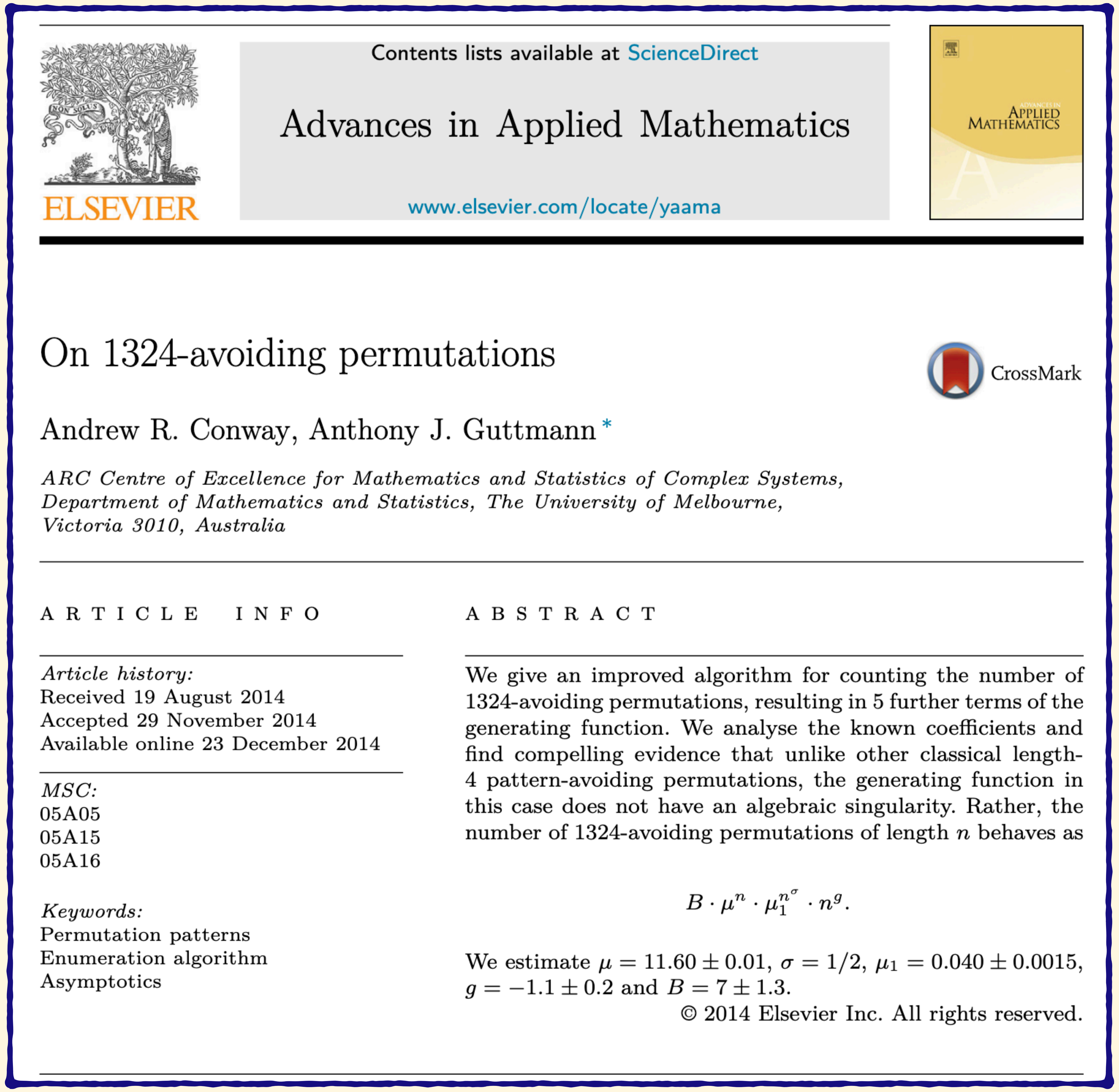
Counting Pattern-Avoiding Permutations Quickly

Jay Pantone
Marquette University

joint work with Christian Bean, Keele University

Guttmann 2025 - 80 and (still) counting
July 1, 2025

How I “met” Tony



Contents lists available at ScienceDirect

Advances in Applied Mathematics

www.elsevier.com/locate/yaama

On 1324-avoiding permutations

Andrew R. Conway, Anthony J. Guttmann *

ARC Centre of Excellence for Mathematics and Statistics of Complex Systems, Department of Mathematics and Statistics, The University of Melbourne, Victoria 3010, Australia

A R T I C L E I N F O

Article history:
Received 19 August 2014
Accepted 29 November 2014
Available online 23 December 2014

