Please add your answers under the question number. If there is already an answer, check it. If you agree with it, add a :[1]) underneath to verify it. Lots of :) means it's a good answer! If you have questions or disagree with an answer, pop a comment! If you want to add notes or explanations of problems, create a new document in this folder and link to it below the relevant question.

- Akira in red font :^)
- Jin
- Rowan in Green
- John

Question 1

~~a)~~ 1

- Believe it should be 1. In general, expectiminimax gives perfect play (although it is useless and we usually end up with 3-ply). This is the "general" case, not the "common" case. :) :) ☺ ☺

b) ~~1~~ ☺ ☺ ☺ I think this should be 3, BFS is not optimal in general, only when step cost is uniform (also note the question refers to BFS in a finite tree)
3 (*quoting from Week 2 Slide 25 [Properties of BFS]: "(BFS is) not optimal in general"*

c) 2 ☺ ☺ ☺

d) 3 ☺ ☺ ☺

e) 3 ☺ ☺ ☺

Question 2

a) The learning rate controls the speed at which the system 'changes its mind'. A high learning rate in TDleaf means that differences in state evaluation functions will have a large impact on the weights. ☺ ☺

b) Using temporal difference between successive states leads to a more accurate overall estimation of each move's impact, as weights will be updated so that 'shallower' states more accurately reflect the utility of the states to which they lead.

c) When lambda is 0, weights are adjusted to make the estimated reward of each state closely approximate that of the next state, rather than the final utility. This is useful when the evaluation function is not very good. ☺
- I had that it will encourage the algorithm to make the current reward more similar to the next state (i.e we don't want to modify weights much). This means we use \lambda = 0 when we have a stable algorithm with good results in the latter stages of training
- Intuitively, the formula for updating weights is dependent on \lambda, hence having small \lambda gives a small update in weights (not accounting for the learning hyperparameter)
We tend to encourage the algorithm to make the current state to the next state, when we have stable algorithm.

[1] 信じたのもは、都合のいい妄想を繰り返し映し出す鏡。 #Lyricova

d) The auctioneer should use a second-price sealed-bid auction. Bidders are more likely to bid their true values (truth revealing), as a second-price auction mitigates the winner's curse. As two buyers value the painting at $5 million, the second price will end up being the same as the first, so this type of auction results in the highest reward for the auctioneer. ☺ ☺

Question 3

a) 7 ☺ ☺ ☺
b) 8, 11, 16-21, 26-31 ☺ ☺ ☺
c)

~~i) Iterative-deepening minimax (IDM) guarantees to find an optimal solution, while depth-limited minimax (DLM) might not, depending on the depth specified.~~

- Hmm I think depth limited is still optimal in this case since a game tree (is a tree hence no loops) and is a finite space.
- I had iterative deepening allows alpha-beta to work effectively since it can prune branches much faster (since we have a shallow searching space initially) :)
- Also had iterative deepening has a much smaller space complexity and is advantageous if space is an issue agree
- If you have time constraints , depth-limited search could either run out of time or not utilize all the time (it doesn't finish in time or it finishes too early). Where as iterative deepening wont. (hope that makes sense)

~~ii) IDM automatically stops at the best depth during the search, while DLM relies on an arbitrary depth predefined which might not be optimal.~~

- IDS can save substantial amount of time if $d < l$.
- DLS might not always return an optimal solution as it is similar to DFS, but with a depth limit. [Ref: Martin Yong]
- Iterative deepening minimax search can make better use of the time and space constraints. [Ref: Yinsong Chen]

Question 4

a) 25 ☺ ☺ ☺
b) N^2 ??

Considering the symmetry between the 2 people, the problem space size is $\frac{N^4}{2}$ ☺

Expl.: N^2 positions per person, squared for 2 people. Divided by 2 as swapping the 2 person is essentially the same problem. Not considering symmetry of map as wall could be non-symmetry.[2]

- Since it is big-oh, we only need to give an upper bound to the state space. Hence O(n^n) is technically a correct upper bound /s

c) Manhattan distance between people? ☺
   - Does manhattan count as non-trivial? (I wasn't sure) Yes.
     - *A trivial admissible heuristic would be something like h(n) = 0 for all n.* [Ref. Matt]

Question 5

a) A=r, B={b,g}, C={g}, D={g}, E=b
   - A={r}, B={g,b},C={g},D={g},E={b} ☺:) ☺
   - This is forward check so we only update neighbours
   - A has neighbours B,C,D and E has neighbours C,D

b) C. It is the most constrained variable.
   ** Removing C makes the remaining graph acyclic, suitable for cutset conditioning.*

c) O(n * d^2).
   Steps to solve tree-structured condition satisfaction problem:
   i) Topological sort: $\mathcal{O}(n+c)$ where c is the number of constraints
   ii) Arc consistency check from leaves to root: $\mathcal{O}(nd^2)$
   iii) Assign values from root to leaves: $\mathcal{O}(nd)$

   Taking the largest component: $\mathcal{O}(nd^2)$
   - For a tree, we get e = n-1, so O(e) = O(n). Each arc is only checked once, I believe, rather than O(d) times. Thus the runtime is O(ed^2) = O(nd^2)

   In a tree structured graph, each of n constraints need to be checked only once for arc consistency, and each check involves checking each of d possible values for each of two variables in the constraint. Therefore the constraint is O(nd^2)

Question 6

a) No. A is dependent on C and V. A consistent notation would be P(C) * P(V) * P(A|C, V).
   $\mathrm{P(C, A, V)}$
   $= \mathrm{P(C|Parents(C))P(A|Parents(A))P(V|Parents(V))}$
   $= \mathrm{P(C)P(V)P(A|C, V)} \neq \mathrm{P(C)P(A)P(V)}$ ☺

b) Yes. G always depends on A, so explicitly including A in the distribution does not change anything.

