# International Mathematical Olympiad 2019 

Manuel Eberl

September 13, 2023


#### Abstract

This entry contains formalisations of the answers to three of the six problem of the International Mathematical Olympiad 2019, namely Q1, Q4, and Q5. The reason why these problems were chosen is that they are particularly amenable to formalisation: they can be solved with minimal use of libraries. The remaining three concern geometry and graph theory, which, in the author's opinion, are more difficult to formalise resp. require a more complex library.


## Contents

1 Q1 2
2 Q4 2
2.1 Auxiliary facts . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2.2 Main result . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3 Q5 5
3.1 Definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
3.2 Correctness of the measure . . . . . . . . . . . . . . . . . . . 8
3.3 Average-case analysis . . . . . . . . . . . . . . . . . . . . . . . 9

## 1 Q1

```
theory IMO2019-Q1
    imports Main
begin
```

Consider a function $f: \mathbb{Z} \rightarrow \mathbb{Z}$ that fulfils the functional equation $f(2 a)+$ $2 f(b)=f(f(a+b))$ for all $a, b \in \mathbb{Z}$.
Then $f$ is either identically 0 or of the form $f(x)=2 x+c$ for some constant $c \in \mathbb{Z}$.

## context

fixes $f::$ int $\Rightarrow$ int and $m::$ int
assumes $f$-eq: $f(2 * a)+2 * f b=f(f(a+b))$
defines $m \equiv(f 0-f(-2))$ div 2
begin
We first show that $f$ is affine with slope $(f(0)-f(-2)) /$ 2. This follows from plugging in $(0, b)$ and $(-1, b+1)$ into the functional equation.

```
lemma f-eq':fx=m*x+f0
<proof\rangle
```

This version is better for the simplifier because it prevents it from looping. lemma $f$-eq'-aux $[s i m p]$ : NO-MATCH $0 x \Longrightarrow f x=m * x+f 0$〈proof〉

Plugging in $(0,0)$ and $(0,1)$.
lemma $f$-classification: $(\forall x . f x=0) \vee(\forall x . f x=2 * x+f 0)$

$$
\langle p r o o f\rangle
$$

end
It is now easy to derive the full characterisation of the functions we considered:

```
theorem
    fixes \(f::\) int \(\Rightarrow\) int
    shows \((\forall a b . f(2 * a)+2 * f b=f(f(a+b))) \longleftrightarrow\)
        \((\forall x . f x=0) \vee(\forall x . f x=2 * x+f 0)(\) is ?lhs \(\longleftrightarrow\) ? \(\quad\) rhs \()\)
\(\langle p r o o f\rangle\)
end
```


## 2 Q4

theory IMO2019-Q4
imports Prime-Distribution-Elementary.More-Dirichlet-Misc
begin
Find all pairs $(k, n)$ of positive integers such that $k!=\prod_{i=0}^{n-1}\left(2^{n}-2^{i}\right)$.

### 2.1 Auxiliary facts

```
lemma Sigma-insert: Sigma (insert x A) f=(\lambday.(x,y))'fx\cupSigma A f
    <proof>
lemma atLeastAtMost-nat-numeral:
    {(m::nat)..numeral k}=
        (if m\leqnumeral k then insert (numeral k) {m..pred-numeral k} else {})
    <proof>
lemma greaterThanAtMost-nat-numeral:
    {(m::nat )<..numeral k}=
        (if m<numeral k then insert (numeral k) {m<..pred-numeral k} else {})
    <proof>
lemma fact-ge-power:
    fixes c :: nat
    assumes fact n0 \geqc^ n0 c \leqn0 + 1
    assumes n\geqn0
    shows fact n \geqc^n
    <proof>
lemma prime-multiplicity-prime:
    fixes p q :: ' }a\mathrm{ :: factorial-semiring
    assumes prime p prime q
    shows multiplicity p q=( if p=q then 1 else 0)
    <proof>
```

We use Legendre's identity from the library. One could easily prove the property in question without the library, but it probably still saves a few lines.
legendre-aux (related to Legendre's identity) is the multiplicity of a given prime in the prime factorisation of $n!$.

```
lemma multiplicity-prime-fact:
    fixes p :: nat
    assumes prime p
    shows multiplicity p (fact n) = legendre-aux n p
<proof\rangle
```

The following are simple and trivial lower and upper bounds for legen-dre-aux:
lemma legendre-aux-ge:
assumes prime $p k \geq 1$
shows legendre-aux $k p \geq$ nat $\lfloor k / p\rfloor$
$\langle p r o o f\rangle$
lemma legendre-aux-less:
assumes prime p $k \geq 1$
shows legendre-aux $k p<k /(p-1)$

