n-1} + \gamma V^\pi(s_n) - V^\pi(s_{n-1}) &= 0 \end{aligned}$$

2. The number of unknowns is equal to the total number of states. It is possible to have more unique equations, since the agent may have encountered different successor states coming from the same state in the observed sequence.

Sample solution:

```
max_diff = 0
for i in range(len(states)-1):
  diff = alpha * (rewards[i] + gamma*V[states[i+1]] - V[states[i]])
  max_diff = max(max_diff, abs(diff))
  V[states[i]] += diff
```

Points we were looking for:

- Looping over all encountered states (not the same as the set of unique states!)
- Correct computation of the TD error, indexing both `rewards` and `states`
- Correct update of `max_diff` (absolute value)
- Correct update of value function

Problem 4

1. $\Pr(F_q \mid f_e) \propto \sum_y \Pr(y) \Pr(f_e \mid y) \Pr(F_q \mid y)$
2. A minimal sampling procedure would only need to sample two variables, Y and F_q, while f_e remains fixed. The sampling distributions are $\Pr(Y)$ and $\Pr(F_q \mid y)$, respectively, both contained in the Naive Bayes parameters. Each sample has a weight equal to $\Pr(f_e \mid y)$. Once sampling is finished, we add up all the weights for each value of F_q and normalize to obtain the desired distribution. (Note that it is still correct but unnecessary to sample all other features as well).

Sample solution:

```
dist = np.sum(fq_given_y * prior * fe_given_y, axis=1)
return dist / np.sum(dist)
```

The main challenge here is to keep in mind that `fq_given_y` is a 2D array/matrix. The above operation in Numpy multiplies the 1D array onto each row of the 2D array separately. We then compute the sum of each row to obtain the desired solution. Don't forget to normalize!