# Algebra

## 更像算子的概念

1. **Algebraic Numbers** \in Real: Numbers that are the roots of polynomial equations with integer coefficients.

*Example:* ( \sqrt{2}, \frac{1}{3} ).

这个概念感觉可以去掉，这种开根、多项式的操作可以体现在语义解析这一步，要么用sqrt(2)、1/3这种python可以处理的形式，要么用latex的形式。

1. **Reciprocal** \in Rational Numbers: A number that, when multiplied by the original number, results in 1.

Example: The reciprocal of (4) is (\frac{1}{4}).

倒数这个概念，更像一种算子操作，如example可以翻译为 "Reciprocal(4) = 1/4"，而输入4和输出1/4直接作为Number概念即可。

1. **PolyDegree** \in Integers : The highest power of the variable in the polynomial. For example, in 3x^4 + 2x + 5, the degree is 4.

这个也像是算子，输入是一个多项式，输出则是最高次幂。毕竟如果题目表述为 "已知多项式A的最高次幂为a，多项式B的最高次幂为b"，可以翻译为 "A: Polynomial; B: Polynomial; PolyDegree(A) = a; PolyDegree(B) = b"

1. **Coefficient** \in Number: The coefficient of the term in a polynomial. In 4x^3 + 2x^2, the coefficient for 4x^3 term is 4.
2. **Polyroots** \in Number: The values of the variable that make the polynomial equal to zero. Ex(x^2- 5x + 6 has two roots x = 2 and x = 3)

多项式中某一项的系数，使多项式为0的根，可以用算子表示。

1. **PolynomialTerm** \in Polynomial: Each individual part of a polynomial separated by addition or subtraction.

Example: In (3x^2 + 4x + 5), (3x^2), (4x), and (5) are terms.

这个也可以转化为算子，用"A: Polynomial; PolynomialTerm(A) = {(3x^2 + 4x + 5), (3x^2), (4x), (5)}" 来表示

1. **ConstantTerm** \in PolynomialTerm: A term in the polynomial with a degree of 0, representing a fixed value.

Example: In (2x^2 + 3x + 4), the constant term is (4).

1. **Rational Root** \in Polyroots: A root of the polynomial that can be expressed as a fraction of integers.

Example: For (x^2 - 3x + 2), the roots (x = 1) and (x = 2) are rational.

1. **Irrational Root** \in Polyroots: A root of the polynomial that cannot be expressed as a fraction.

Example: The roots of (x^2 - 2 = 0) are (x = \pm \sqrt{2}), which are irrational.

1. **Factor** \in Polynomial: A polynomial that divides another polynomial exactly.

Example: ((x - 2)) and ((x - 3)) are factors of (x^2 - 5x + 6).

1. **Irreducible Polynomial** \in Polynomial: A polynomial that cannot be factored into polynomials of lower degree with coefficients in a field.

Example: (x^2 + 1) is irreducible over the real numbers.

1. **Critical Point** \in Polyroots: A value of the variable where the derivative of the polynomial equals zero.

Example: For (f(x) = x^3 - 3x^2), the critical points are (x = 0) and (x = 2).

........