〈proof〉

### 2.2 Main result

Now we move on to the main result: We fix two numbers $n$ and $k$ with the property in question and derive facts from that.
The triangle number $T=n(n+1) / 2$ is of particular importance here, so we introduce an abbreviation for it.

```
context
    fixes kn :: nat and rhs T :: nat
    defines rhs \equiv(\prodi<n.2^n-2 ^}i
    defines T\equiv(n*(n-1)) div 2
    assumes pos:k>0n>0
    assumes k-n: fact k=rhs
begin
```

We can rewrite the right-hand side into a more convenient form:

```
lemma rhs-altdef: rhs \(=2{ }^{\wedge} T *\left(\prod i=1 . . n .2{ }^{\wedge} i-1\right)\)
\(\langle p r o o f\rangle\)
```

The multiplicity of 2 in the prime factorisation of the right-hand side is precisely $T$.
lemma multiplicity-2-rhs [simp]: multiplicity 2 rhs $=T$ $\langle p r o o f\rangle$

From Legendre's identities and the associated bounds, it can easily be seen that $\lfloor k / 2\rfloor \leq T<k$ :
lemma $k$-gt- $T: k>T$ $\langle p r o o f\rangle$
lemma T-ge-half- $k: T \geq k$ div 2
$\langle p r o o f\rangle$
It can also be seen fairly easily that the right-hand side is strictly smaller than $2^{n^{2}}$ :
lemma rhs-less: rhs $<2^{\wedge} n^{2}$
$\langle p r o o f\rangle$
It is clear that $2^{n^{2}} \leq 8^{T}$ and that $8^{T}<T$ ! if $T$ is sufficiently big. In this case, 'sufficiently big' means $T \geq 20$ and thereby $n \geq 7$. We can therefore conclude that $n$ must be less than 7 .

```
lemma n-less-7: n < 7
<proof\rangle
```

We now only have 6 values for $n$ to check. Together with the bounds that we obtained on $k$, this only leaves a few combinations of $n$ and $k$ to check,
and we do precisely that and find that $n=k=1$ and $n=2, k=3$ are the only possible combinations．
lemma $n$－$k$－in－set：$(n, k) \in\{(1,1),(2,3)\}$
$\langle p r o o f\rangle$
end
Using this，deriving the final result is now trivial：

```
theorem \(\left\{(n, k) . n>0 \wedge k>0 \wedge\right.\) fact \(k=\left(\prod i<n\right.\). 2 \(\left.\left.n-2{ }^{\wedge} i:: n a t\right)\right\}=\)
\(\{(1,1),(2,3)\}\)
    (is ?lhs = ? rhs \()\)
\(\langle p r o o f\rangle\)
end
```


## 3 Q5

theory IMO2019－Q5
imports Complex－Main
begin
Given a sequence $\left(c_{1}, \ldots, c_{n}\right)$ of coins，each of which can be heads $(H)$ or tails $(T)$ ，Harry performs the following process：Let $k$ be the number of coins that show $H$ ．If $k>0$ ，flip the $k$－th coin and repeat the process．Otherwise， stop．
What is the average number of steps that this process takes，averaged over all $2^{n}$ coin sequences of length $n$ ？

## 3．1 Definition

We represent coins as Booleans，where True indicates $H$ and False indicates T．Coin sequences are then simply lists of Booleans．
The following function flips the $i$－th coin in the sequence（in Isabelle，the convention is that the first list element is indexed with 0 ）．
definition flip ：：bool list $\Rightarrow$ nat $\Rightarrow$ bool list where

$$
\text { flip xs } i=x s[i:=\neg x s!i]
$$

lemma flip－Cons－pos［simp］：$n>0 \Longrightarrow$ flip $(x \# x s) n=x \#$ flip $x s(n-1)$〈proof〉
lemma flip－Cons－0［simp］：flip $(x \# x s) 0=(\neg x) \# x s$〈proof〉
lemma fip－append1［simp］：$n<$ length $x s \Longrightarrow$ flip（xs＠ys）$n=$ flip xs $n$＠ys and flip－append2［simp］：$n \geq$ length $x s \Longrightarrow n<$ length $x s+$ length $y s \Longrightarrow$ flip $(x s @ y s) n=x s @ f l i p y s(n-l e n g t h x s)$

```
<proof\rangle
```

lemma length－flip［simp］：length（fip xs $i$ ）$=$ length $x s$ $\langle p r o o f\rangle$

