Olympiad Combinatorics

Pranav A. Sriram

August 2014

Copyright notices

About the Author

Pranav Sriram graduated from high school at The International School Bangalore, India, and will be a Freshman at Stanford University this Fall.

2. ALGORITHMS – PART II

In this chapter we focus on some very important themes in the study of algorithms: *recursive algorithms, efficiency* and *information*. A recursive algorithm is one which performs a task involving n objects by breaking it into smaller parts. This is known as a "divide and conquer" strategy. Typically, we either do this by splitting the task with n objects into two tasks with $n/2$ objects or by first reducing the task to a task with (n-1) objects. The latter approach, which is essentially induction, is very often used to solve Olympiad problems.

Induction

We first look at two problems which use induction. In the first one, we use the technique of ignoring one object and applying the induction hypothesis on the remaining (n-1) objects. This obviously needs some care: we cannot completely ignore the nth object if it has some effect on the other objects!

Example 1 [China Girls Math Olympiad 2011-7]
There are n boxes B_1, B_2, …, B_n in a row. N balls are distributed amongst them (not necessarily equally). If there is at least one ball

in B_1, we can move one ball from B_1 to B_2. If there is at least 1 ball in B_n, we can move one ball from B_n to B_{n-1}. For $2 \le k \le (n-1)$, if there are at least two balls in B_k, we can remove two balls from B_k and place one in B_{k+1} and one in B_{k-1}. Show that whatever the initial distribution of balls, we can make each box have exactly one ball.

Answer:
We use induction and monovariants. The base cases $n = 1$ and 2 are trivial. Suppose we have an algorithm A_{n-1} for $n-1$ boxes; we construct an algorithm A_n for n boxes. We use two steps. The first step aims to get a ball into B_n and the second uses the induction hypothesis.

Step 1: If B_n contains at least one ball, move to step two. Otherwise, all n balls lie in the first $(n-1)$ boxes. Assign a weight 2^k to box B_k. Now keep moving balls from the boxes $B_1, B_2, \ldots, B_{n-1}$ as long as possible. This cannot go on indefinitely as the total weight of the balls is a positive integer and strictly increases in each move but is bounded above by $n2^n$. Thus at some point this operation terminates. This can only happen if B_1 has 0 balls and $B_2, B_3, \ldots, B_{n-1}$ each has at most 1 ball. But then B_n will have at least 2 balls. Now go to step 2.

Step 2: If B_n has $k > 1$ balls, move $(k-1)$ balls from B_n to B_{n-1}. Now B_n has exactly one ball and the remaining $(n-1)$ boxes have $(n-1)$ balls. Color these $(n-1)$ balls red and color the ball in B_n blue. Now we apply the induction hypothesis. Use algorithm A_{n-1} to make each of the first $(n-1)$ boxes have one ball each. The only time we run into trouble is when a move needs to be made from B_{n-1}, because in A_{n-1}, B_{n-1} only needed 1 ball to make a move, but now it needs 2. We can easily fix this. Whenever A_{n-1} says we need to move a ball from B_{n-1} to B_{n-2}, we first move the blue ball to B_{n-1}. Then we move a ball from B_{n-1} to B_{n-2} and pass the blue ball back to B_n. This completes the proof. ■

Example 2 [IMO Shortlist 2005, C1]
A house has an even number of lamps distributed among its rooms in such a way that there are at least three lamps in every room. Each lamp shares a switch with exactly one other lamp, not necessarily from the same room. Each change in the switch shared by two lamps changes their states simultaneously. Prove that for every initial state of the lamps there exists a sequence of changes in some of the switches at the end of which each room contains lamps which are on as well as lamps which are off.

Answer:
Call a room *bad* if all its lamps are in the same state and *good* otherwise. We want to make all rooms good. We show that if $k \geq 1$ rooms are bad, then we can make a finite sequence of switches so that (k-1) rooms are bad. This will prove our result.

Call two lamps connected if they share a switch. Take a bad room R_1 and switch a lamp there. If this lamp is connected to a lamp in R_1, we are done since each room has at least 3 lamps. If this lamp is connected to a lamp in another room R_2, then R_1 becomes good but R_2 might become bad. If R_2 doesn't become bad, we are done. If R_2 does become bad, then repeat the procedure so that R_2 becomes good but some other room R_3 becomes bad. Continue in this manner. If we ever succeed in making a room good without making any other room bad we are done, so assume this is not the case. Then eventually we will reach a room we have already visited before. We prove that at this stage, the final switch we made would not have made any room bad.