---

[2] The goal of the problem is to get the 2 people onto one certain spot. It doesn't matter who is the first one or the second one. Thus there are redundancy in N^4, all (coord1, coord2) and (coord2, coord1) are counted twice.

$$
\begin{aligned}
P(J|G,A) &= \frac{P(J,G,A)}{P(G,A)} \\
&= \frac{\sum_C \sum_V P(J,G,A,C,V)}{\sum_J \sum_C \sum_V P(J,G,A,C,V)} \\
&= \frac{P(J|G)\sum_C P(C)\sum_V P(V)P(G|C,A,V)P(A|C,V)}{\sum_J P(J|G)\sum_C P(C)\sum_V P(V)P(G|C,A,V)P(A|C,V)} \\
&= \frac{P(J|G)\sum_C \sum_V P(G|C,A,V)P(A|C,V)}{\sum_J P(J|G)\sum_C \sum_V P(G|C,A,V)P(A|C,V)} \\
&= \frac{P(J|G)}{\sum_J P(J|G)} \\
&= P(J|G)
\end{aligned}
$$

[3]

c) $P(j \mid c,a,v)$

$= \frac{P(j,c,a,v)}{P(c,a,v)}$

$= \frac{\sum_g P(j,g,c,a,v)}{P(c)P(v)P(a \mid c,v)}$

$= \frac{\sum_g P(j|g)P(g|c,a,v)P(c)P(v)P(a|c,v)}{P(c)P(v)P(a|c,v)}$

$= \sum_g P(j|g)P(g|c,a,v)$

$= P(j|g)P(g|c,a,v) + P(j|\neg g)P(\neg g|c,a,v)$

$= 0.9 \times 0.9 + 0.0 \times 0.1$

$= 0.81$ ☹ ☺

Question 7

[3] LaTeX source of equations is attached to the alt text of pictures if you want to make changes to it.

a) $\frac{2}{n+1}$

$$
\begin{aligned}
P(t|h) &= \frac{P(h|t)P(t)}{P(h)} \\
&= \frac{1 \times \frac{1}{n}}{P(h|t)P(t) + P(h|\neg t)P(\neg t)} \\
&= \frac{1}{n} \times \frac{1}{1 \times \frac{1}{n} + 0.5\left(1 - \frac{1}{n}\right)} \\
&= \frac{1}{n} \times \frac{1}{0.5\left(1 + \frac{1}{n}\right)} \\
&= \frac{1}{n} \times 2 \times \frac{1}{1 + \frac{1}{n}} \\
&= \frac{2}{n} \times \frac{n}{n+1} \\
&= \frac{2}{n+1}
\end{aligned}
$$

1 Introduction

Let h=head, t=biased coin, $\neg t$=unbiased coin. Then we can infer that $P(t) = \frac{1}{n}$ and $P(\neg t) = \frac{n-1}{n}$.

Hence,

$$
\begin{aligned}
P(t|h) &= \alpha P(h|t)P(t), \qquad \alpha = \frac{1}{P(h)} \\
&= \alpha \times 1 \times \frac{1}{n}
\end{aligned}
$$

Then for α (take the reciprocal for easier typing out),

$$
\begin{aligned}
\frac{1}{\alpha} &= P(h) \\
&= P(h|t)P(t) + P(h|\neg t)P(\neg t) \\
&= \frac{1}{n} + \left(\frac{1}{2}\right)\left(\frac{n-1}{n}\right) \\
&= \frac{2}{2n} + \frac{n-1}{2n} \\
&= \frac{n+1}{2n}.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
P(t|h) &= \left(\frac{2n}{n+1}\right)\left(\frac{1}{n}\right) \\
&= \frac{2}{n+1}.
\end{aligned}
$$

b) $\frac{2^k}{2^k+n-1}$

$$
\begin{aligned}
P_k(t) &= \frac{P(h|t)P_{k-1}(t)}{P(h|t)P_{k-1}(t) + P(h|\neg t)P_{k-1}(\neg t)} \\
&= \frac{P_{k-1}(t)}{P_{k-1}(t) + 0.5(1 - P_{k-1}(t))} \\
&= 2\frac{P_{k-1}(t)}{1 + P_{k-1}(t)} \\
&= 2 - \frac{2}{1 + P_{k-1}(t)}
\end{aligned}
$$

Also, $P_0(t) = \frac{1}{n}$, by mathematical induction, $P_k(t) = \frac{2^x}{2^x+n-1}$. *(derived by Wolfram|Alpha)*

Let H = heads, T = trick coin. Use h^k to denote k consecutive heads.

$$
\begin{aligned}
P(t|h^k) &= \frac{P(h^k|t)P(t)}{P(h^k)} = \frac{P(h^k|t)P(t)}{P(h^k|t)P(t) + P(h^k|\neg t)P(\neg t)} \\
&= \frac{1 \times \frac{1}{n}}{1 \times \frac{1}{n} + \frac{1}{2^k}\left(1 - \frac{1}{n}\right)} = \frac{\frac{1}{n}}{\frac{1}{n} + \frac{1}{2^k} - \frac{1}{n2^k}} = \frac{2^k}{2^k + n - 1}
\end{aligned}
$$

Here is LaTeX of my solution. It at least checks out for k=1 haha

> *$P(h^k|t)$ is the probability of k consecutive heads given a trick coin. The trick coin is ALWAYS heads, so this is 1 no matter what k is. $P(h^k|\neg t)$ is the probability of k consecutive heads given a standard coin. This is $(1/2)^k = 1/2^k$ since a standard coin has a $1/2$ chance of showing heads.*

P.S.: Equation editor in Google Docs is really a pain to use.