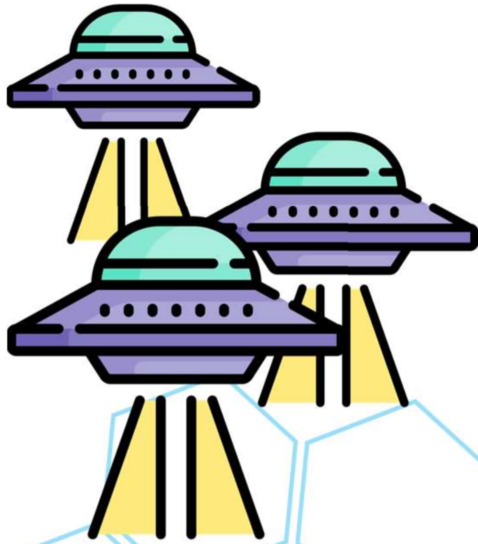
We will begin at 1:05

Read this problem while we wait…

Aliens abducted 100 mathematicians and put them into 100 separate rooms. Each room has a surveillance camera and each mathematician can see the 99 other mathematician's rooms except their own. Each room is painted in red or in blue, but the colour of the paint can only be seen in camera, not by naked eye. Then, the alien overlord makes an public announcement that *at least one of the rooms is painted blue, and that whoever that can figure out (with proof) the colour of their own room will be sent back to Earth!* Starting from that day, the alien overlord will privately talk to each mathematician once per day, asking for the proof (so they cannot just guess the colour). Suppose that every mathematician is perfectly smart i.e. they will know if such proof exists and will try to give it as soon as possible. Show that all mathematicians will eventually return to Earth.



Record the meeting…

Some Housekeeping

- It is the case that I put the due date wrong. So, Problem Set 1 diamond-problems will be due tomorrow (midnight).

- Since Ko Naing Zaw Lu, Ko Phyoe Min Khant, Ko Kyaw Shin Thant and me will all use the same google classroom, it makes sense to change the name and recategorize everything. And I did exactly that.

- As you can see in the outline, lecture 6 is "counting in two ways". If you don't know basic counting (permutations, combinations, etc.) please study them before next Wednesday.

Content so far...

L1: Monovariants
L2: Invariants
L3: Alternating-variants
I

L4: Inductive constructions
L5: Greedy and RUST
II

L6: Counting in two ways
L7: Inequalities and bounding
L8: Counting in graphs
L9: Injections and bijections
III

L10: Pigeonhole principle
L11: Continuity and descent
L12: Leveraging symmetry
IV

L13: Combinatorial games
V

L14: Combinatorial geometry
VI

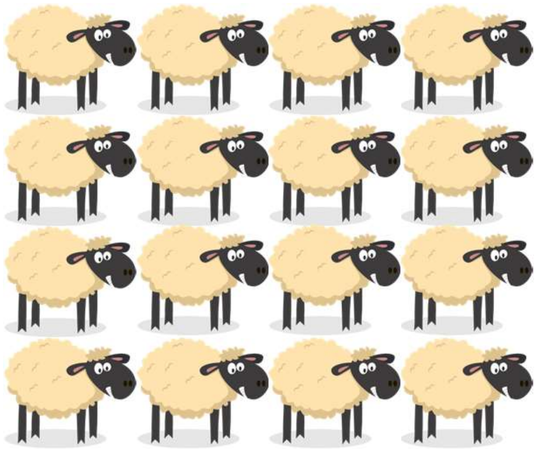
L15: Results in graph theory I
L16: Results in graph theory II
VII

Lecture – 4

Inductive Constructions

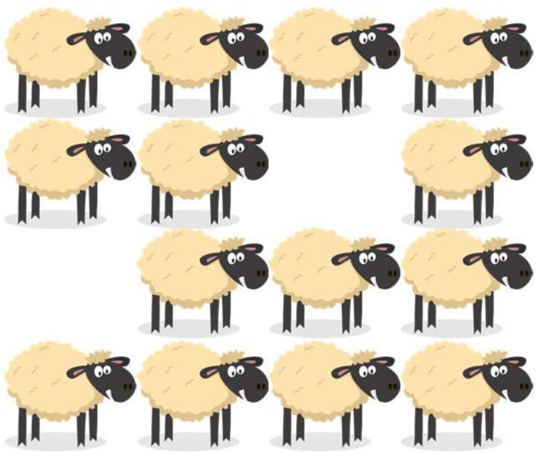
Comparison Game

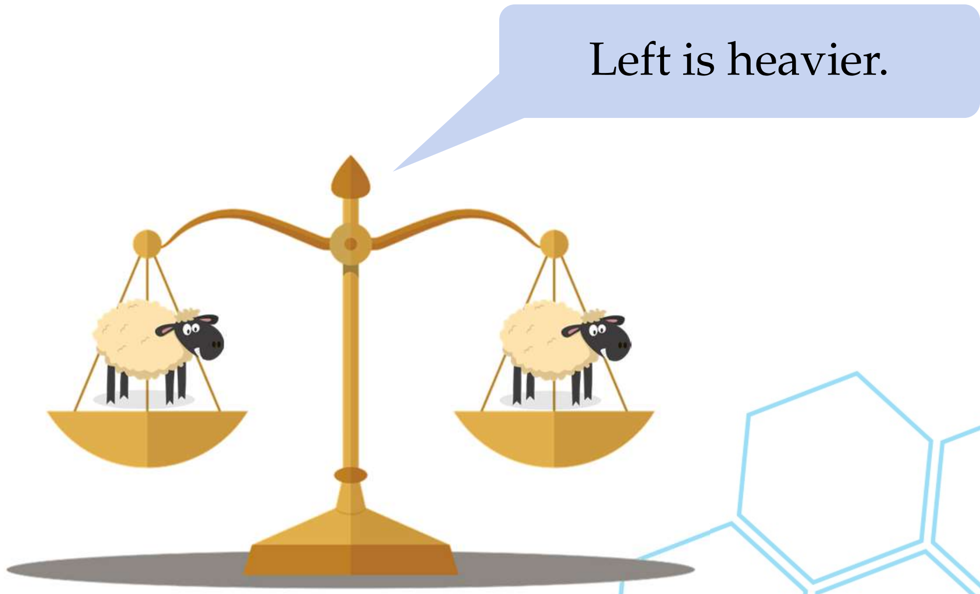
There are sixteen sheep with distinct weights, and you have a sheep-scale. You can put two sheep on the sheep-scale at a time, and the sheep-scale tells you which sheep is heavier. We would like to arrange the sheep in an increasing order by weights. Can we do this by using the sheep-scale less than 50 times?



