Question 1

An Algorithm

The algorithm is a variation of the DFS algorithm taught in class.

The Algorithm: In stage $t$ of the algorithm each of the robots searches all the squares of distance $n^t$ (say, rounded up to the nearest integer). The algorithm ends if the two robots map the same square. The order of the stages is $\frac{1}{2}, \frac{1}{2} + \epsilon, \ldots , 1$. If the algorithm didn’t end at one of the stages we simply search all board.

Notice that the number of stages is at most constant, for every constant $\epsilon > 0$.

Theorem 0.1 The algorithm provides a competitive ratio of $n^{1+\epsilon}$, for any fixed $\epsilon > 0$.

Proof: Suppose that the optimal solution is of distance $x$. Now, in each stage $t$ each of the robots explores (via the DFS) $O(n^{2t})$ squares (as we have seen in class). If $x \leq \sqrt{n}$ we’ll find a meeting path in the first stage, and get a competitive ratio of $O(n)$.

The total distance traveled up to stage $t$ (including the distance traveled in stage $t$) is the distance traveled in stage $t$ plus the total distance traveled in all of the previous stages. The total distance traveled in previous stages can be easily upper bounded by $O(n^{2t-2\epsilon})$ (the number of previous stages times the distance traveled in the previous stage). So the total distance up to now is: $O(n^{2t-2\epsilon}) + O(n^{2t}) = O(n^{2t})$. Therefore, if $X \geq n$, the total distance traveled is $O(n^2)$, giving us a competitive ratio of at most $O(n)$.

Else, let $t$ be the minimal stage in the algorithm where $n^t \geq x$. Observe that $n^{t-\epsilon} \leq x \leq n^t$. The competitive ratio is at most $\frac{O(n^{2t})}{n^{t-\epsilon}} = O(n^{t+\epsilon}) = O(n^{1+\epsilon})$, since $t$ takes values between $\frac{1}{2}$ and 1. \qed

A Lower Bound

Similarly to the lower bound presented in class, consider the following instance:
Quick legend: the instance represents an $n \times n$ map, one robot is on the upper left corner, the other one is on the upper right corner (both marked in $R$). An occupied square is marked in $X$.

Notice that there are about $\frac{n}{2}$ corridors, and to get to middle of it, a robot need to travel a distance of $\Theta(n)$. If we remove the barrier between the two parts of one of the corridors, we get that the optimal solution is that both robots have to travel together a distance of $\Theta(n)$. However, in order to know which barrier was removed, at least one robot has to travel a distance of $\Theta(n)$. Since there are $n$ corridors, in the worst case the total traveling distance of both robots (in order to find the path) is $\Theta(n^2)$, which gives us a lower bound of $\Omega(n)$ on the competitive ratio, as needed.

**Question 2**

1. We assume that $i$ is equal to the $j$-width of the graph, and that the problem is strongly $(i,j)$-consistent, and we prove that there exists a vertical search order that guarantees $j$-bounded backtrack search. By definition, there exists a vertical order $o$ on $G$ with $j$-width $i$. Let $v_l$ be the $l$-th variable in $o$. There exists $k$, $k \leq j$ and a set $v_{l-k+1}, \ldots, v_l$ of variables that depend on at most $i$ preceding variables $v_1, \ldots, v_{m}$ ($m \leq i$). Since the problem is strongly $(i,j)$-consistent, it is possible to complete the assignment to $v_1, \ldots, v_{m}$ with some assignment to $v_{l-k+1}, \ldots, v_l$ to a legal assignment to all these $m+k$ variables. This way we are reconsidering only the assignment to $v_{l-k+1}, \ldots, v_{l-1}$ and giving a value to $v_l$, i.e. we are doing $k$-bounded backtrack search, and since $k \leq j$, it is also a $j$-bounded backtrack search. We will be doing such a $j$-bounded backtrack search for each $v_l$, and hence, by induction on $l$, it follows that $o$ guarantees $j$-bounded backtrack search.