MSC:
05A05
05A15
05A16

Keywords:
Permutation patterns
Enumeration algorithm
Asymptotics

A B S T R A C T

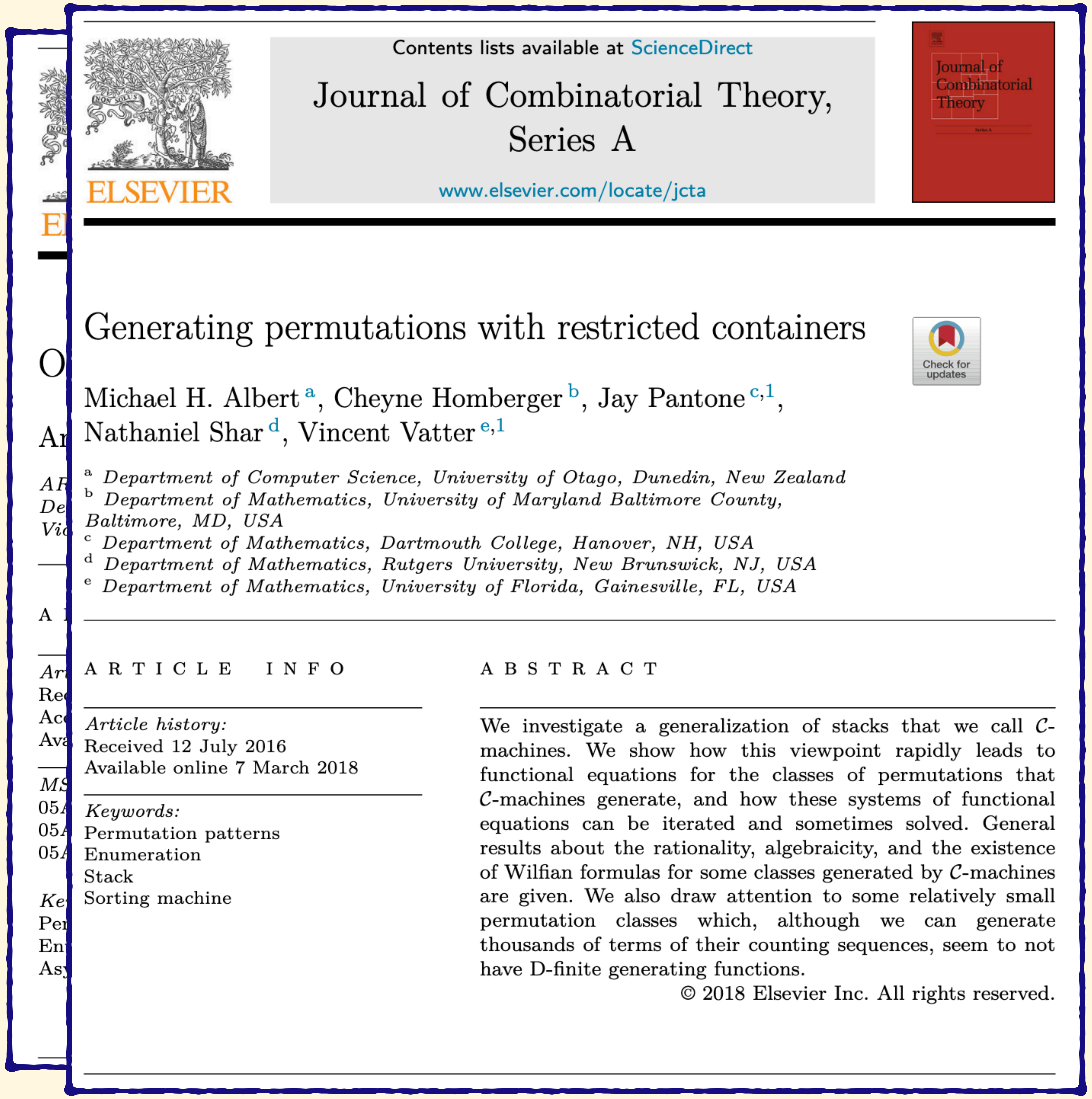
We give an improved algorithm for counting the number of 1324-avoiding permutations, resulting in 5 further terms of the generating function. We analyse the known coefficients and find compelling evidence that unlike other classical length-4 pattern-avoiding permutations, the generating function in this case does not have an algebraic singularity. Rather, the number of 1324-avoiding permutations of length n behaves as

$$B \cdot \mu^n \cdot \mu_1^{n^\sigma} \cdot n^g.$$

We estimate $\mu = 11.60 \pm 0.01$, $\sigma = 1/2$, $\mu_1 = 0.040 \pm 0.0015$, $g = -1.1 \pm 0.2$ and $B = 7 \pm 1.3$.

- computed the number of 1324-avoiding permutations of lengths 31–36
- asymptotic analysis

How I "met" Tony


Contents lists available at ScienceDirect

Journal of Combinatorial Theory, Series A

www.elsevier.com/locate/jcta

Generating permutations with restricted containers

Michael H. Albert[a], Cheyne Homberger[b], Jay Pantone[c,1], Nathaniel Shar[d], Vincent Vatter[e,1]

[a] *Department of Computer Science, University of Otago, Dunedin, New Zealand*
[b] *Department of Mathematics, University of Maryland Baltimore County, Baltimore, MD, USA*
[c] *Department of Mathematics, Dartmouth College, Hanover, NH, USA*
[d] *Department of Mathematics, Rutgers University, New Brunswick, NJ, USA*
[e] *Department of Mathematics, University of Florida, Gainesville, FL, USA*

ARTICLE INFO

Article history:
Received 12 July 2016
Available online 7 March 2018

Keywords:
Permutation patterns
Enumeration
Stack
Sorting machine

ABSTRACT

We investigate a generalization of stacks that we call $\mathcal{C}$-machines. We show how this viewpoint rapidly leads to functional equations for the classes of permutations that $\mathcal{C}$-machines generate, and how these systems of functional equations can be iterated and sometimes solved. General results about the rationality, algebraicity, and the existence of Wilfian formulas for some classes generated by $\mathcal{C}$-machines are given. We also draw attention to some relatively small permutation classes which, although we can generate thousands of terms of their counting sequences, seem to not have D-finite generating functions.

- computed the number of
  - (1432, 1324)-avoiding permutations up to size 1000
  - (1432, 1243)-avoiding permutations up to size 1000
  - (1324, 1234)-avoiding permutations up to size 600
  - (1432, 1324, 1243)-avoiding permutations up to size 5000
- could not conjecture a generating function
- did not know how to do asymptotic analysis!