Comparison Game

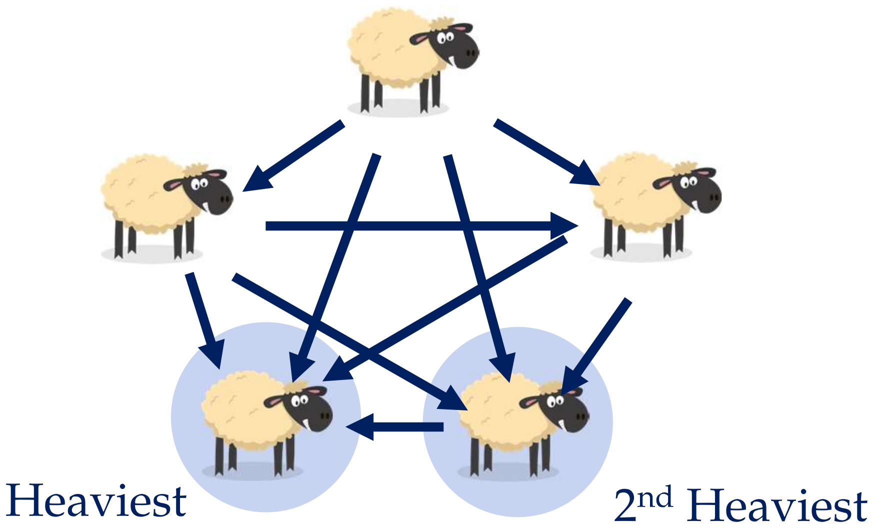
There are sixteen sheep with distinct weights, and you have a sheep-scale. You can put two sheep on the sheep-scale at a time, and the sheep-scale tells you which sheep is heavier. We would like to arrange the sheep in an increasing order by weights. Can we do this by using the sheep-scale less than 50 times?





Let's try the most obvious way…

The most obvious way is to compare any two of the sheep… and then find the heaviest sheep, then the second heaviest sheep, etc.



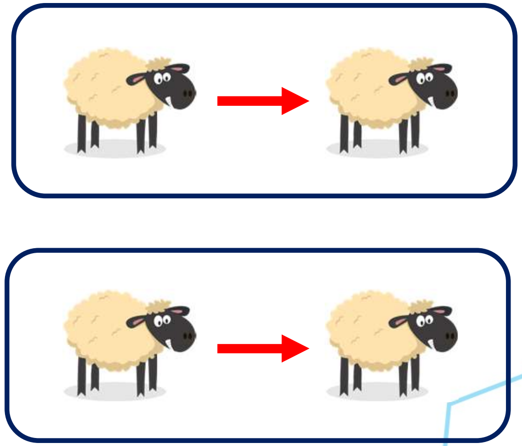
But then, we would have to use the sheep-scale $\binom{16}{2} = 120$ times!

Building up

If we have 2 sheep, we can figure out with 1 use.

If we have 4 sheep, how many do we need?

- Break the sheep into 2 herds of equal size.
- We know how to arrange each herd…
- Do we know how to 'merge' the herds?



Merging sheep

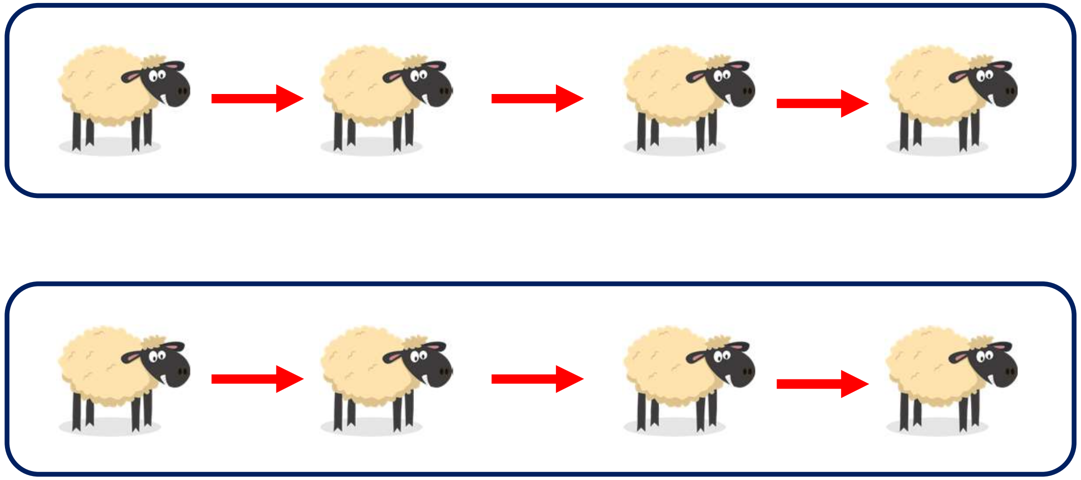
- Now, with one comparison, we can figure out the lightest sheep!
- Now, with another comparison, we can figure out the 2^{nd} lightest sheep.
- With another comparison, we can figure out the 3^{rd} lightest sheep.

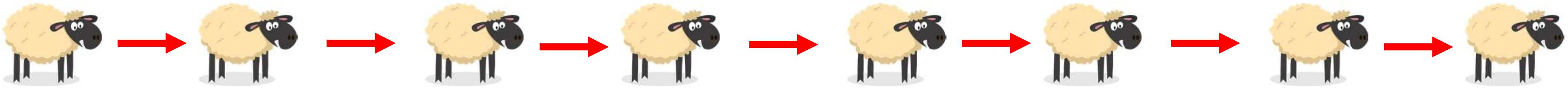
What about 8 sheep?

- Break the sheep into 2 herds of equal size.
- We know how to arrange each herd.
- Now, we can know the lightest sheep by using 1 comparison.
- So, we can know the 2nd lightest sheep by using 1 comparison.
- …
- So, we can know the 7th lightest sheep by using 1 comparison.

$5 \times 2 = 10$ comparisons

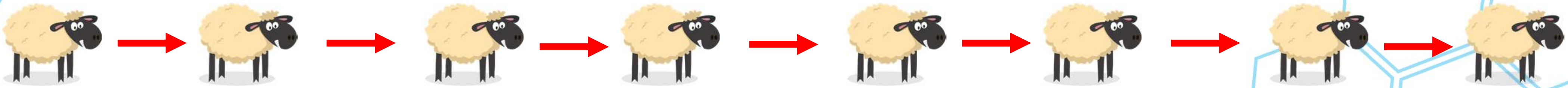
7 comparisons





Thus, for 16 sheep,

we only need to compare at most $17 \times 2 + 15 = 49$ times!



For 2^n sheep?

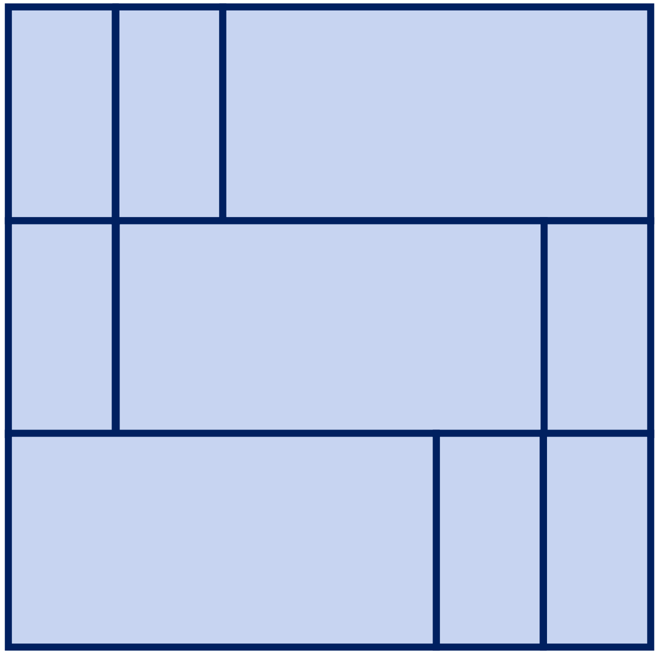
Let a_n be the minimum number of comparisons we need to arrange 2^n sheep.