The following function computes the number of $H$ in a coin sequence．
definition heads $::$ bool list $\Rightarrow$ nat where heads $x s=$ length（filter id xs）
lemma heads－True $[$ simp $]$ ：heads（True $\#$ xs $)=1+$ heads xs
and heads－False［simp］：heads（False \＃xs）＝heads xs
and heads－append $[$ simp $]$ ：heads（xs＠ys）＝heads $x s+$ heads ys
and heads－Nil［simp］：heads []$=0$
〈proof〉
lemma heads－Cons：heads $(x \# x s)=($ if $x$ then heads $x s+1$ else heads $x s)$〈proof〉
lemma heads－pos：True $\in$ set $x s \Longrightarrow$ heads $x s>0$
〈proof〉
lemma heads－eq－0［simp］：True $\notin$ set $x s \Longrightarrow$ heads $x s=0$ $\langle p r o o f\rangle$
lemma heads－eq－0－iff［simp］：heads $x s=0 \longleftrightarrow$ True $\notin$ set xs $\langle p r o o f\rangle$
lemma heads－pos－iff［simp］：heads xs $>0 \longleftrightarrow$ True $\in$ set $x s$〈proof〉
lemma heads－le－length：heads $x s \leq$ length $x s$ $\langle p r o o f\rangle$

The following function performs a single step of Harry＇s process．
definition harry－step $::$ bool list $\Rightarrow$ bool list where
harry－step $x s=$ flip xs（heads xs -1 ）
lemma length－harry－step［simp］：length（harry－step xs）＝length xs $\langle p r o o f\rangle$

The following is the measure function for Harry＇s process，i．e．how many steps the process takes to terminate starting from the given sequence．We define it like this now and prove the correctness later．

```
function harry-meas where
    harry-meas xs =
        (if xs = [] then 0
            else if hd xs then 1 + harry-meas (tl xs)
            else if \neglast xs then harry-meas (butlast xs)
            else let n = length xs in harry-meas (take (n-2) (tl xs)) +2*n-1)
```

```
    <proof>
termination <proof\rangle
lemmas [simp del] harry-meas.simps
```

We now prove some simple properties of harry－meas and harry－step．
We prove a more convenient case distinction rule for lists that allows us to distinguish between lists starting with True，ending with False，and starting with False and ending with True．

```
lemma head-last-cases [case-names Nil True False False-True]:
    assumes \(x s=[] \Longrightarrow P\)
    assumes \(\bigwedge\) ys. \(x s=\) True \(\# y s \Longrightarrow P\) yss. xs \(=y s @[\) False \(] \Longrightarrow P\)
        \(\wedge\) ys. xs \(=\) False \# ys @ \([\) True \(] \Longrightarrow P\)
    shows \(P\)
〈proof〉
lemma harry-meas-Nil [simp]: harry-meas []\(=0\)
    〈proof〉
```

lemma harry-meas-True-start [simp]: harry-meas (True \# xs) $=1+$ harry-meas
xs
〈proof〉
lemma harry-meas-False-end [simp]: harry-meas (xs @ [False]) = harry-meas xs
〈proof〉
lemma harry-meas-False-True: harry-meas (False \# xs @ [True]) = harry-meas
$x s+2 *$ length $x s+3$
$\langle$ proof〉
lemma harry-meas-eq-0 [simp]:
assumes True $\notin$ set xs
shows harry-meas xs $=0$
〈proof〉

If the sequence starts with $H$ ，the process runs on the remaining sequence until it terminates and then flips this $H$ in another single step．

```
lemma harry-step-True-start [simp]:
    harry-step (True \# xs) \(=\) (if True \(\in\) set xs then True \# harry-step xs else False
\# xs)
    〈proof〉
```

If the sequence ends in $T$ ，the process simply runs on the remaining sequence as if it were not present．

```
lemma harry-step-False-end [simp]:
    assumes True \in set xs
    shows harry-step (xs @ [False]) = harry-step xs @ [False]
```

```
\langleproof\rangle
```

If the sequence starts with $T$ and ends with $H$, the process runs on the remaining sequence inbetween as if these two were not present, eventually leaving a sequence that consists entirely if $T$ except for a single final $H$.

```
lemma harry-step-False-True:
    assumes True \in set xs
    shows harry-step (False # xs @ [True]) = False # harry-step xs @ [True]
\langleproof\rangle
```

That sequence consisting only of $T$ except for a single final $H$ is then turned into an all- $T$ sequence in $2 n+1$ steps.

```
lemma harry-meas-Falses-True [simp]: harry-meas (replicate n False @ [True]) =
\(2 * n+1\)
\(\langle\) proof〉
```

lemma harry-step-Falses-True [simp]:
$n>0 \Longrightarrow$ harry-step (replicate $n$ False @ $[$ True $]$ ) $=$ True $\#$ replicate $(n-1)$
False @ [True]
$\langle p r o o f\rangle$

### 3.2 Correctness of the measure

We will now show that harry-meas indeed counts the length of the process. As a first step, we will show that if there is a $H$ in a sequence, applying a single step decreases the measure by one.

```
lemma harry-meas-step-aux:
    assumes True \(\in\) set xs
    shows harry-meas xs \(=\) Suc (harry-meas (harry-step xs))
    \(\langle p r o o f\rangle\)
```

lemma harry-meas-step: True $\in$ set $x s \Longrightarrow$ harry-meas (harry-step xs) $=$ harry-meas
$x s-1$
$\langle p r o o f\rangle$