How I "met" Tony

Contents lists available at ScienceDirect

Journal of Combinatorial Theory, Series A

www.elsevier.com/locate/jcta

ELSEVIER

Journal of Combinatorial Theory

Gen…

Micha…
Natha…

a Depa…
b Depa… Baltim…
c Depa…
d Depa…
e Depa…

A R T …

Article…
Receive…
Availab…

Keywords:
Permutation patterns
Enumeration
Stack
Sorting machine

… equations can be iterated and sometimes solved. General results about the rationality, algebraicity, and the existence of Wilfian formulas for some classes generated by $\mathcal{C}$-machines are given. We also draw attention to some relatively small permutation classes which, although we can generate thousands of terms of their counting sequences, seem to not have D-finite generating functions.

Jay Pantone <jay.pantone@gmail.com> Jul 19, 2014, 7:44 PM

to tony.guttmann, Vince

Dear Prof. Guttmann,

Vince told me today that he shared some of our data with you. While the sequence he sent is correct, the functional equation is not. I've attached a pdf which has the correct pair of functional equations that give the desired series.

It should be noted that we did not use these functional equations to actually get the 350 terms. We used a structural description of the class itself (with a dynamic programming approach) to get the terms in about a day of computation on a laptop.

It's very exciting that you are interested in the sequence. It would be great if we could show that it is non-D-finite!

Best,
Jay Pantone

- computed the number of
  - (1432, 1324)-avoiding permutations up to size 1000
  - (1432, 1243)-avoiding permutations up to size 1000
  - …ing permutations up to size 600
  - …voiding permutations up to size 5000
- …ecture a generating
- …ow to do asymptotic

How I actually met Tony

National Science Foundation – East Asia and Pacific Summer Institutes

Jay Pantone <jay.pantone@gmail.com> Wed, Oct 22, 2014, 3:37 PM

to tonyg

Dear Professor Guttmann,

My name is Jay Pantone, and I am a graduate student of Vince Vatter at the University of Florida. We corresponded briefly in July about the asymptotics of the class Av(4231, 4123) — in particular, that it seems likely to be non-D-finite.

The National Science Foundation runs a summer program each year called East Asia and Pacific Summer Institutes (EAPSI) in which they fund travel for graduate students to various counties in the East Asia and Pacific regions for eight weeks over the summer. I am considering applying for this program, and I was wondering if you would be willing to be my mentor / host.

I would be interested in continuing the study of the permutation class Av(4231, 4123), as well as other classes that arise by a similar construction. This is all very closely related to the study of sorting / generating permutations by stacks in series, which is a problem that Vince tells me you have been thinking about. Of course, I'm also open to any other problems you may be working on.

The program runs between late June and mid-August of 2015, and is co-sponsored by the Australian Academy of Science.

Thanks for your time.

Best wishes,
Jay Pantone

How I actually met Tony

National Science Foundation – East Asia and Pacific Summer Institutes

Jay Pantone <jay.pantone@gmail.com> Wed, Oct 22, 2014, 3:37 PM

to tonyg

Dear Pr

My name
correspo
to be no

The Nati
Institutes
regions f
would be

I would l
arise by
stacks in
any othe

The prog
Science.

Thanks f

Best wishes,
Jay Pantone

Tony Guttmann <tony.guttmann@gmail.com> Wed, Oct 22, 2014, 6:33 PM

to Vince, me

Dear Jay,

I'd be very happy to have you visit. I'm overseas from early April until early June, but will be back for the period of the program, and have no teaching duties then, so the timing is perfect.
We can provide you with office space, computing facilities, library access etc etc. There may be some local bureaucracy at this end, but we can sort that out closer to the time.

I'm sure there will be plenty of things to work on.