Then, $a_{n+1} \leq 2a_n + 2^{n+1} - 1$ for all $n \geq 1$ and $a_1 = 1$.

We just need to solve this recursion. $\longleftarrow$ Easy Job

An Oxford Interview Problem

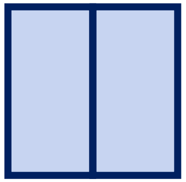
Call a rectangle **silver** if it is similar to a 2×1 rectangle. For which integers $n \geq 2$ is it possible to tile a square with n silver rectangles which are not necessarily congruent to each other?



Let's Play Around…

Let's play around with some small values of n.

Can you make a square if $n = 2$?

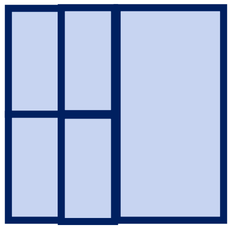


Can you make a square if $n = 3$?

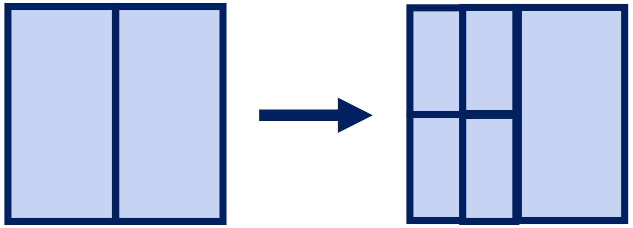
Can you make a square if $n = 4$?

Seems no…

Can you make a square if $n = 5$?



Main Observation…



You can generate more configurations by subdividing a silver rectangle into 4 silver rectangles.

If n silver rectangles make a square, then $n+3$ silver rectangles also make a square.

This observation causes chain reaction.

Possible with 2 rectangles → Possible with 5 rectangles → Possible with 8 rectangles → … → Possible with $3n+2$ rectangles →

Constructions for small n

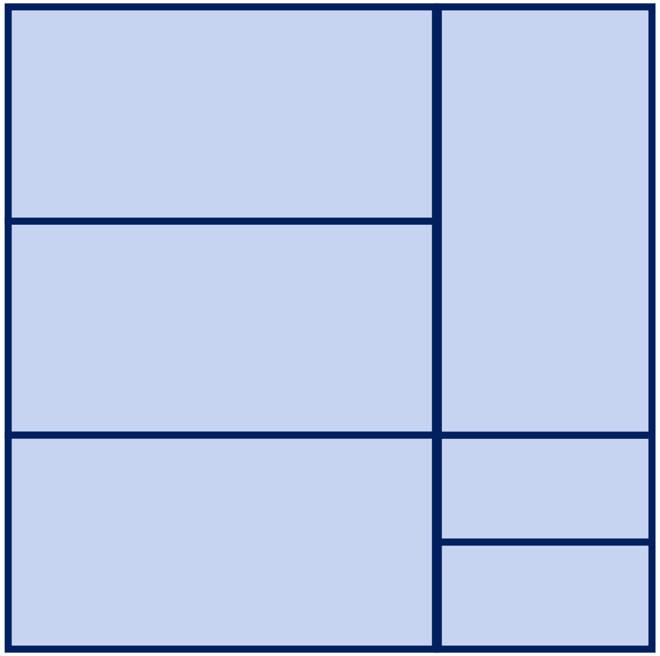
Possible with 2 rectangles → Possible with 5 rectangles → Possible with 8 rectangles → … → Possible with $3n + 2$ rectangles →

~~Possible with 3 rectangles~~ → Possible with 6 rectangles → Possible with 9 rectangles → … → Possible with $3n$ rectangles →

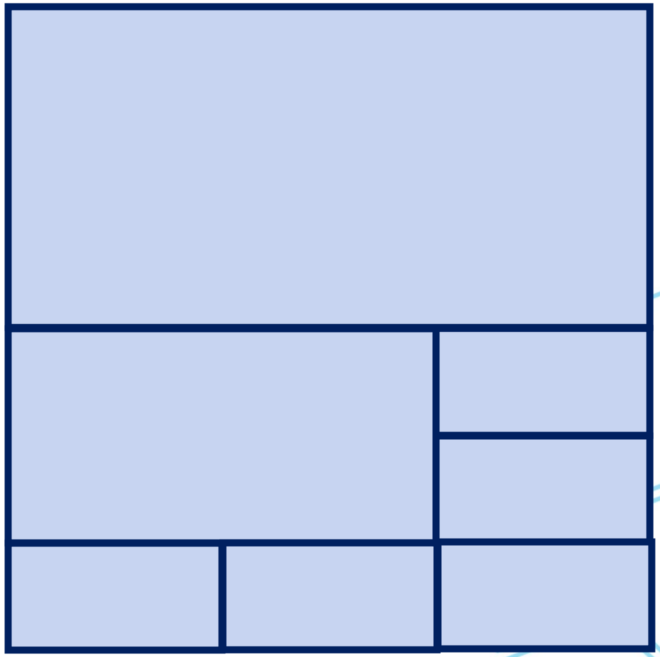
~~Possible with 4 rectangles~~ → Possible with 7 rectangles → Possible with 10 rectangles → … → Possible with $3n + 1$ rectangles →

Constructions for small n

Possible with 6 rectangles

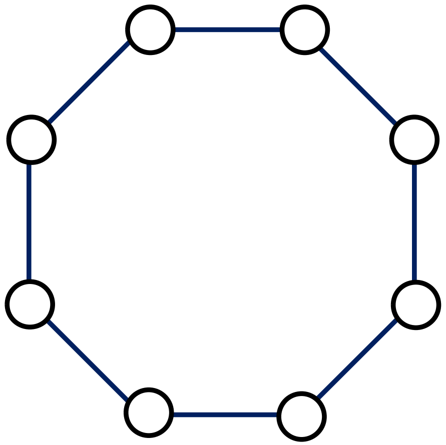


Possible with 7 rectangles



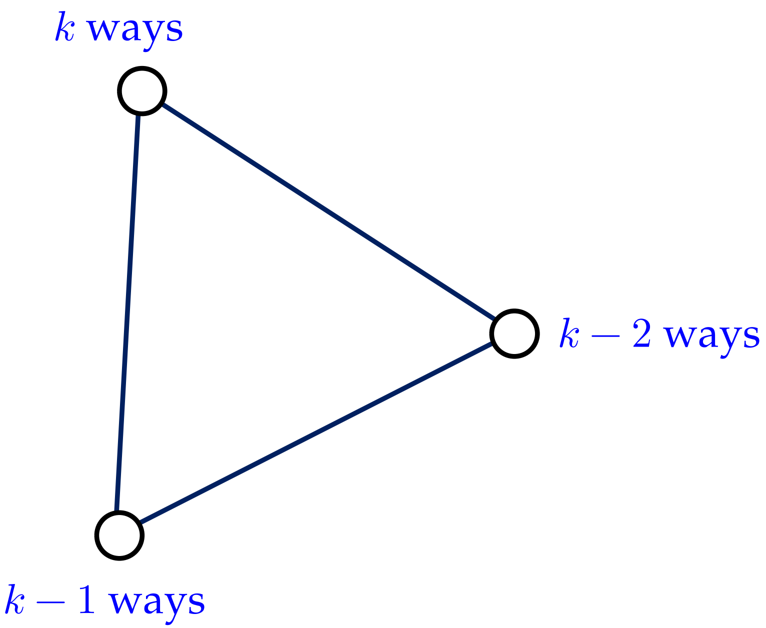
Colouring Made Easy...