Consider the first time this happens and let $R_m = R_n$ for some $m > n$. We claim that R_m is good at this stage. The first time we switched a lamp in R_n, we converted it from bad to good by switching one lamp. Now when we go to R_m (= R_n), we cannot switch the same lamp, since this lamp was connected to a lamp in room R_{n-1}, whereas the lamp we are about to switch is connected to a lamp in R_{m-1}. So two distinct lamps have been switched in R_m and hence R_m is good (since there are at least three lamps, at least

one lamp hasn't been switched, and initially all lamps were in the same state since the room was bad before). Thus our final switch has made R_{m-1} good without making R_m bad. Hence we have reduced the number of bad rooms by one, and repeating this we eventually make all rooms good. ■

The next two examples demonstrate how to construct objects inductively.

Example 3:
Given a graph ***G*** in which each vertex has degree at least (n-1), and a tree ***T*** with n vertices, show that there is a subgraph of ***G*** isomorphic to ***T***.

Answer:
We find such a subgraph inductively. Assume the result holds for (n-1); we prove it holds for n. Delete a terminal vertex v from ***T***. By induction we can find a tree ***H*** isomorphic to ***T*** \ {v} as a subgraph of ***G***. This is because ***T*** \ {v} has (n-1) vertices and each vertex in ***G*** has degree at least (n-1) > (n-1) - 1, so we can apply the induction hypothesis. Now suppose v was adjacent to vertex u in ***T*** (remember that v is adjacent to only one vertex). Let w be the vertex in ***G*** corresponding to u. w has at least (n-1) neighbors in ***G***, and at most (n-2) of them are in ***H*** since ***H*** has (n-1) vertices and w is one of them. Thus w has at least 1 neighbor in ***G*** that is not in ***H***, and we take this vertex as the vertex corresponding to v. ■

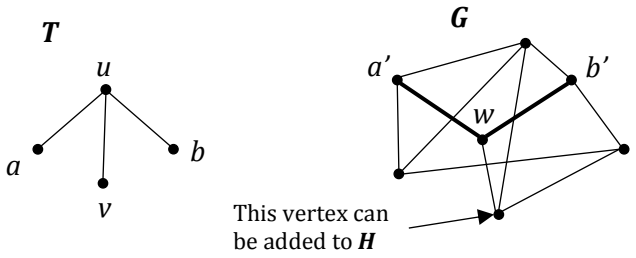

*Figure 2.1: Finding **H** inductively*

Example 4 [USAMO 2002]
Let $\boldsymbol{S}$ be a set with 2002 elements, and let N be an integer with $0 \le N \le 2^{2002}$. Prove that it is possible to color every subset of $\boldsymbol{S}$ black or white, such that:

a) The union of two white subsets is white
b) The union of two black subsets is black
c) There are exactly N white subsets.

Answer
You may have thought of inducting on N, but instead we induct on the number of elements of $\boldsymbol{S}$. In this problem $|\boldsymbol{S}| = 2002$, but we prove the more general result with $|\boldsymbol{S}| = n$ and $0 \le N \le 2^n$. The result trivially holds for $n = 1$, so suppose the result holds for $n = k$. Now we prove the result for $n = k+1$. If $N \le 2^{n-1}$, note that by induction there is a coloring for the same value of N and $n = k$. We use this coloring for all sets that do not contain the $(k+1)$th element of $\boldsymbol{S}$, and all subsets containing the $(k+1)$th element of $\boldsymbol{S}$ (which were not there in the case $|\boldsymbol{S}| = k$) are now colored black. (Essentially, all "new" subsets are colored black while the old ones maintain their original color). Clearly, this coloring works.

If $N \ge 2^{n-1}$, simply interchange the roles of white and black, and then use the same argument as in the previous case. ■

Information, Efficiency and Recursions

The next few problems primarily deal with collecting information and performing tasks efficiently, that is, with the minimum possible number of moves. Determining certain information with the least number of moves or questions is extremely important in computer science.

The next example is a simple and well-known problem in

computer science.