2. A counterexample for the claim: $x \in \{3,5\}$, $y \in \{6,10\}$, $z \in \{3,5\}$. The constraints are $x|y$, $x|z$. The problem is definitely $(1,2)$-consistent, but it is not $(2,1)$-consistent, because if we have an assignment $y = 6$ and $z = 5$, then we cannot add an assignment to $x$ to have a legal assignment to $x, y$ and $z$.

**Question 3**

1. We cannot use this algorithm to efficiently manipulate Dodgson’s rule because determining the Dodgson score of a candidate is $NP$-hard, so in the iterative step of the algorithm it will be $NP$-hard to determine whether we can place some candidate in the next available spot without preventing $p$ from winning.

2. **The Algorithm:** Let $A$ be the set of candidates that beat $a$ in pairwise elections.

   
   $l = 0$
   
   while $A \neq \emptyset$ do
Let $S$ be the set of voters who do not have $a$ on top of their preferences.
All the voters in $S$ push $a$ one place upper
$l = l + |S|$
recalculate $A$
endwhile
return $l$

**Theorem 0.2** The above algorithm is an $n$-approximation to the Dodgson score problem.

**Proof:** Let $t$ be the Dodgson score of $a$. The algorithm returns (not necessarily the least) number of exchanges between adjacent alternatives which are made till $a$ becomes a Condorcet winner (till $A = \emptyset$), and so the value returned by the algorithm, $l \geq t$. We need to show that $l \leq nt$. Let $\succ^*$ be the profile which is obtained after $t$ optimal exchanges of places between adjacent alternatives such that $a$ is a Condorcet winner in $\succ^*$. After $t$ iterations of the while loop of the algorithm, $a$ is ranked by every voter $i$ at least as high as in $\succ^*$, and all the rest candidates are ranked in the same places, and so $a$ will be a Condorcet winner also after $t$ iterations of the algorithm. In each iteration $l$ is growing by at most $n$, and so after $t$ iterations the algorithm will return $l \leq nt$.

**Question 4**
We start with some definitions. Player $i$ is *interested* in a piece $X$ if $v_i(X) \geq \frac{1}{3}$. He is *not interested* otherwise. He is *very interested* in $X$ if $v_i(X) \geq \frac{2}{3}$.

The algorithm is as follows. First, some player, without loss of generality player 1, cuts the cake into to equal pieces (from his point of view): $A$ and $B$. Every player is asked to declare if he is interested, very interested, or not interested at all. The algorithm now branches as follows:

- **There are 2 players that are interested in $A$, and the other two are interested in $B$:** In this case we run the equal-share protocol\(^1\) described in class on each side (with the corresponding two players). Each player is left with a piece that worth $\frac{1}{6}$ to him, as the equal-share protocol guarantees each player at least half of the value of the whole part, which was at least $\frac{1}{3}$.

- **There are 3 players that are very interested in the same piece (wlog, $A$):** In that case we use the following sub-protocol: player 1 gets $B$. Player 2 cuts $A$ into $A_1$ and $A_2$ both have the same value for him (so he is interested in both). Now players 3 and 4 declare if they are interested in $A_1$ or in $A_2$. Notice that they are interested in at least one of $A_1$ and $A_2$, since they are very interested in $A$. If both 2 and 3 are interested in the same piece, we give 2 the other piece, and use the equal-share protocol to divide between 2 and 3 the piece are interested in. Clearly all players get a piece that has a value of at least $\frac{1}{6}$ to them. Else, share using the equal-share protocol the piece that 2 and 3 are interested in, and give 3 the other one. Again, all players receive a piece that they value with at least $\frac{1}{6}$.

Simple case enumeration shows that the cases complement each other, as it is easy to see that given a partition of the cake into two pieces each player is either interested in both parts, or very interested in one part, and since the player 1 is interested in both parts.

---

\(^1\)One player cuts the part into two equal pieces, the other one chooses the more valuable piece.