How many ways are there to colour the vertices of a regular n-gon using at most k colours so that the vertices on every side are of different colours?



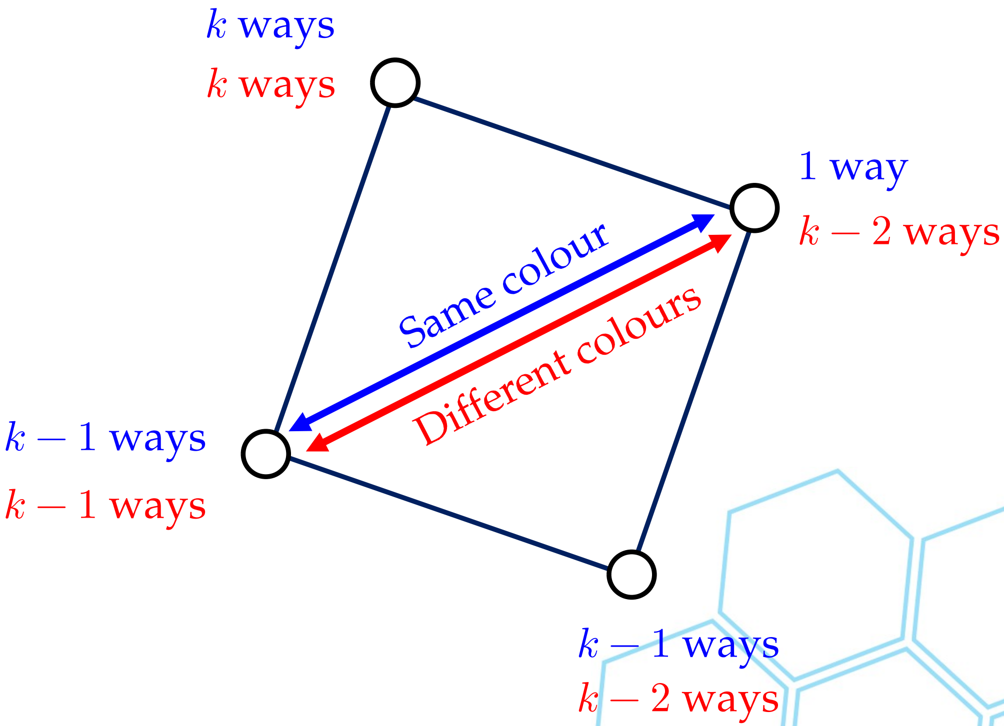
Small Examples

$\boxed{n = 3}$



Answer: $k(k-1)(k-2)$

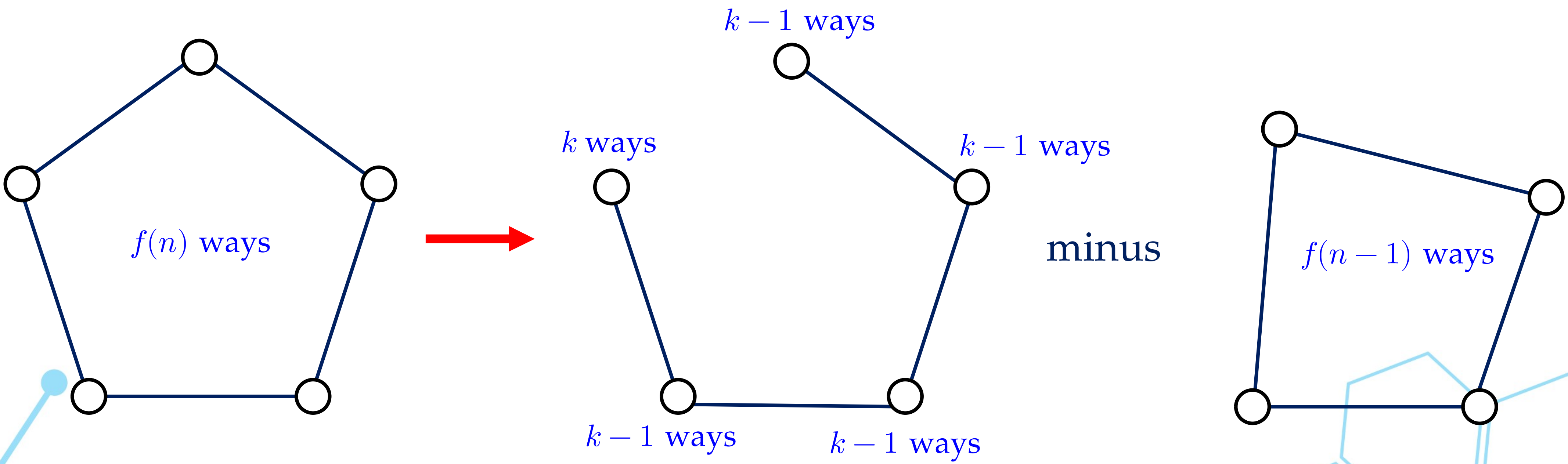
$\boxed{n = 4}$



Answer: $k(k-1)^2 + k(k-1)(k-2)^2$

Reduce to a smaller problem

Let $f(n)$ be the number of ways to colour a n-cycle with at most k colours.