Example 5 [Merge Sort Algorithm]
Given n real numbers, we want to sort them (arrange them in non-decreasing order) using as few comparisons as possible (in one comparison we can take two numbers a and b and check whether $a < b$, $b < a$ or $a = b$). Clearly, we can sort them if we make all possible $n(n\text{-}1)/2$ comparisons. Can we do better?

Answer:
Yes. We use a recursive algorithm. Let $f(n)$ be the number of comparisons needed for a set of n numbers. Split the set of n numbers into 2 sets of size $n/2$ (or if n is odd, sizes $(n\text{-}1)/2$ and $(n+1)/2$. For the rest of this problem, suppose n is even for simplicity). Now sort these two sets of numbers individually. This requires $2f(n/2)$ comparisons. Suppose the resulting sorted lists are $a_1 \le a_2 \le \ldots \le a_{n/2}$ and $b_1 \le b_2 \le \ldots \le b_{n/2}$. Now we want to combine or 'merge' these two lists. First compare a_1 and b_1. Thus after a comparison between a_i and b_j, if $a_i \le b_j$, compare a_{i+1} and b_j and if $b_j < a_i$, compare b_{j+1} and a_i in the next round. This process terminates after at most n comparisons, after which we would have completely sorted the list. We used a total of at most $2f(n/2) + n$ comparisons, so $f(n) \le 2f(n/2) + n$.

From this recursion, we can show by induction that $f(2^k) \le k \times 2^k$ and in general, for n numbers the required number of comparisons is of the order $n\log_2(n)$, which is much more efficient than the trivial bound $n(n\text{-}1)/2$ which is of order n^2. ■

Example 6
Suppose we are given n lamps and n switches, but we don't know which lamp corresponds to which switch. In one operation, we can specify an arbitrary set of switches, and all of them will be switched from off to on simultaneously. We will then see which lamps come on (initially they are all off). For example, if $n = 10$ and we specify the set of switches {1, 2, 3} and lamps L_6, L_4 and L_9

come on, we know that switches {1, 2, 3} correspond to lamps L_6, L_4 and L_9 in *some order*. We want to determine which switch corresponds to which lamp. Obviously by switching only one switch per operation, we can achieve this in n operations. Can we do better?

Answer:
Yes. We actually need only $\lceil \log_2(n) \rceil$ operations, where $\lceil\ \rceil$ is the *ceiling* function. This is much better than n operations. For example, if n is one million, individually testing switches requires 999,999 operations, whereas our solution only requires 20. We give two solutions. For convenience assume n is even.

Solution 1:
In the first operation specify a set of $n/2$ switches. Now we have two sets of $n/2$ switches, and we know which $n/2$ lamps they both correspond to. Now we want to apply the algorithm for $n/2$ lamps and switches to the two sets. Hence it initially appears that we have the recursion $f(n) = 2f(n/2)+1$, where $f(n)$ is the number of steps taken by our algorithm for n lamps. However, note that we can actually apply the algorithms for both sets simultaneously, since we know which set of switches corresponds to which set of lamps. Thus the actual recursion is $f(n) = f(n/2)+1$. Since $f(1) = 0$, we inductively get $f(n) = \lceil \log_2(n) \rceil$.

Solution 2:
The algorithm in this solution is essentially equivalent to that in solution 1, but the thought process behind it is different. Label the switches 1, 2, …, n. Now read their labels in binary. Each label has at most $\lceil \log_2(n) \rceil$ digits. Now in operation 1, flip all switches that have a 1 in the units place of the binary representation of their labels. In general, in operation k we flip all switches that have a 1 in the kth position of their binary representation. At the end of $\lceil \log_2(n) \rceil$ operations, consider any lamp. Look at all the operations in which it came on. For example, if a lamp comes on in the second, third and fifth operations, but not in the first, fourth and

6^{th} operations, then it must correspond to the switch with binary representation 010110 (1s in the 2^{nd}, 3^{rd} and 5^{th} positions from the right). Thus each lamp can be uniquely matched to a switch and we are done. ■

Example 7 [Generalization of IMO shortlist 1998, C3]
Cards numbered 1 to n are arranged at random in a row with $n \geq 5$. In a move, one may choose any block of consecutive cards whose numbers are in ascending or descending order, and switch the block around. For example, if n=9, then 91 6 5 3 2 7 4 8 may be changed to 91 3 5 6 2 7 4 8. Prove that in at most $2n$ - 6 moves, one can arrange the n cards so that their numbers are in ascending or descending order.