Next, we show that the measure is zero if and only if there is no $H$ left in the sequence.

```
lemma harry-meas-eq-0-iff [simp]: harry-meas xs =0 \longleftrightarrow True & set xs
<proof\rangle
```

It follows by induction that if the measure of a sequence is $n$, then iterating the step less than $n$ times yields a sequence with at least one $H$ in it, but iterating it exactly $n$ times yields a sequence that contains no more $H$.

```
lemma True-in-funpow-harry-step:
    assumes n < harry-meas xs
    shows True \in set ((harry-step ~n n) xs)
    <proof\rangle
```

```
lemma True-notin-funpow-harry-step: True \(\notin\) set ((harry-step ~~harry-meas xs)
xs)
\(\langle p r o o f\rangle\)
```

This shows that the measure is indeed the correct one：It is the smallest number such that iterating Harry＇s step that often yields a sequence with no heads in it．
theorem harry－meas $x s=\left(\right.$ LEAST $n$ ．True $\notin$ set $\left(\left(\right.\right.$ harry－step $\left.\left.\left.{ }^{\sim} n\right) x s\right)\right)$
$\langle p r o o f\rangle$

## 3．3 Average－case analysis

The set of all coin sequences of a given length．
definition seqs where seqs $n=\{x s$ ：：bool list．length $x s=n\}$
lemma length－seqs $[$ dest $]: x s \in$ seqs $n \Longrightarrow$ length $x s=n$
$\langle p r o o f\rangle$
lemma seqs－0［simp］：seqs $0=\{[]\}$
〈proof〉
The coin sequences of length $n+1$ are simply what is obtained by appending either $H$ or $T$ to each coin sequence of length $n$ ．
lemma seqs－Suc：seqs $($ Suc $n)=(\lambda x s$ ．True $\#$ xs $)$＇seqs $n \cup(\lambda x s$ ．False $\#$ xs $)$＇ seqs $n$ $\langle p r o o f\rangle$

The set of coin sequences of length $n$ is invariant under reversal．
lemma seqs－rev $[$ simp $]$ ：rev＇seqs $n=$ seqs $n$
$\langle p r o o f\rangle$
Hence we get a similar decomposition theorem that appends at the end．
lemma seqs－Suc＇：seqs $($ Suc $n)=(\lambda x s . x s @[T r u e])$＇seqs $n \cup(\lambda x s . x s @[F a l s e])$
＇seqs $n$
$\langle p r o o f\rangle$
lemma finite－seqs［intro］：finite（seqs $n$ ）
$\langle p r o o f\rangle$
lemma card－seqs $\left[\right.$ simp］：card（seqs $n$ ）$=2{ }^{\text {へ }} n$
$\langle p r o o f\rangle$
lemmas seqs－code $[$ code $]=$ seqs－0 seqs－Suc
The sum of the measures over all possible coin sequences of a given length （defined as a recurrence relation；correctness proven later）．

```
fun harry-sum :: nat \(\Rightarrow\) nat where
    harry-sum 0 \(=0\)
| harry-sum (Suc 0) \(=1\)
| harry-sum \((\) Suc \((\) Suc \(n))=2 * \operatorname{harry-sum}(\) Suc \(n)+(2 * n+4) * 2^{\wedge} n\)
lemma Suc-Suc-induct: \(P 0 \Longrightarrow P(\) Suc 0\() \Longrightarrow(\bigwedge n . P n \Longrightarrow P(\) Suc \(n) \Longrightarrow P\)
\((\) Suc \((\) Suc \(n))) \Longrightarrow P n\)
    〈proof〉
```

The recurrence relation really does describe the sum over all measures：

```
lemma harry-sum-correct: harry-sum \(n=\) sum harry-meas (seqs \(n\) )
```

$\langle p r o o f\rangle$
lemma harry－sum－closed－form－aux： 4 ＊harry－sum $n=n *(n+1) * \boldsymbol{2}^{\text {＾}} n$〈proof〉

Solving the recurrence gives us the following solution：
theorem harry－sum－closed－form：harry－sum $n=n *(n+1) * 2$＾$n$ div 4 $\langle p r o o f\rangle$

The average is now a simple consequence：
definition harry－avg where harry－avg $n=$ harry－sum $n /$ card（seqs $n$ ）
corollary harry－avg $n=n *(n+1) / 4$
$\langle p r o o f\rangle$
end

## References

［1］60th International Mathematical Olympiad．https：／／www．imo2019．uk／ wp－content／uploads／2018／07／solutions－r856．pdf．11th－22nd July 2019.