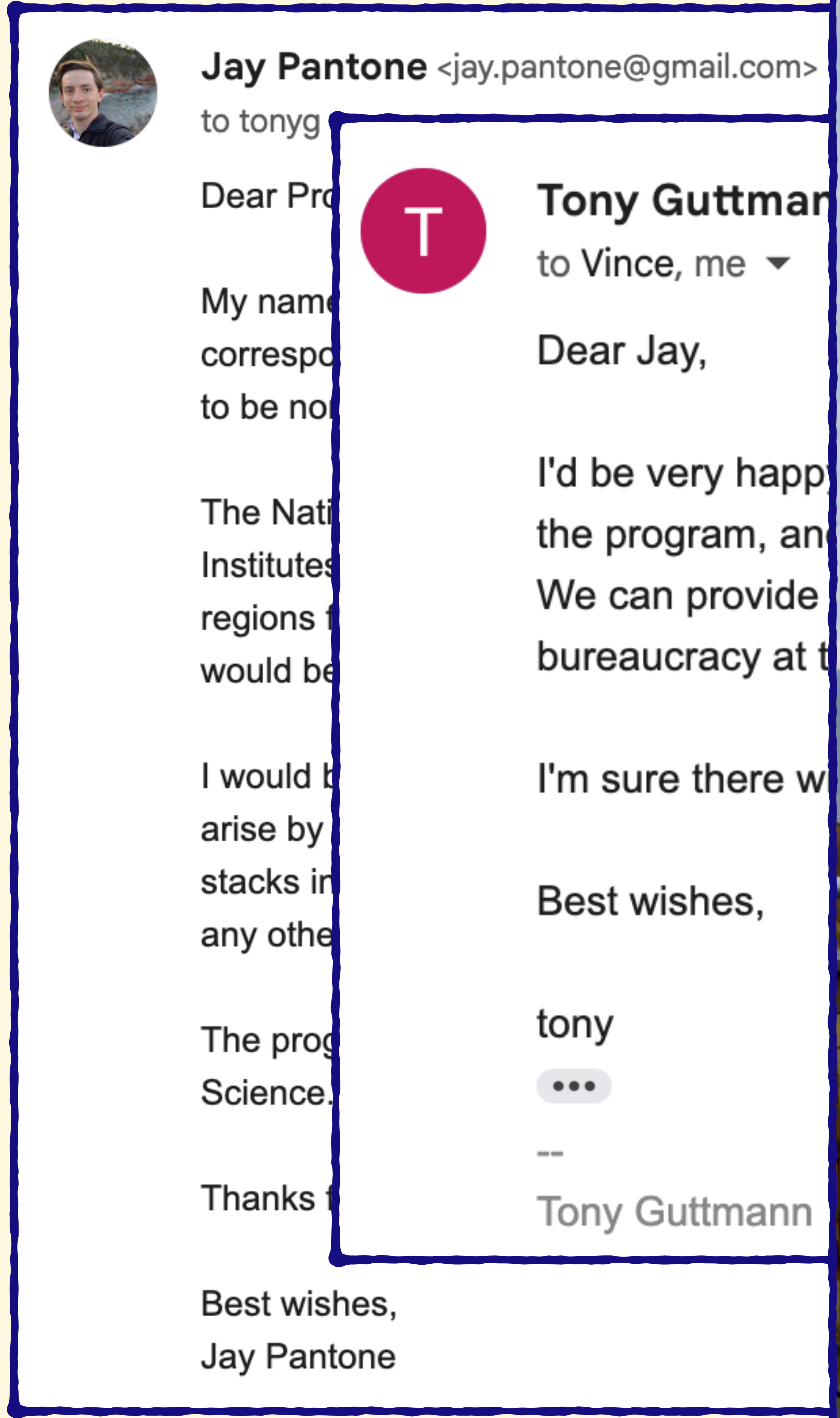
Best wishes,

tony

--
Tony Guttmann

How I actually met Tony

National Science Foundation - East Asia and Pacific Summer Institutes


Jay Pantone <jay.pantone@gmail.com>
to tonyg

Dear Pro

My name
correspo
to be no

The Nati
Institutes
regions f
would be

I would l
arise by
stacks in
any othe

The prog
Science.

Thanks f

Best wishes,
Jay Pantone

Tony Guttman
to Vince, me

Dear Jay,

I'd be very happy
the program, an
We can provide
bureaucracy at t

I'm sure there w

Best wishes,

tony

--
Tony Guttmann

1324 Again

Contents lists available at ScienceDirect

Advances in Applied Mathematics

www.elsevier.com/locate/yaama

ELSEVIER

1324-avoiding permutations revisited

Andrew R. Conway, Anthony J. Guttmann, Paul Zinn-Justin *

School of Mathematics and Statistics, The University of Melbourne, Victoria 3010, Australia

ARTICLE INFO

Article history:
Received 28 October 2017
Accepted 9 January 2018
Available online 7 February 2018

MSC:
05A05
05A15
05A16

ABSTRACT

We give an improved algorithm for counting the number of 1324-avoiding permutations, resulting in 14 further terms of the generating function, which is now known for all lengths ≤ 50. We re-analyse the generating function and find additional evidence for our earlier conclusion that unlike other classical length-4 pattern-avoiding permutations, the generating function does not have a simple power-law singularity, but rather, the number of 1324-avoiding permutations of length n behaves as

$$B \cdot \mu^n \cdot \mu_1^{\sqrt{n}} \cdot n^g.$$

We estimate $\mu = 11.600 \pm 0.003$, $\mu_1 = 0.0400 \pm 0.0005$, $g = -1.1 \pm 0.1$ while the estimate of B depends sensitively on the precise value of μ, μ_1 and g. This reanalysis provides substantially more compelling arguments for the presence of the stretched exponential term $\mu_1^{\sqrt{n}}$.

- computed the number of 1324-avoiding permutations up to length 50
- asymptotic analysis

1324 Again

ELS

1324

Andr

School
Austra

A R T

Article
Receiv
Accept
Availab

MSC:
05A05
05A15
05A16

Table 1
An example of the state as the permutation 5 4 2 7 10 8 9 1 3 6 is built up. The numbers above the link diagrams indicate the actual numbers that the link end represents. The numbers in parentheses correspond to the four types of insertion given in the text. Conversely, consider the similar permutation 5 4 2 7 10 6 9 1 3 8 which is not 1324 avoiding. The first five elements are the same; the sixth element is not allowed, as it would have to go inside the loop ending at 7.

Element	Notes	Result
	Start state	∅
5	Not consecutive with anything; future elements could go either side. (1)	5
4	Consecutive with 5 and so merged into it. (3)	4 – 5
2	New link as not consecutive with anything. (1)	2 4 – 5
7	Larger than a previous link, makes constraint that no new elements between 2 and 7 may be added until every element greater than 7 has been added. (1)	2 4 – 5 7
10	Removed from consideration as largest element. (3)	2 4 – 5 7
8	Merged with 7. (2)	2 4 – 5 7 – 8
9	Merges the 7 – 8 link with the largest element; said link removed from consideration. (4)	2 4 – 5
1	Merges with 2 link. (3)	1 – 2 4 – 5
3	Merges the 1 – 2 and 3 – 5 links. (4)	1 – 5
6	Merges the 1 – 5 link with the largest element. (4)	∅