这些问题可以待初步构建完成后整体讨论

## 缺失概念

1. 正整数 PositiveIntegers
2. 分段函数 PiecewiseFunction

## 缺失算子

1. 对数字进行向上取整或者向下取整

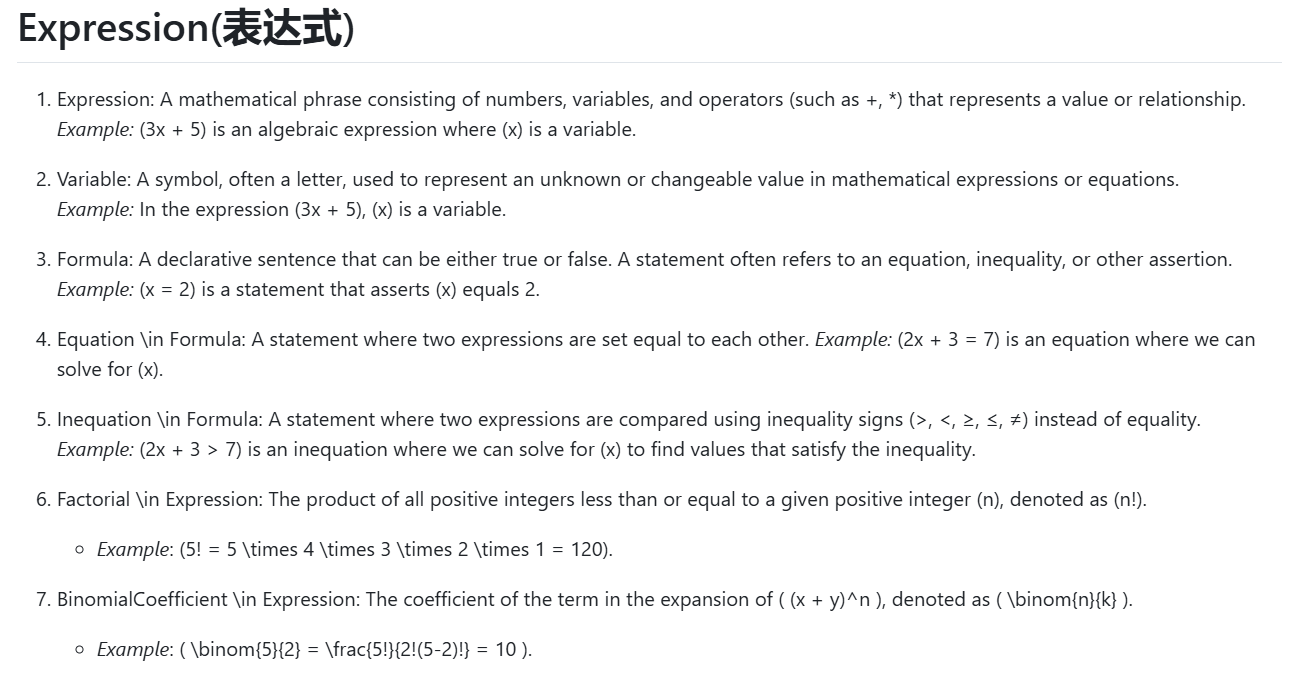
Get\_Number\_Floor(Number) -> Integar

Get\_Number\_Ceil(Number) -> Integar

1. 对数字进行取绝对值

Abs(a: Number) -> 非负数

**所有方程、函数、多项式都是一个抽象的概念，在题目中才会有具体的”表达式“，即“实例”，所以最好有个算子专门表达这个”实例“。**



**所以对于这里的定义，我建议抽象的概念：**

**函数对应Function, 例如f(x) = 2x**

**方程对应Equation / Inequation，例如 2x + 1 = 3 or x + 3 > 2x**

**多项式对应Polynomial，例如 2x + 2**

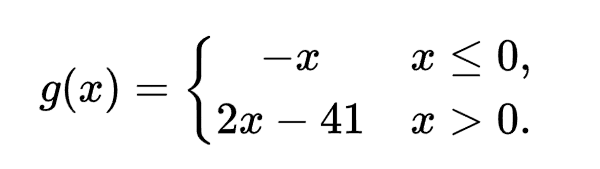
**关于他们具体的示例表达，上述的例子，就统一用**

**Get\_xxx\_Expression( Union[Function, Formula, Equation]) -> Expression获取实例表达式**

**Get\_xxx\_Variable(Union[Function, Formula, Equation]) -> Set(Variable)获取变量集合**

**Get\_xxx\_Value(Union[Function, Formula], Variable：Number) -> Number获取变量取某个值时的函数、formula取值**

但针对函数的时候会出现下列分段函数情况：



一种解决方法是在获取表达式时加上定义域

g:Function

Get\_Function\_Variable(g) = {x}  
Get\_Function\_Expression(g) = {(-x, Is\_LessOrEqualThan(x,0)), (2\*x-41, Is\_GreaterThan(x,0)}

但这种方法的缺点在于针对非分段函数时，表示比较复杂。

所以另一种解决方法是再设计“分段函数”这个概念并针对他设计算子。

**我标注的时候采用的第二种方法。新增PiecewiseFunction概念**

1. 获取 函数 的表达式，变量，以及变量取某个值时对应的值

Get\_Function\_Expression(f: Function) -> Expression

Get\_Function\_Variable(f: Function) -> Set(Variable)