$$f(n) = k(k-1)^{n-1} - f(n-1)$$

Solve the recurrence

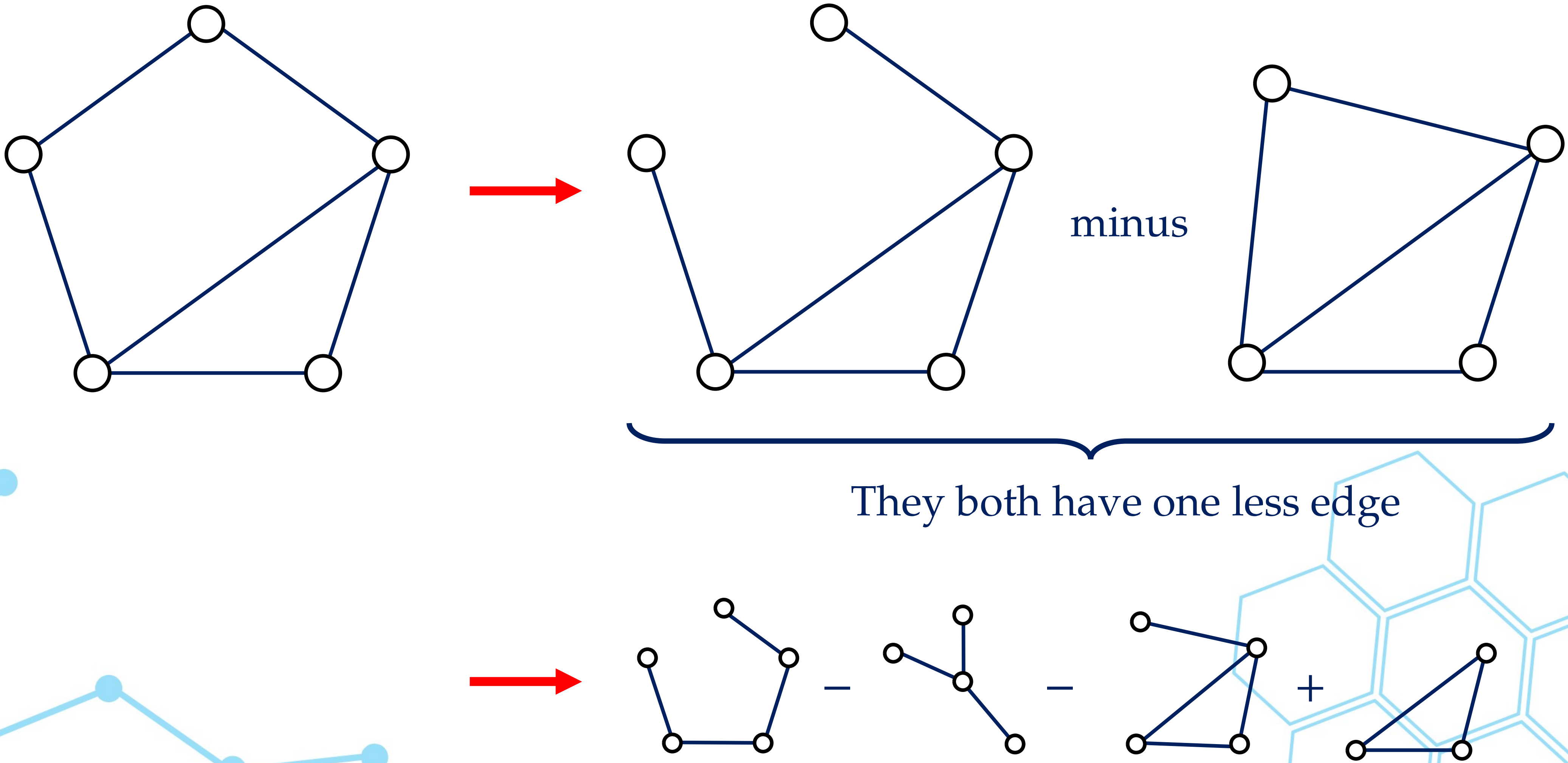
Now, backward substitute and simplify to get $f(n)$. This looks messy, but straightforward.

$$
\begin{aligned}
f(n) &= k(k-1)^{n-1} - f(n-1) \\
&= k(k-1)^{n-1} - k(k-1)^{n-2} + f(n-2) \\
&= k(k-1)^{n-1} - k(k-1)^{n-2} + k(k-1)^{n-3} - f(n-3) \\
&= \cdots \\
&= k(k-1)^{n-1} - k(k-1)^{n-2} + k(k-1)^{n-1} - \cdots + (-1)^{n-1}k(k-1)^2 + \overbrace{(-1)^n k(k-1)}^{f(2)} \\
&= k\left[(k-1)^{n-1} - (k-1)^{n-2} + \cdots + (-1)^n(k-1)\right] \\
&= k(k-1)\frac{(-1)^{n-1}(k-1)^{n-1} - 1}{-(k-1) - 1}
\end{aligned}
$$

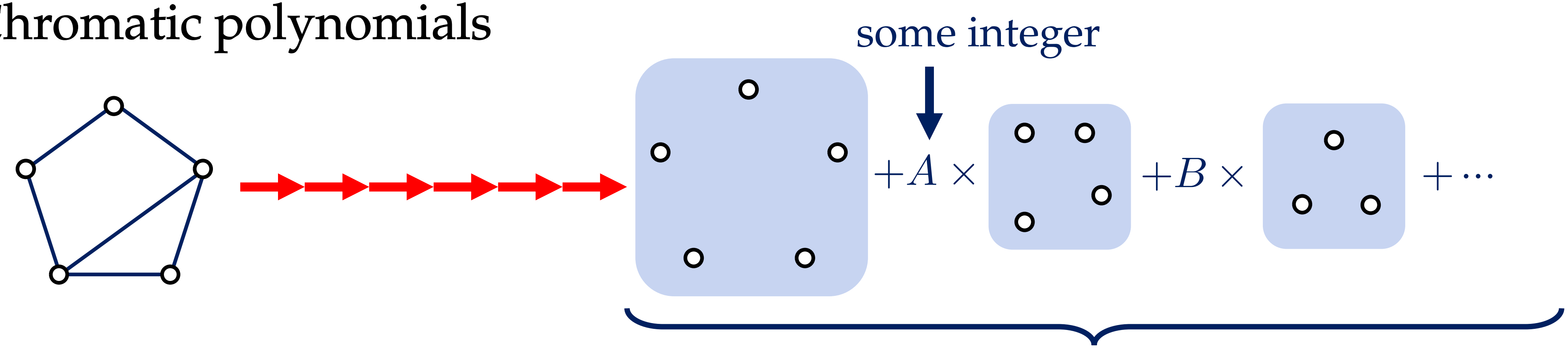
Geometric series

$$
\boxed{f(n) = (k-1)^n + (-1)^n(k-1)}
$$

For general graphs



Chromatic polynomials



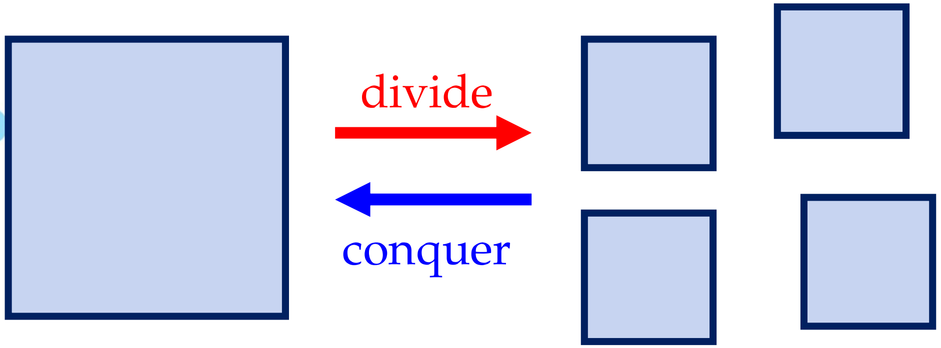
number of colourings $= k^5 + Ak^4 + Bk^3 + Ck^2 + Dk$