Answer:
We use a recursive algorithm relating the situation with n cards to the situation with n-1 cards. Let $f(n)$ be the minimum number of moves required to 'monotonize' any permutation of the n cards. Suppose we have a permutation with starting card k. In $f(n\text{-}1)$ moves, we can monotonize the remaining (n-1) cards to get either the sequence (k, 1, 2, …, k-1, k+1, …, n) or (k, n, n-1, …, k+1, k–1, …, 2, 1). In one move, we can make the former sequence (k, k-1, k-2, …, 1, k+1, k+2, …, n) and with one more move we get the sequence (1, 2, 3, …., n) and we are done. Similarly in the latter case we need only two additional moves to get (n, n-1, …., 1). Thus in either case, we can complete the task using $f(n\text{-}1) + 2$ moves, so $f(n) \leq f(n\text{-}1) + 2$.

Now to prove the bound for general $n \geq 5$, it suffices to prove it for $n = 5$ and then induct using $f(n) \leq f(n\text{-}1) + 2$. To prove that $f(5) \leq 4$, first note that $f(3) = 1$ and $f(4) = 3$. With a little case work (do this), we can show that any permutation of 4 cards can be monotonized *either way* in at most 3 moves (thus both {1, 2, 3, 4} and {4, 3, 2, 1} can be reached after at most 3 moves, regardless of the initial permutation). Now given a permutation of {1, 2, 3, 4, 5}, use one move if necessary to ensure that either 1 or 5 is at an extreme position. Now monotonize the remaining 4 numbers in 3

moves, in such a way that the whole sequence is monotonized (we can do this by the previous statement). Hence at most 4 moves are required for 5 cards, and we are done. ■

Remark: Since we wanted a linear bound in this problem, we tried to relate $f(n)$ to $f(n\text{-}1)$. However, when we want a logarithmic bound, we generally relate $f(n)$ to $f(n/2)$, or use binary representations. Thus the question itself often gives us a hint as to what strategy we should use.

Example 8 [Russia 2000]
Tanya choses a natural number $X \leq 100$, and Sasha is trying to guess this number. She can select two natural numbers M and N less than 100 and ask for the value of gcd $(X+M, N)$. Show that Sasha can determine Tanya's number with at most seven questions (the numbers M and N can change each question).

Answer:
Since $2^6 < 100 < 2^7$ we guess that more generally $\lceil \log_2(n) \rceil$ guesses are needed, where n is the maximum possible value of X and $\lceil\ \rceil$ is the ceiling function.

Our strategy is to determine the digits of X in binary notation; that is, the bits of X. First ask for gcd $(X+2, 2)$. This will tell us whether X is even or odd, so we will know the units bit of X. If X is even, ask for gcd $(X+4, 4)$. This tells us whether or not X is divisible by 4. Otherwise ask for gcd $(X+1, 4)$. This tells us if X is 1 or 3 mod 4 (if the gcd is 4, then $X+1$ is divisible by 4 and so $X \equiv 3 \mod 4$). With this information we can determine the next bit of X. For example, if X is odd and is 3 mod 4, its last two bits will be 11. Now suppose $X = i \mod 4$. To determine the next digit, ask for gcd $(X + (4\text{-}i), 8)$. This gcd is either 4 or 8, according as $X = i$ or $4+i \mod 8$. This gives us the next bit. For example, if $X = 3 \mod 4$ but $X = 7 \mod 8$, then the last 3 bits of X will be 111, but if $X = 3 \mod 8$, then the last 3 bits would be 011. Now the pattern is clear. We continue in this manner until we obtain all the bits of X. This takes k

questions, where k is the number of bits of n (since $X \le n$, we don't have to ask for further bits), which is at most equal to $\lceil \log_2(n) \rceil$. ■

Example 9 [Generalization of Russia 2004, grade 9 problem 3]
On a table there are n boxes, where n is even and positive, and in each box there is one ball. Some of the balls are white and the number of white balls is even and greater than 0. In each turn we are allowed to point to two arbitrary boxes and ask whether there is at least one white ball in the two boxes (the answer is yes or no). Show that after $(2n - 3)$ questions we can indicate two boxes which definitely contain white balls.

