

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30} .

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30} .
- **Solution 1:** We can (ab)use leading zeroes.
For example, “00” could encode a different value than “0”.

K: Cracked Encoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30} .
- **Solution 1:** We can (ab)use leading zeroes.
For example, “00” could encode a different value than “0”.
- **Solution 2:** Treat the initial ‘1’ as implicit (we do not need to encode 0).
For example, encode 42 as “01010”. When decoding, prepend the implicit ‘1’ again.
(This is also how the mantissa of IEEE-754 floating-point numbers are encoded.)

K: Cracked Encoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30} .
- **Solution 1:** We can (ab)use leading zeroes.
For example, "00" could encode a different value than "0".
- **Solution 2:** Treat the initial '1' as implicit (we do not need to encode 0).
For example, encode 42 as "01010". When decoding, prepend the implicit '1' again.
(This is also how the mantissa of IEEE-754 floating-point numbers are encoded.)
 - Edge-case: you cannot encode the integer 1 as the empty string ("").
Either increment the values by one, or encode 1 as a string of 29 '1's.

K: Cracked Encoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30} .
- **Solution 1:** We can (ab)use leading zeroes.
For example, "00" could encode a different value than "0".
- **Solution 2:** Treat the initial '1' as implicit (we do not need to encode 0).
For example, encode 42 as "01010". When decoding, prepend the implicit '1' again.
(This is also how the mantissa of IEEE-754 floating-point numbers are encoded.)
 - Edge-case: you cannot encode the integer 1 as the empty string ("").
Either increment the values by one, or encode 1 as a string of 29 '1's.

Statistics: ... submissions, ... accepted, ... unknown