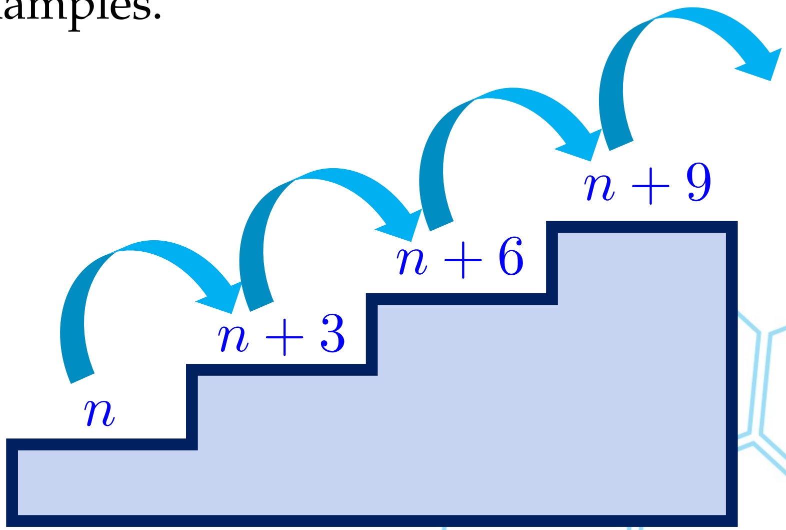
So, for any graph G, number of ways to colour the vertices with at most k colours so that no two vertices of the same colour are adjacent is equal to some polynomial of k. This polynomial is called the chromatic polynomial of G.

Question: Given a polynomial P, how can we know if it is the chromatic polynomial of some graph? ← No one knows the answer

Uses of Induction

- Divide and Conquer: To break the problem into smaller (but similar) pieces, then combine them.
- Recurrences: To setup recurrence relations to count/bound something.
- Staircase (Chain reaction): To construct a sequence of examples.



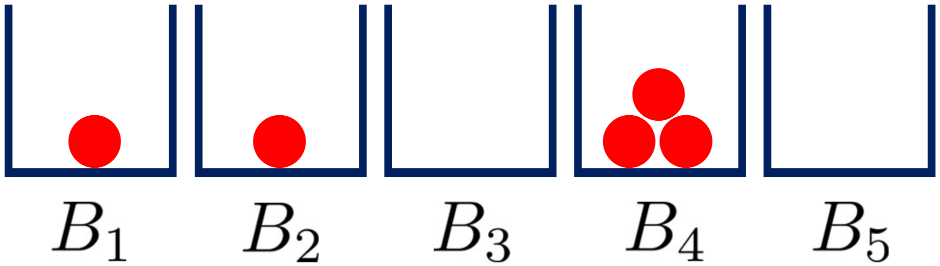


Distributing Marbles (China Girls MO, Problem 7)

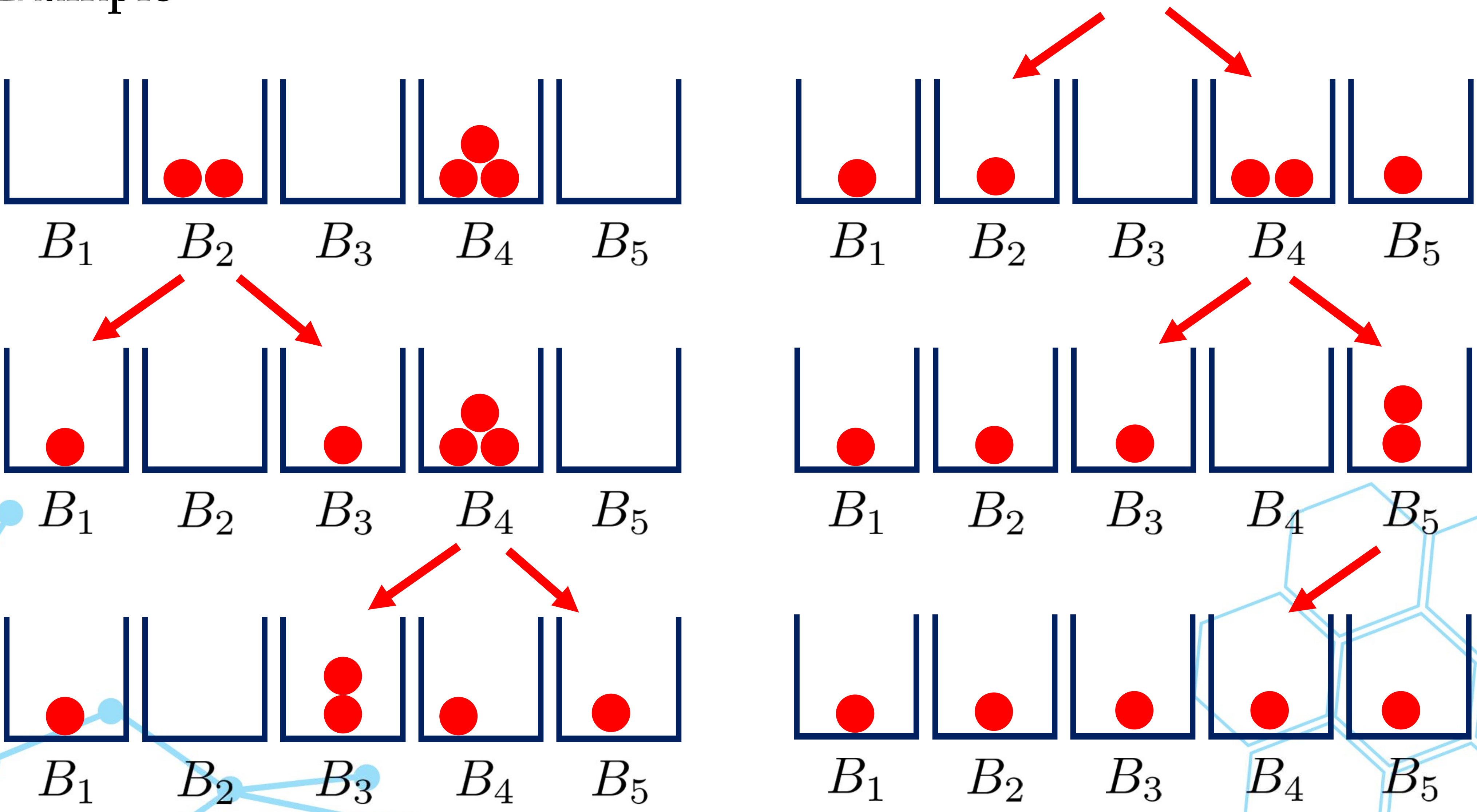
Let n be a positive integer. n marbles are distributed among n boxes $B_1, B_2, \ldots, B_n$ arranged in a row. We are allowed to make following moves:

- For $1 < k < n$, if B_k has at least 2 marbles in it, we may remove 2 marbles from B_k and put one in each of B_{k-1} and B_{k+1}.
- If B_1 has a marble in it, we may move it to B_2.
- If B_n has a marble in it, we may move it to B_{n-1}.

Show that it is possible to make each box have one marble each.