Get\_Function\_Value(f: Function, {Variable: Numebr}) -> Number

为了应对h = f(g(x))以及f(2x)这样的情况，新增算子, 返回f(g(x))

Get\_Function\_NestExpression(f: Function, g: Function) -> Expression

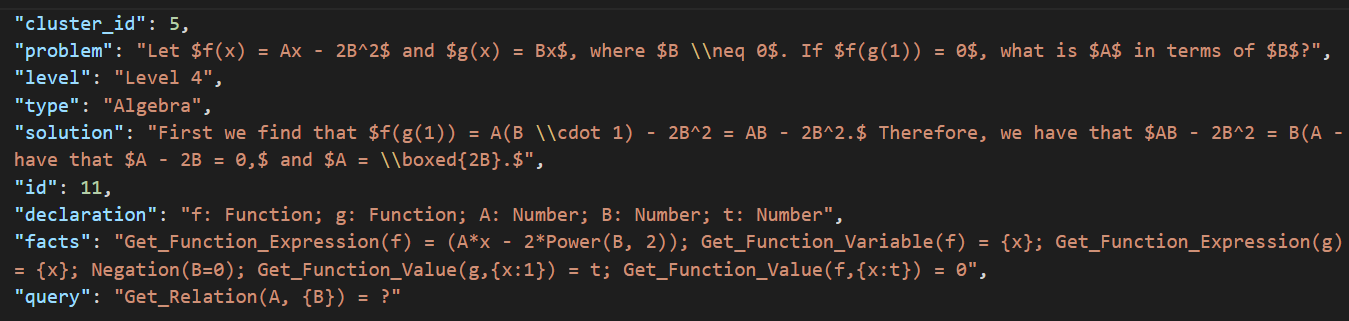
1. 获取 方程式 的表达式

Get\_Equation\_Expression(q:Equation) -> Expression (这里不太确定其类型）

Get\_Equation\_Variable(q:Equation) -> Set(Variable)

1. Get\_Relation(a: Union[Number,Variable], b: Set(Union[Number,Variable])) -> Formula

获取a和b之间的关系，主要是a = xxxb。a和b要么是未知具体值的Number，要么是两个变量，其中b是一个集合，毕竟有时候需要用x和y来表示a。



1. 获取函数的逆函数

Get\_Inverse\_Function(f: Function) -> Function

1. 表示否定

Negation(a: Assertion)

比如 m ≠ 0 表示为 Negation( m=0 )

1. 表示集合中元素的个数

NumberOfTerms(s: Set) -> 非负整数

1. 返回集合中的最大值，最小值

Max(s: Set) -> Number

Min(s: Set) -> Number

返回集合中的任意一个元素

OneOf(s: Set) -> SetTerm

返回集合中的非零元素

Get\_Set\_NonZeroTerms(s: Set) -> Set[SetTerm]

感觉也可以带个限制term的参数，Get\_Set\_Term(s: Set, 限制条件）

比如Get\_Set\_Term(s, x>0)表示返回集合中大于0的元素

返回集合中的元素的类型

Get\_Set\_Term\_Type(s:Set) -> NumberType

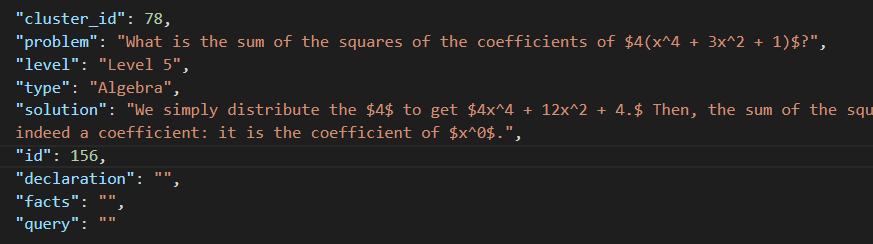
1. 返回集合所有元素的和

Get\_Set\_Sum(s: Set) -> Number

返回集合所有元素的积

Get\_Set\_Product(s: Set) -> Number

1. 对集合中的每个元素都施加某种操作，比如统一加1，每个元素都小于0，......