Answer:
Label the boxes from 1 to n. Ask for the pairs of boxes $(1, j)$, where $j = 2, 3, \ldots, n$. If at some stage we get a no, this means box 1 contains a black ball. Then for all j such that we got a 'yes' for $(1, j)$, box j contains a white ball and we are done. The only other possibility is if we got a yes for all boxes $(1, j)$, in which case there are 2 possibilities: either box 1 has a white ball or box 1 has a black ball and all the other $(n-1)$ boxes have white balls. The latter case is ruled out since we are given that an even number or boxes have black balls, and $(n-1)$ is odd. Hence box 1 has a white ball. Now ask for the pairs $(2, j)$ where $j = 3, 4, \ldots, n$. Note that now we have asked a total of $(n-1) + (n-2) = (2n-3)$ questions. Again if we get a 'no' somewhere, then box 2 has a black ball and all yeses tell us which boxes have white balls. In this case we are done. The other case is if all the answers are yes. The same argument we used earlier shows that box 2 has a white ball and we are done. ■

Now we look at a simple problem from computer science (part *a* of the next problem), which also happens to be a game I played when I was a little kid. Part *b* is a little trick question I just came up with.

Example 10 a) [Binary Search Algorithm]

My friend thinks of a natural number X between 1 and 2^k inclusive. In one move I can ask the following question: I specify a number n and he says bigger, smaller or correct according as $n < X$, $n > X$, or $n = X$. Show that I can determine the number X using at most k moves.

Answer:
In my first move I say 2^{k-1}. Either I win, or I have reduced number of possibilities to 2^{k-1}. Repeat this process- in each stage reduce the number of possibilities by a factor of 2. Then in k moves there is only one possibility left.

Example: If $k = 6$, first guess 32. If the answer is "smaller", guess 16. If the answer is now "bigger", guess 24 (the average of 16 and 32). If the answer is now "smaller", guess 20. If the answer is again smaller, guess 18. If the answer is now "bigger", the number is 19.

In general if we replace 2^k with n, we need $\lceil \log_2(n) \rceil$ questions. ■

Example 10 b) [Binary Search with noise – a little trick]
We play the same game, but with a couple changes. First, I can now ask **any** yes or no question. Second, now my friend is allowed to lie - at most once in the whole game though. Now, from part a) I can win with $2k + 1$ moves: I simply ask each question twice, and if the answer changes, that means my friend has lied. Then I ask the question again, and this time I get the true answer (he can only lie once). Thus I ask each question twice except for possibly one question which I ask thrice, for a total of $2k + 1$ questions. Can I do better?

Answer:
First I ask about each of the k digits in the binary representation of X. If the game didn't involve lying, I would be done. Now I need to account for the possibility that one answer was a lie. I ask the question, "did you ever lie this game?" If the answer is no, we are done (if he had lied, he would have to say yes now as he can't lie

twice). If the answer is yes, I ask the question, "was the previous answer a lie?" If the answer to this is yes, then that means he never lied in the first k questions and again we are done. If the answer is no, then we can be sure that one of the first k answers we received (about the binary digits) was a lie. Note that he cannot lie anymore. We want to determine which answer was a lie. But using part a), we can do this in at most $\lceil \log_2(k) \rceil$ moves! This is because determining which of k moves was a lie is equivalent to guessing a number X' with $X' \leq k$, and for this I use the algorithm in part a). After this, I know which digit in my original binary representation of X is wrong and I change it, and now I am done. I have used $k + 2 + \lceil \log_2(k) \rceil$ questions, which is much less than $2k + 1$ questions for large k.

In general, if 2^k is replaced by n, this algorithm takes $\lceil \log_2(n) \rceil + \left\lceil \log_2 \lceil \log_2(n) \rceil \right\rceil + 2$ moves. ■

As in the previous chapter, we end this section with one of the hardest questions ever asked at the IMO. Only 3 out of over 550 contestants managed to completely solve it. However, the difficulty of this now famous problem has been hotly debated on AOPS, with many people arguing that it is a lot easier that the statistics indicate. We'll let the reader be the judge of that. The following solution is based on one found during the contest. The official solution is much more complicated.