ed the number of 1324-
g permutations up to length 50

otic analysis

1324 Again

ELS

1324

Andr

School
Austra

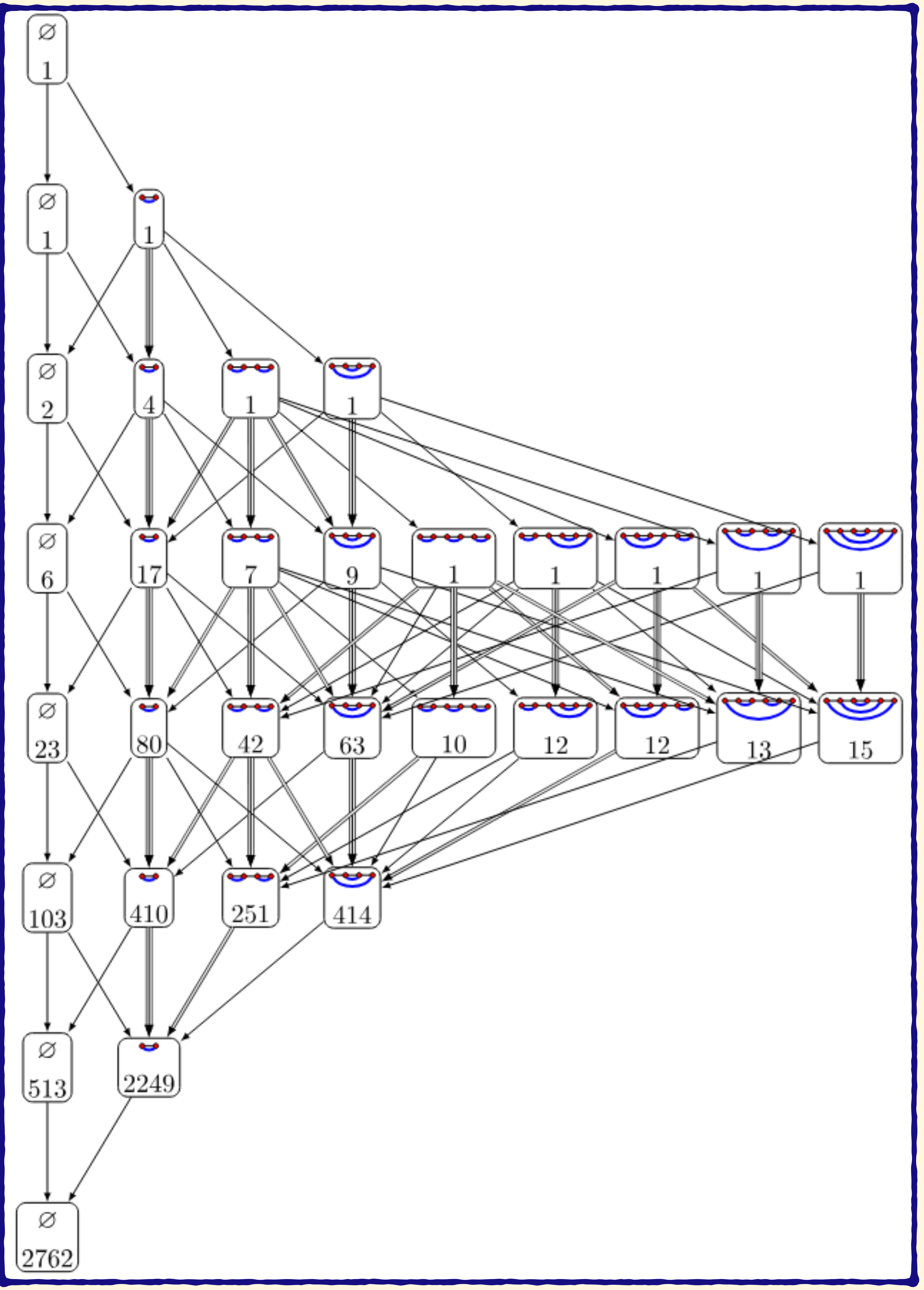
A R T

Article
Receiv
Accept
Availab

MSC:
05A05
05A15
05A16

Table 1
An example of the state as the permutation 5 4 2 7 10 8 9 1 3 6 is built up. The numbers above the link diagrams indicate the actual numbers that the link end represents. The numbers in parentheses correspond to the four types of insertion given in the text. Conversely, consider the similar permutation 5 4 2 7 10 6 9 1 3 8 which is not 1324 avoiding. The first five elements are the same; the sixth element is not allowed, as it would have to go inside the loop ending at 7.

Element	Notes	Result
	Start state	∅
5	Not consecutive with anything; future elements could go either side. (1)	5
4	Consecutive with 5 and so merged into it. (3)	4 – 5
2	New link as not consecutive with anything. (1)	2 4 – 5
7	Larger than a previous link, makes constraint that no new elements between 2 and 7 may be added until every element greater than 7 has been added. (1)	2 4 – 5 7
10	Removed from consideration as largest element. (3)	2 4 – 5 7
8	Merged with 7. (2)	2 4 – 5 7 – 8
9	Merges the 7 – 8 link with the largest element; said link removed from consideration. (4)	2 4 – 5
1	Merges with 2 link. (3)	1 – 2 4 – 5
3	Merges the 1 – 2 and 3 – 5 links. (4)	1 – 5
6	Merges the 1 – 5 link with the largest element. (4)	∅



1324 Again

Based on the first 50 terms, they predict the exponential growth rate for the number of 1324-avoiding permutations is ~11.600.

1324 Again

Based on the first 50 terms, they predict the exponential growth rate for the number of 1324-avoiding permutations is ~11.600.

The number of link patterns of size n is the nth Catalan number, so their exponential growth rate is 4.

1324 Again

Based on the first 50 terms, they predict the exponential growth rate for the number of 1324-avoiding permutations is ~11.600.

The number of link patterns of size n is the nth Catalan number, so their exponential growth rate is 4.

BUT: each operation adds at most one link, and takes away at most one link, and the counts we care about are the ones with zero links.

1324 Again

Based on the first 50 terms, they predict the exponential growth rate for the number of 1324-avoiding permutations is ~11.600.

The number of link patterns of size n is the nth Catalan number, so their exponential growth rate is 4.