1. 序列元素的类型

Get\_Sequence\_Element\_Type(s: Sequence) -> NumberType

序列的第i个元素

Get\_Sequence\_Term(s: Sequence, n: Integers) -> 序列元素

1. 获取多项式第i项（幂为i)的系数

Get\_Polynomial\_Coefficient(p: Polynomial, i: 非负整数) -> Number

获取多项式最高次幂

Get\_Polynomial\_Degree(p: Polynomial) -> Numebr

获取多项式中所有的term（由加减法分开）

Get\_Polynomial\_Term(p: Polynomial) -> Set[PolynomialTerm]

1. 获取多项式的因子

Get\_Factors(p: Polynomial) -> Set(Formula)

获取多项式的 因式表达式

Get\_Factors\_Expression(p: Polynomial) -> Expression

获取多项式的标准展开式

Get\_Polynomial\_ExpandExpression(p: Polynomial) -> Expression

**遇到了一些几何题目（大多参考conic10k的本体）：**

**补充概念：**

Point：点

Curve: 曲线

Circle \in Curve：圆

Line \in Curve：直线

LineSegment：线段

axis：二维坐标系中的坐标轴

xAxis \in axis：二维坐标系中的横坐标

yAxis \in axis：二维坐标系中的纵坐标

Origin \in Point：二维坐标系中的原点

其实也可以是三维坐标中的，只要提前确定坐标空间即可

**补充算子：**

1. 点在曲线上

PointOnCurve(p: Point, c: Curve) -> bool

1. 线段的端点

EndpointOfLinesegment(l: Linesegment) = Set(Point)

线段的垂直平分线

PerpendicularBisector(l: LineSegment) -> Line

线段的中点

MidPoint(l: LineSegment) -> Point

1. 点的坐标

Coordinates(p: Point) = (x,y) 或者 (x,y,z) 具体看在什么坐标系下

点的横、纵坐标

XCoordinate(p: Point) -> Number

YCoordinate(p: Point) -> Number

1. 两点、点与曲线、两曲线之间的距离

Distance(a: Union[Point, Curve], b: Union[Point, Curve]) -> Number

1. 获取曲线的表达式

Get\_Curve\_Expression(c: Curve) -> Expression

1. 获取圆的圆心

Get\_Circle\_Center(c: Circle) -> Point

1. 获取椭圆，抛物线，双曲线的顶点

Get\_Vertex(p: Union[ Parabola, ...]) -> Union[ Set[Point], Point]

1. 两曲线的交点，返回交点集合

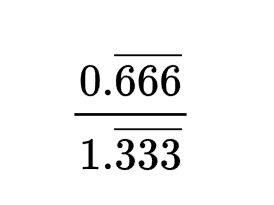
Intersection(a: Union[Axis, Curve], b: Union[Axis, Curve]) -> Set[point]

## 标注问题

1. 有的算子结果是集合或者元素，比如两曲线的交点，可以是一个点，也可以是多个点的集合，然后需要求这些点的坐标。而坐标算子Coordinate的输入只能是一个点？？

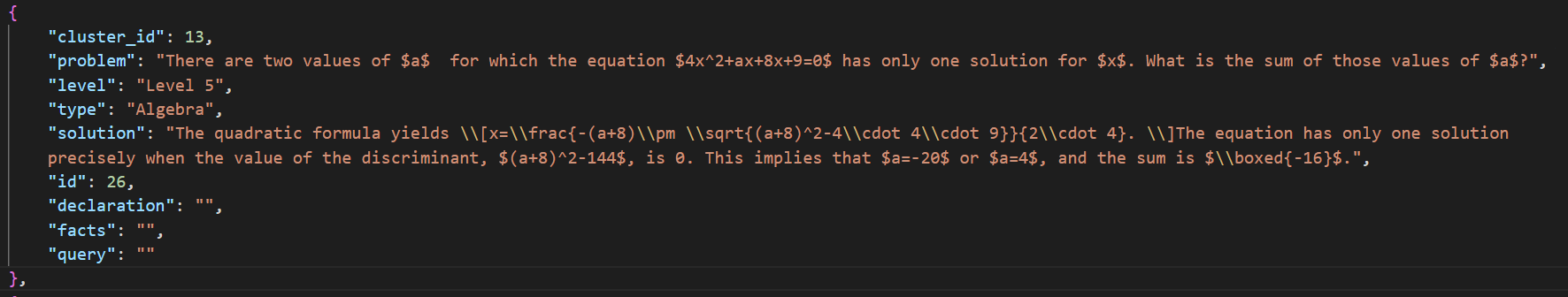
最简单的处理方法就是让Coordinate接受多种输入，是单个点就返回单个点坐标，多个点集合就返回集合中每个点的坐标。（暂时是这么处理的）