Example 11 [IMO 2009, Problem 6]
Let n be a nonnegative integer. A grasshopper jumps along the real axis. He starts at point 0 and makes $n + 1$ jumps to the right with pairwise different positive integral lengths $a_1, a_2, \ldots a_{n+1}$ in an arbitrary order. Let $\boldsymbol{M}$ be a set of n positive integers in the interval $(0, s)$, where $s = a_1 + a_2 + \cdots + a_{n+1}$. Prove that the grasshopper can arrange his jumps in such a way that he never lands on a point from $\boldsymbol{M}$.

Answer:
We construct an algorithm using induction and the extremal principle. The case $n = 1$ is trivial, so let us assume that $n > 1$ and that the statement holds for 1, 2, ..., $n-1$. Assume that $a_1 < \cdots < a_n$. Let m be the smallest element of $\boldsymbol{M}$. Consider the following cases:

Case 1: $m < a_{n+1}$: If a_{n+1} does not belong to $\boldsymbol{M}$ then make the first jump of size a_{n+1}. The problem gets reduced to the sequence a_1, ..., a_n and the set $\boldsymbol{M}\backslash\{m\}$, which immediately follows by induction. So now suppose that $a_{n+1} \in \boldsymbol{M}$. Consider the following n pairs: (a_1, a_1+a_{n+1}), ..., (a_n, a_n+a_{n+1}). All numbers from these pairs that are in $\boldsymbol{M}$ belong to the $(n-1)$-element set $\boldsymbol{M}\backslash\{a_n\}$, hence at least one of these pairs, say (a_k, a_k+a_n), has both of its members outside of $\boldsymbol{M}$. If the first two jumps of the grasshopper are a_k and a_k+a_{n+1}, it has jumped over at least two members of $\boldsymbol{M}$: m and a_{n+1}. There are at most $n-2$ more elements of $\boldsymbol{M}$ to jump over, and $n-1$ more jumps, so we are done by induction.

Case 2: $m \geq a_{n+1}$: Note that it is equivalent to solve the problem in reverse: start from $s = a_1 + a_2 + \cdots + a_{n+1}$ and try to reach 0 without landing on any point in $\boldsymbol{M}$. By the induction hypothesis, the grasshopper can start from s make n jumps of sizes a_1, ..., a_n to the left, and avoid all the points of $\boldsymbol{M}\backslash\{m\}$. If it misses the point m as well, then we are done, since we can now make a jump of size a_{n+1} and reach 0. So suppose that after making the jump a_k the grasshopper landed at m. If it changes the jump a_k to the jump a_n, it will jump past m and all subsequent jumps will land outside of $\boldsymbol{M}$ because m is the left-most point. ■

Exercises

1. **[Spain 1997]**
   The exact quantity of gas needed for a car to complete a single loop around a track is distributed among n containers placed along the track. Show that there exists a point from which the car can start with an empty tank and complete the loop (by collecting gas from tanks it encounters along the way). [Note: assume that there is no limit to the amount of gas the car can carry].

2. **[Russia]**
   Arutyun and Amayak perform a magic trick as follows. A spectator writes down on a board a sequence of N (decimal) digits. Amayak covers two adjacent digits by a black disc. Then Arutyun comes and says both closed digits (and their order). For which minimal N can this trick always work?

3. **[Generalization of Russia 2005]**
   Consider a game in which one person thinks of a permutation of {1, 2, …, n} and the other's task is to deduce this permutation (n is known to the guesser). In a turn, he is allowed to select three positions of the permutation and is told the relative order of the three numbers in those positions. For example, if the permutation is 2, 4, 3, 5, 1 and the guesser selects positions 1, 4 and 5, the other player will reveal that 5^{th} number < 1^{st} number < 4^{th} number. Determine the minimum number of moves for the guesser to always be able to figure out the permutation.

4. **[IMO Shortlist 1990]**
   Given n countries with three representatives each, m committees $A_1, A_2, \ldots, A_m$ are called a *cycle* if

i. each committee has n members, one from each country;
ii. no two committees have the same membership;
iii. for $1 \le i \le m$, committee A_i and committee A_{i+1} have no member in common, where A_{m+1} denotes A_1
iv. if $1 < |i-j| < m-1$, then committees A_i and A_j have at least one member in common.