BUT: each operation adds at most one link, and takes away at most one link, and the counts we care about are the ones with zero links.

So, they compute 1324-avoiding permutations to length n, we only need link patterns with at most $n/2$ links.

1324 Again

Based on the first 50 terms, they predict the exponential growth rate for the number of 1324-avoiding permutations is ~11.600.

The number of link patterns of size n is the nth Catalan number, so their exponential growth rate is 4.

BUT: each operation adds at most one link, and takes away at most one link, and the counts we care about are the ones with zero links.

So, they compute 1324-avoiding permutations to length n, we only need link patterns with at most $n/2$ links.

That makes this a $o((4+\epsilon)^{n/2}) = o((2+\epsilon)^n)$ algorithm!

1324 Again

It always bugged me that I didn't see the "big picture" of the paper.

1324 Again

It always bugged me that I didn’t see the “big picture” of the paper.

I downloaded the paper onto my iPad to re-read on the flight home from Permutation Patterns 2023 in Dijon.

1324 Again

It always bugged me that I didn’t see the “big picture” of the paper.

I downloaded the paper onto my iPad to re-read on the flight home from Permutation Patterns 2023 in Dijon.

By the time I landed, I understood the big picture, which gave me the idea for this project:

1324 Again

It always bugged me that I didn't see the "big picture" of the paper.

I downloaded the paper onto my iPad to re-read on the flight home from Permutation Patterns 2023 in Dijon.

By the time I landed, I understood the big picture, which gave me the idea for this project:

> Counting permutations avoiding *any* set of patterns by automatically discovering the "link patterns" for that set.

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

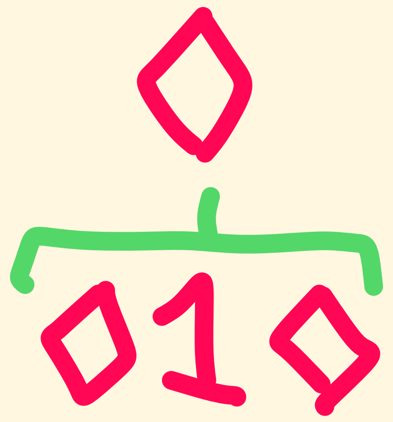
689274153 ◇

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153

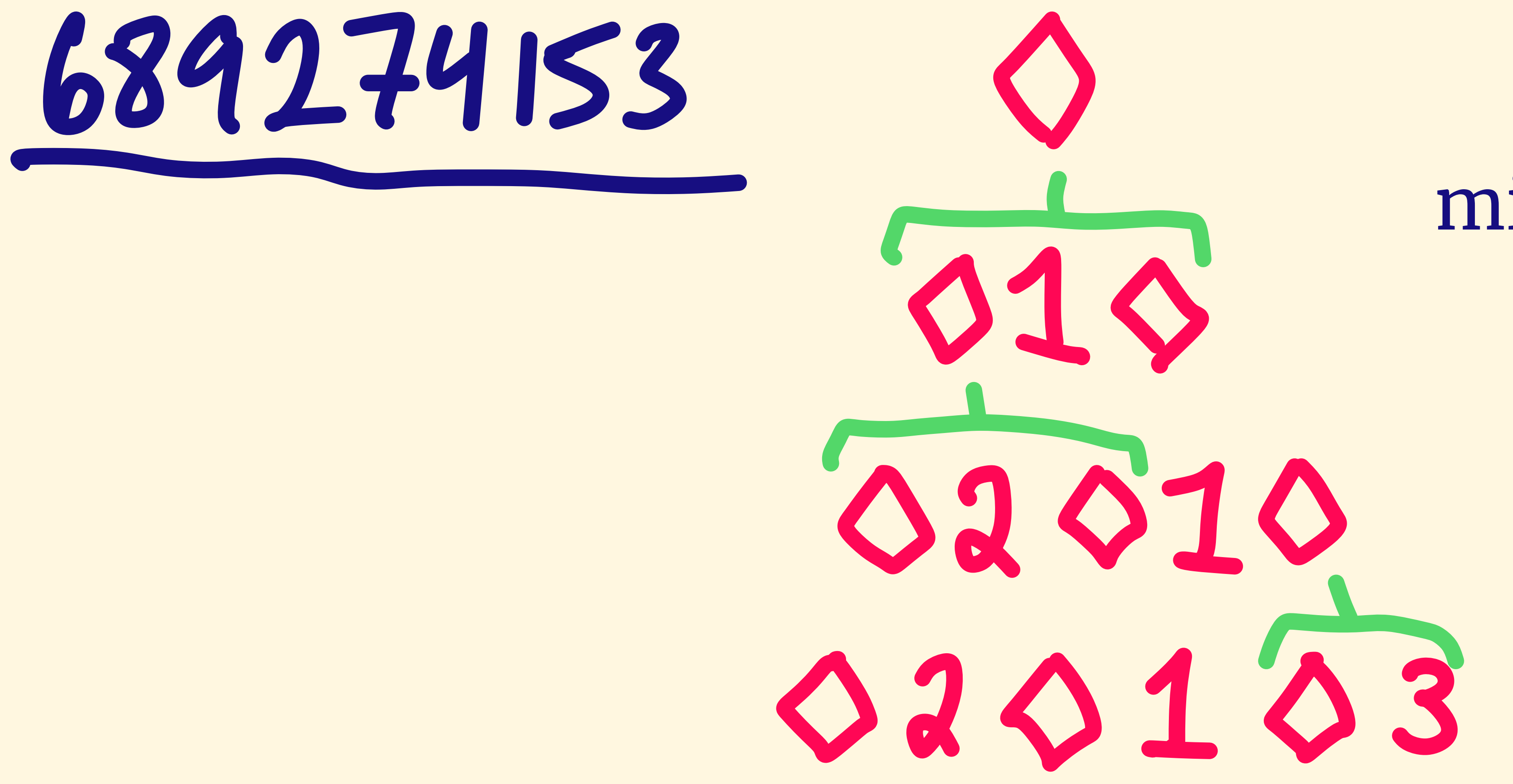
middle placement

middle placement

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.