1. 对于加减乘除法，遇到x+1这种也要表示成 "Add({x: Formula}, {1: Formula})" 这种形式吗？对于解析这一步会不会太麻烦了？能否把这一步放到推理步骤？解析直接照搬原文本中的latex形式。
2. 循环数的表示



0.1212345345345...

暂时用latex形式表示，声明为RepatingDecimals类型



不太清楚该怎么标。

标注一：

a1: Number; a2: Number; f: Equation; g: Equation

Get\_Equation\_Expression(f) = (4\*Power(x,2) + a1\*x + 8\*x + 9 = 0);

Get\_Equation\_Expression(f) = (4\*Power(x,2) + a2\*x + 8\*x + 9 = 0);

NumberOfElements(Solve\_equation(f,x)) = 1

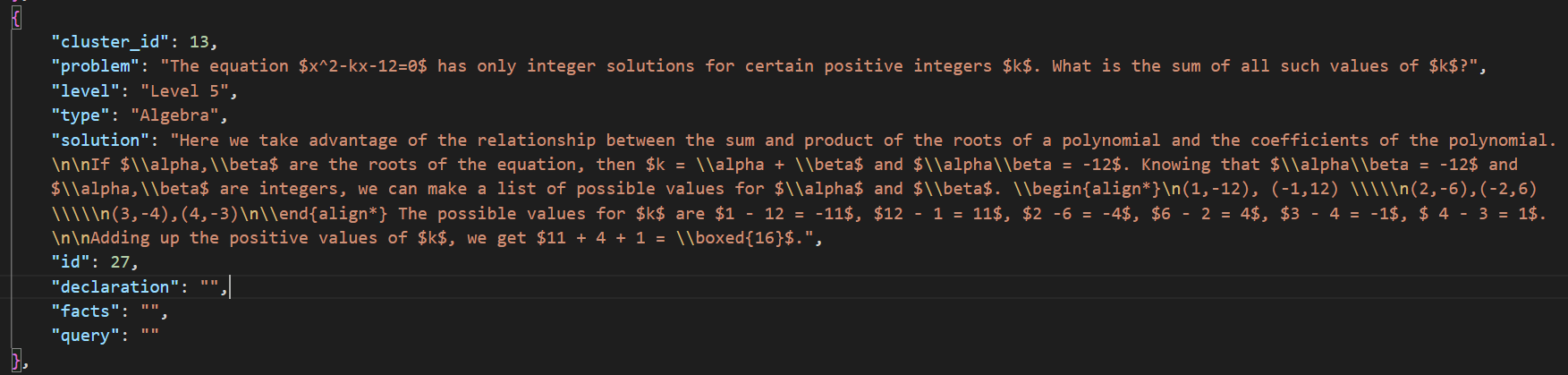
NumberOfElements(Solve\_equation(g,x)) = 1

Negation(a1 = a2)

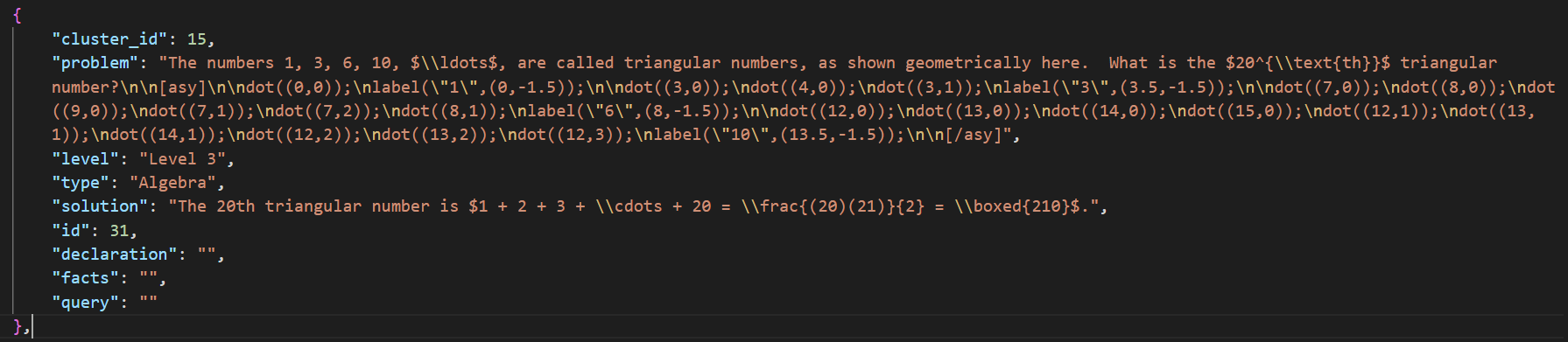
a1 + a2 = ?