Is it possible to have a cycle of 1990 committees with 11 countries?

5. [Canada 2012 – 4]

A number of robots are placed on the squares of a finite, rectangular grid of squares. A square can hold any number of robots. Every edge of each square of the grid is classified as either passable or impassable. All edges on the boundary of the grid are impassable. A move consists of giving one of the commands up, down, left or right. All of the robots then simultaneously try to move in the specified direction. If the edge adjacent to a robot in that direction is passable, the robot moves across the edge and into the next square. Otherwise, the robot remains on its current square.

Suppose that for any individual robot, and any square on the grid, there is a finite sequence of commands that will move that robot to that square. Prove that you can also give a finite sequence of commands such that all of the robots end up on the same square at the same time.

6. [IMO Shortlist 2002, C4]

Let $\boldsymbol{T}$ be the set of ordered triples (x, y, z), where x, y, z are integers with $0 \le x, y, z \le 9$. Players A and B play the following guessing game. Player A chooses a triple in $\boldsymbol{T}$ (x, y, z), and Player B has to discover A's triple in as few moves as possible. A *move* consists of the following: B gives A a triple (a, b, c) in $\boldsymbol{T}$, and A replies by giving B the number $|x + y - a - b| + |y + z - b$

$-c| + |z + x - c - a|$. Find the minimum number of moves that B needs in order to determine A's triple.

7. **Given a finite set of points in the plane,**
each with integer coordinates, is it always possible to color the points red or white so that for any straight line L parallel to one of the coordinate axes the difference (in absolute value) between the numbers of white and red points on L is not greater than 1?

8. **[Generalization of Russia 1993]**
There are n people sitting in a circle, of which some are truthful and others are liars (we don't know who is a liar and who isn't though). Each person states whether the person to in front of him is a liar or not. The truthful people always tell the truth, whereas the liars may either lie or tell the truth. The aim is for us to use the information provided to find one person who is definitely truthful. Show that if the number of liars is at most $2\sqrt{n} - 3$, we can always do this.

9. On each square of a chessboard is a light which has two states- on or off. A move consists of choosing a square and changing the state of the bulbs in that square and in its neighboring squares (squares that share a side with it). Show that starting from any configuration we can make finitely many moves to reach a point where all the bulbs are switched off

10. **[Indian Postal Coaching 2011]**
Let C be a circle, $A_1, A_2, \ldots, A_n$ be distinct points inside C and $B_1, B_2, \ldots, B_n$ be distinct points on C such that no two of the segments $A_1B_1, \ldots, A_nB_n$ intersect. A grasshopper can jump from A_r to A_s if the line segment A_rA_s does not intersect any line segment A_tB_t ($t \neq r, s$). Prove that after a certain number of jumps, the grasshopper can jump from any A_u to any A_v.

11. [USAMO 2000, Problem 3]

You are given R red cards, B blue cards and W white cards and asked to arrange them in a row from left to right. Once arranged, each card receives a score as follows. Each blue card receives a score equal to the number of white cards to its right. Each white card receives a score equal to twice the number of red cards to its right. Each red card receives a score equal to three times the number of blue cards to its right. For example, if the arrangement is Red Blue White Blue Red Blue, the total score will be 9 + 1 + 2 + 0 + 3 + 0 = 15. Determine, as a function of R, B and W, the minimum possible score that can obtained, and find all configurations that achieve this minimum.

12. [IMO Shortlist 2005, C7]

Suppose that $a_1, a_2, \ldots, a_n$ are integers such that $n \mid (a_1 + a_2 + \ldots + a_n)$. Prove that there exist two permutations $b_1, b_2, \ldots, b_n$ and $c_1, c_2, \ldots, c_n$ of $(1, 2, \ldots, n)$ such that for each integer i with $1 \le i \le n$, we have $n \mid a_i - b_i - c_i$.

13. [St. Petersburg 2001]

In the parliament of the country of Alternativia, for any two deputies there exists a third who is acquainted with exactly one of the two. There are two parties, and each deputy belongs to exactly one of them. Each day the President (not a member of the parliament) selects a group of deputies and orders them to switch parties, at which time each deputy acquainted with at least one member of the group also switches parties. Show that the President can eventually ensure that all deputies belong to the same party.