middle placement

middle placement

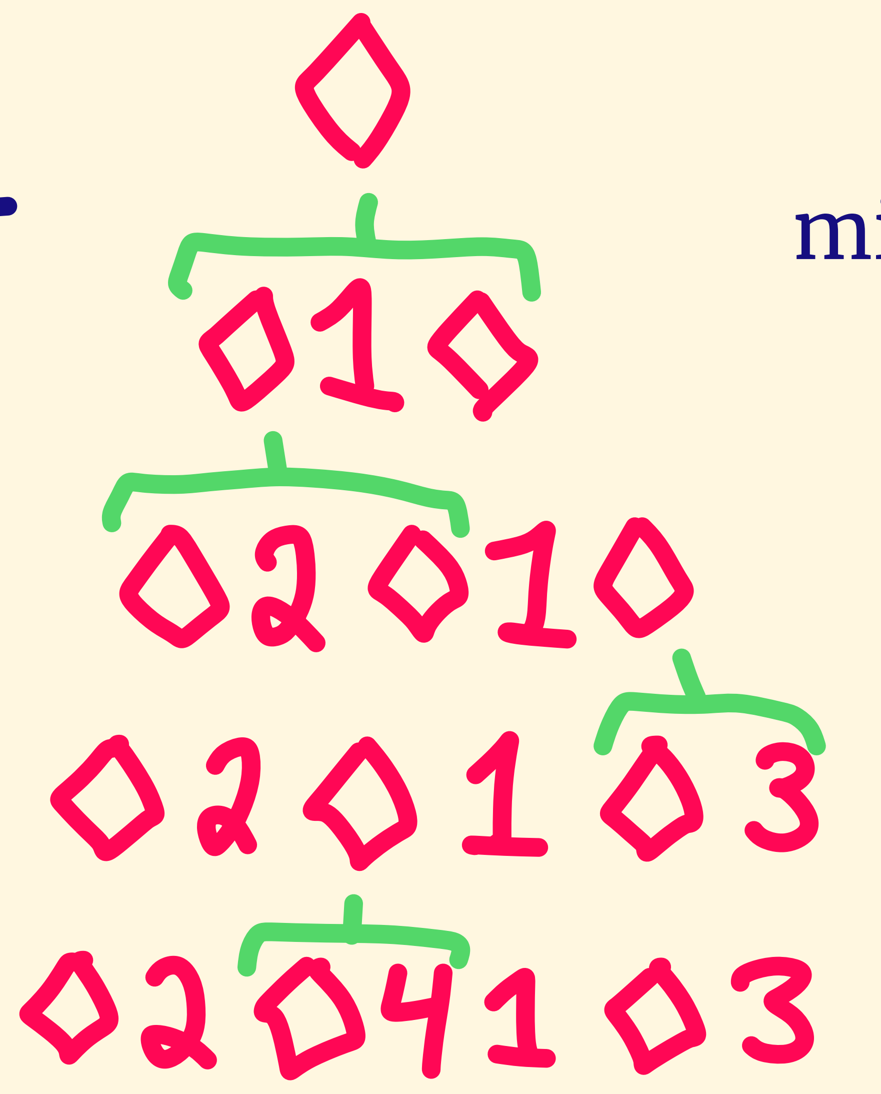
right placement

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

middle placement

right placement

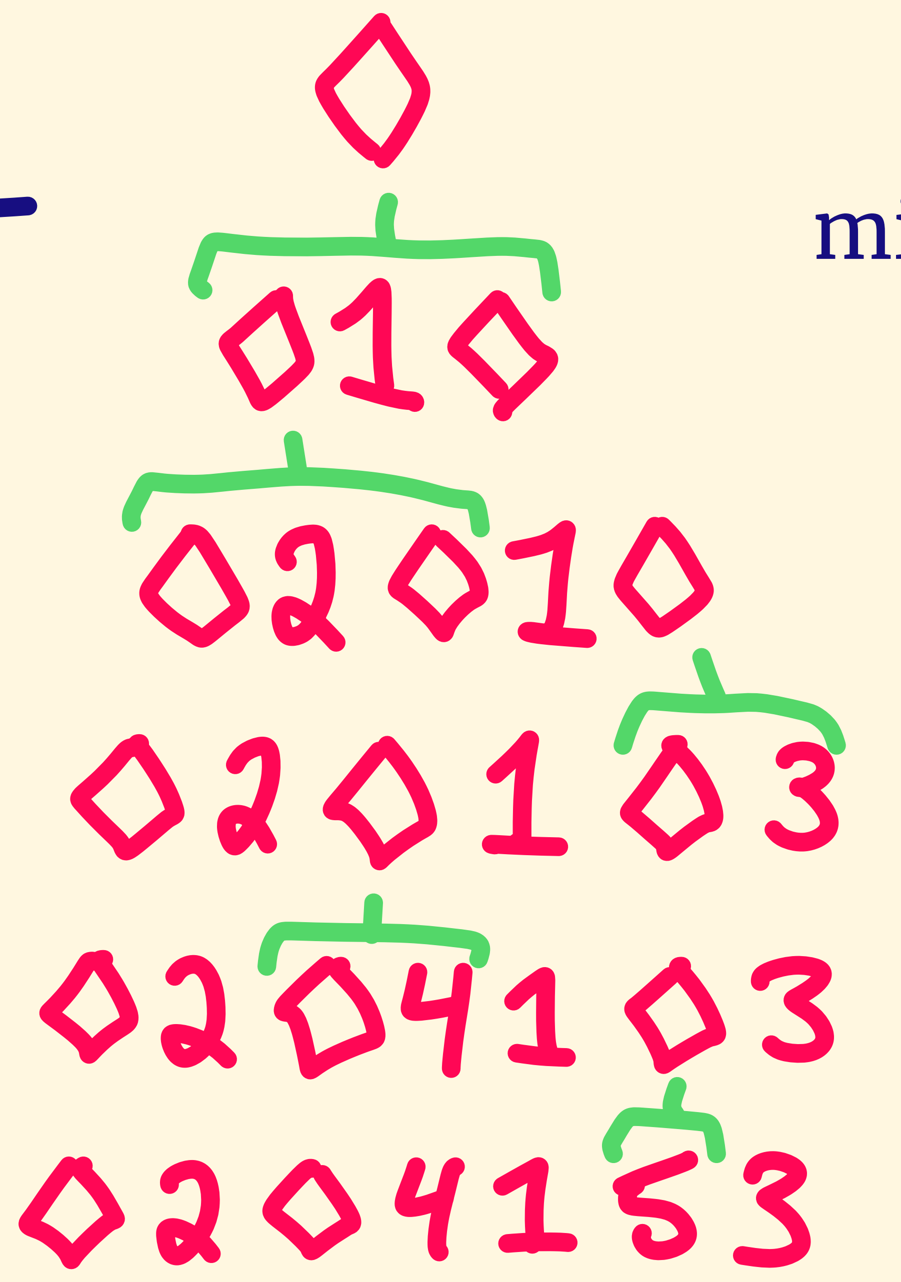
right placement