标注二：

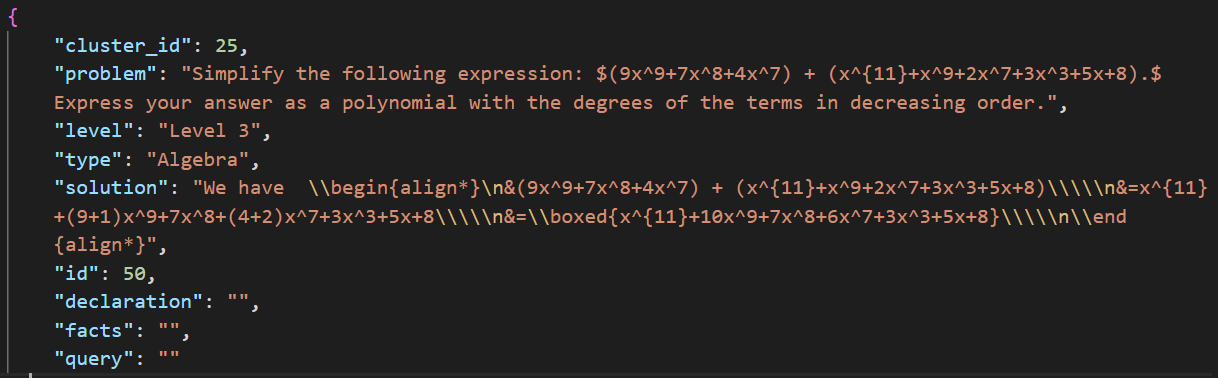
。。。



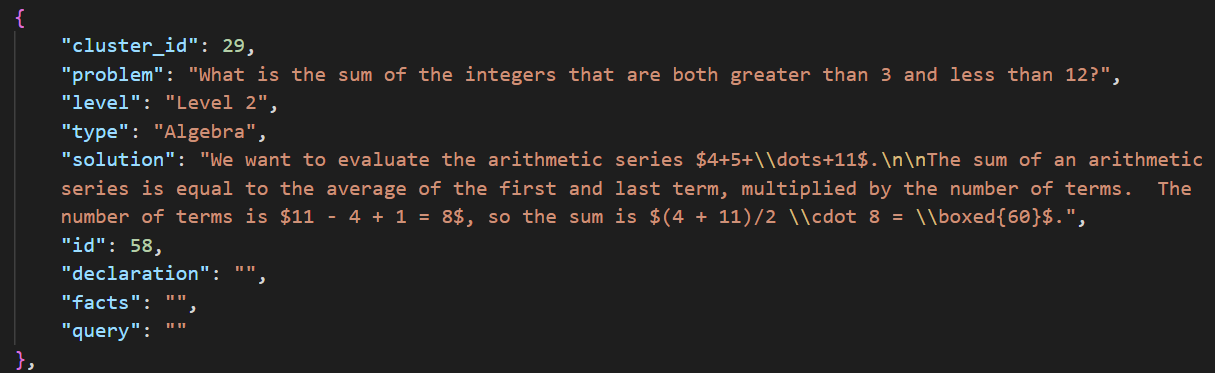
1. 问题带有几何的图形信息，【asy】之间的部分

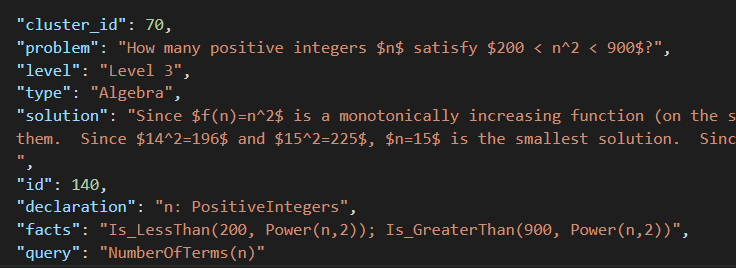


1. 从普通多项式到标准多项式