Example

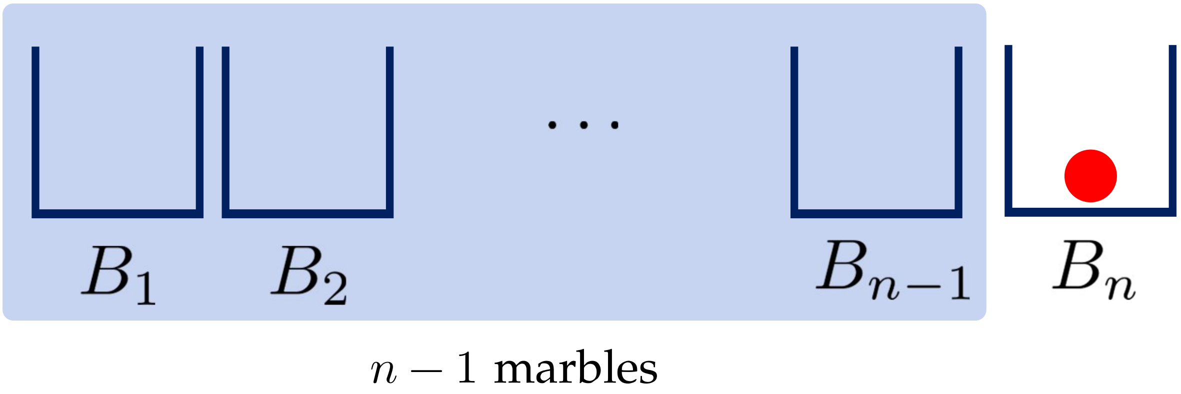


B_1 B_2 B_3 B_4 B_5
B_1 B_2 B_3 B_4 B_5
B_1 B_2 B_3 B_4 B_5
B_1 B_2 B_3 B_4 B_5
B_1 B_2 B_3 B_4 B_5
B_1 B_2 B_3 B_4 B_5

Let's Attempt to Induct…

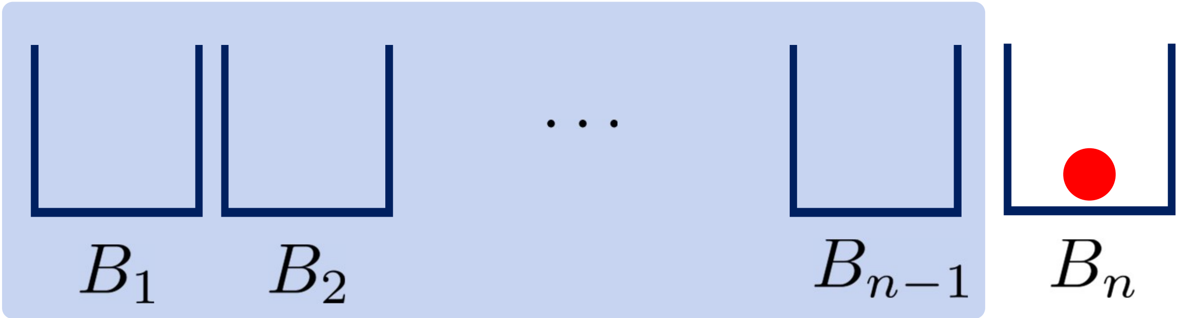
Suppose that we know how to do this if we have $n-1$ marbles and $n-1$ boxes.

So… the following is the ideal situation to apply induction:



Let's just say we are in this situation. Does induction kill the problem?

Let's Attempt to Induct…



$n-1$ marbles

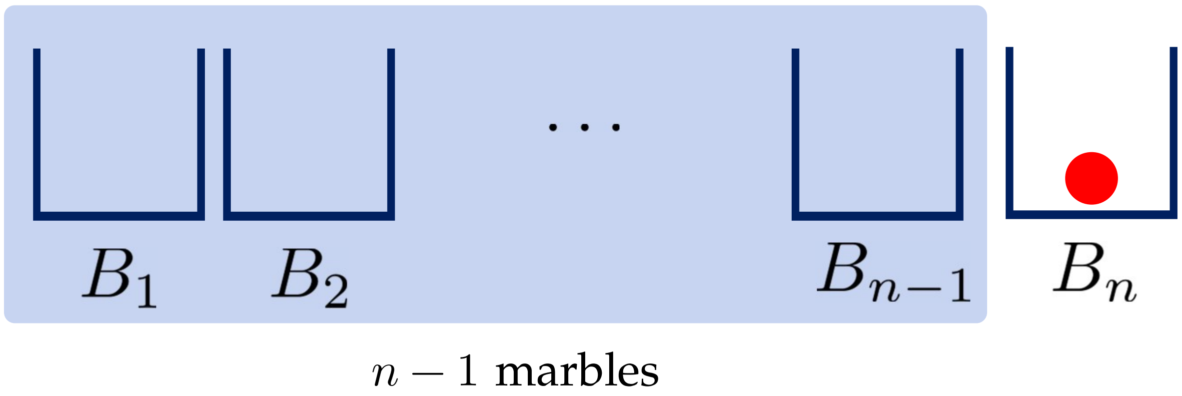
- If B_{n-1} is the last box, we know how to make each of $B_1, B_2, \ldots, B_{n-1}$ have one marble, let's say via a sequence S of moves on the boxes.

- The problem happens when B_{n-1} appears in S because in the situation for n boxes, we need to have 2 marbles in B_{n-1} whereas we only need to have 1 in B_{n-1} for the inductive situation.

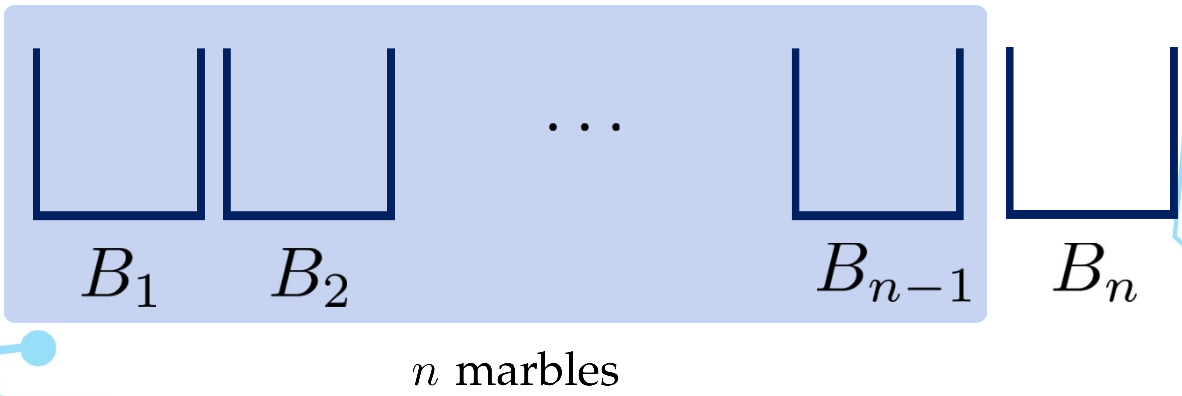
- Easy, whenever we need to use B_{n-1} , move marble from B_n into B_{n-1} first, then we continue as in the inductive algorithm.

When can we induct?