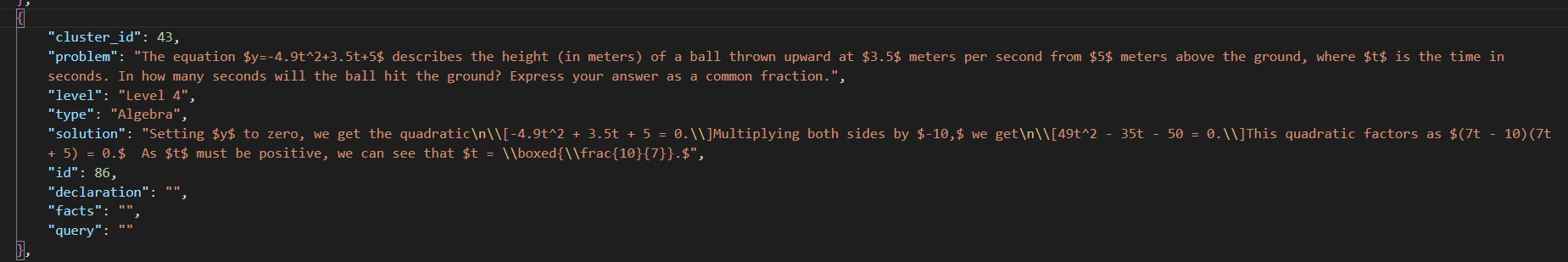


1. 某些数满足xx条件，然后求这些数的和

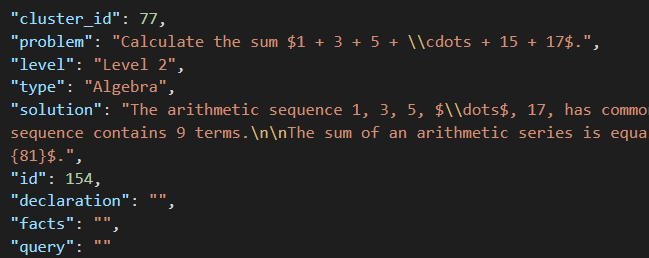




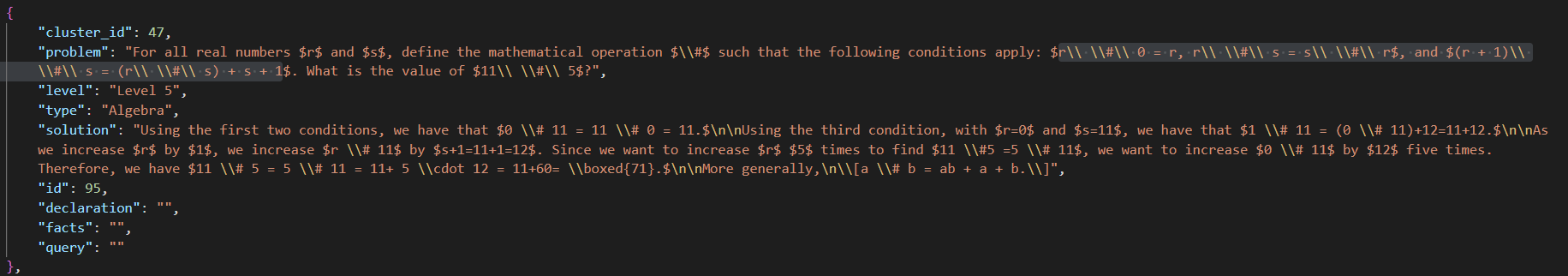
1. 表示数列中a1项到a100项的和
2. 带有常识



需要自己找规律



1. 定义新的运算符



1. 涉及form