So, we are done once B_n is non-empty. Because then, we can make the following situation:



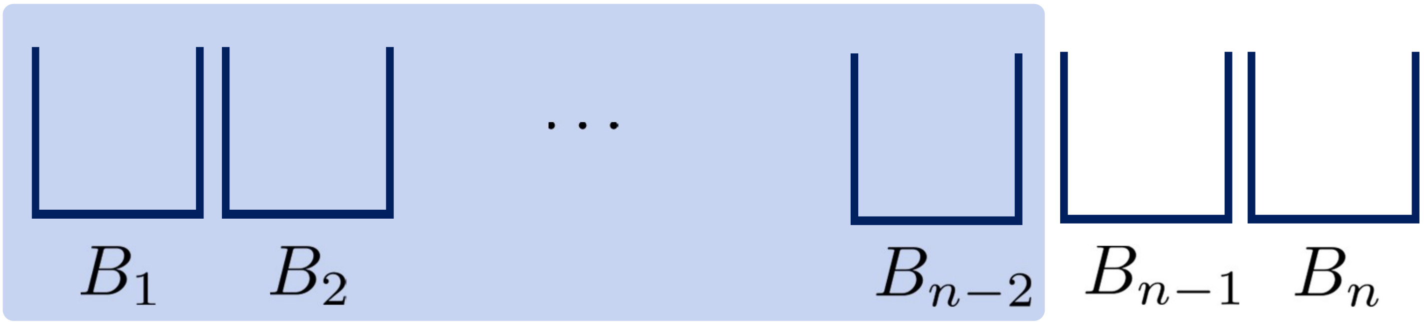
How do we do this? Note: if there are no marbles in B_n, then we have the following situation:



How to send a marble to B_n ?

Observation: If B_n is empty, we can always make a move, so we never get stuck.

Goal: Get 2 marbles into B_{n-1} .



n or $n-1$ marbles

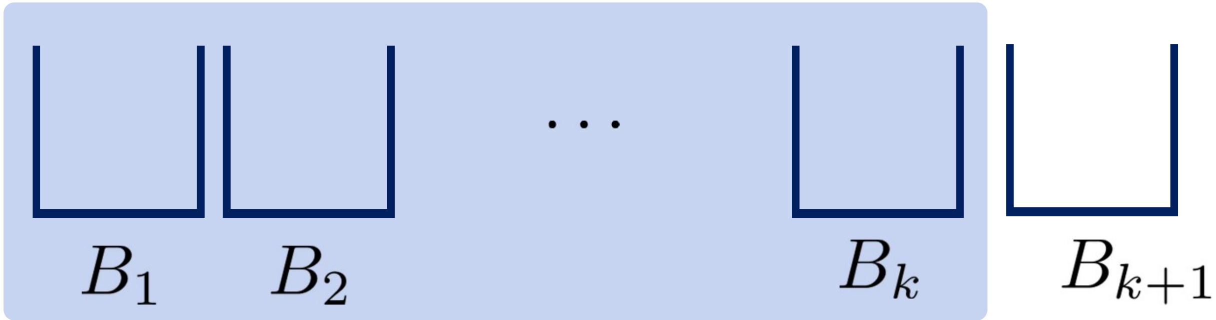
So, if we can prove the following, then we are done!

For $1 \leq k < n$, if $B_1, B_2, \ldots, B_k$ contain more than k marbles altogether,
Then we can put one more marble into B_{k+1} .

Another Induction

For $1 \leq k < n$, if $B_1, B_2, \ldots, B_k$ contain more than k marbles altogether,
Then we can put one more marble into B_{k+1} .

- This is obvious if $k = 1$ or 2.
- Suppose we have proven this for all numbers less than k. Let's prove for k boxes.



$> k$ marbles

- Goal: Put two marbles into B_k .
- If B_k has less than 2 marbles, then $B_1, B_2, \ldots, B_{k-1}$ altogether contain more than $k-1$ marbles. So, by induction, we can put one more marble into B_k . Do this twice if necessary.

Writing up…

We will induct on n. The problem is obvious if $n = 2$. Now, suppose that we have an algorithm A that does what we need in the case where we have $n - 1$ boxes. We will construct an algorithm for the case with n boxes.

Step 1: We will first show that we can make B_n non-empty. To do this, we shall use the following lemma:

Lemma. For $1 \leq k < n$, if $B_1, B_2, \ldots, B_k$ contain more than k marbles altogether, then we can put one more marble into B_{k+1}.

Proof: If $k = 1$ or $k = 2$, this is obvious. If B_k contains less than 2 marbles, then the boxes $B_1, B_2, \ldots, B_{k-1}$ altogether contain more than $k - 1$ marbles. So, by inductive assumption, we can put one more marble into B_k. Hence, we can make B_k contain at least 2 marbles. Thus, we can put one more marble into B_{k+1}.

So, if B_n is empty, then the boxes $B_1, B_2, \ldots, B_{n-1}$ altogether contain more than $n - 1$ marbles. Therefore, we can put one marble into B_n.

Writing up…

Step 2: Once B_n is non-empty, dump all but one marble from B_n into B_{n-1}. Now, we will apply the algorithm A on the boxes $B_1, B_2, \ldots, B_{n-1}$ with the following modification: whenever we have to make a move on B_{n-1}, we move the marble in B_n into B_{n-1} first, and then do the desired move on B_{n-1}. This ensures that B_{n-1} contains at least 2 marbles whenever we need to make a move on B_{n-1}. When A is finished, all boxes $B_1, B_2, \ldots, B_{n-1}, B_n$ will each contain one marble. So, we are done!

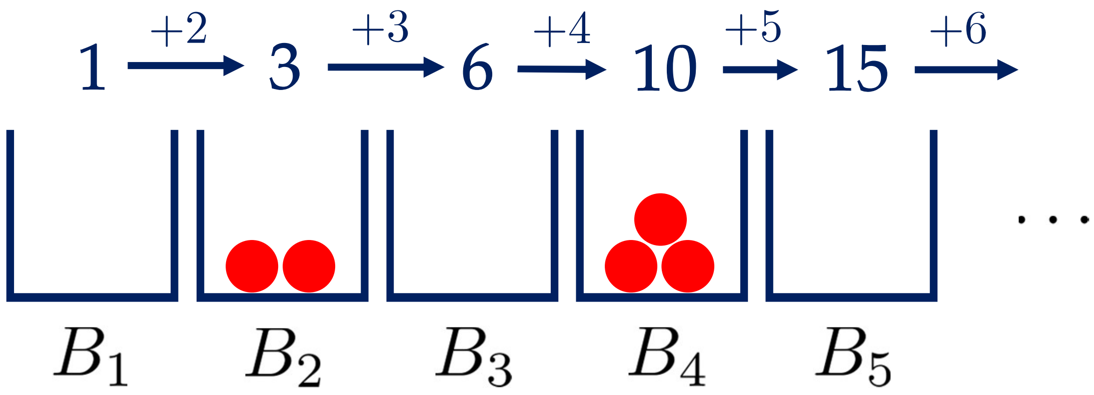
Alternate method to put a marble into B_n

As long as B_n is empty, we can still make a move.

Strategy: Just do any move on any of $B_1, B_2, \ldots, B_{n-1}$ as long as it is possible.

If we cannot do our strategy anymore, then it means that we have put a marble into B_n. So, we just need to show that our strategy terminates.

Alternate method to put a marble into B_n



- To each marble in box B_k , assign the weight $1+2+3+\cdots+k$.
- Let W be the sum of all the weights of the marbles. What is the behaviour of W?
- W increases with each move.
- But, since there are only finitely many possible configurations, W cannot increase forever. Bravo!

It’s time for a break!

See you on Problem Solving Session.