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

middle placement

right placement

right placement

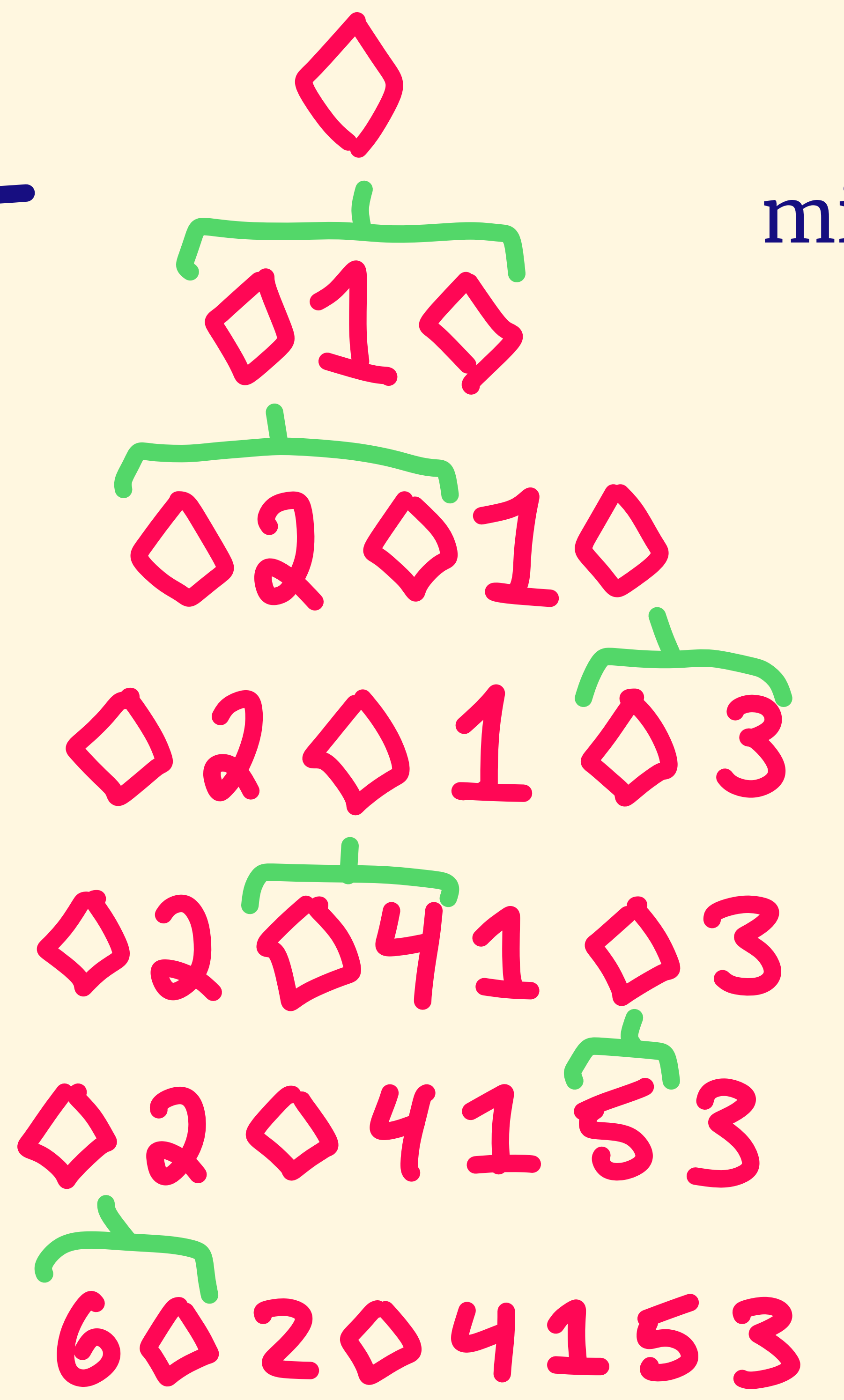
fill

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

middle placement

right placement

right placement

fill

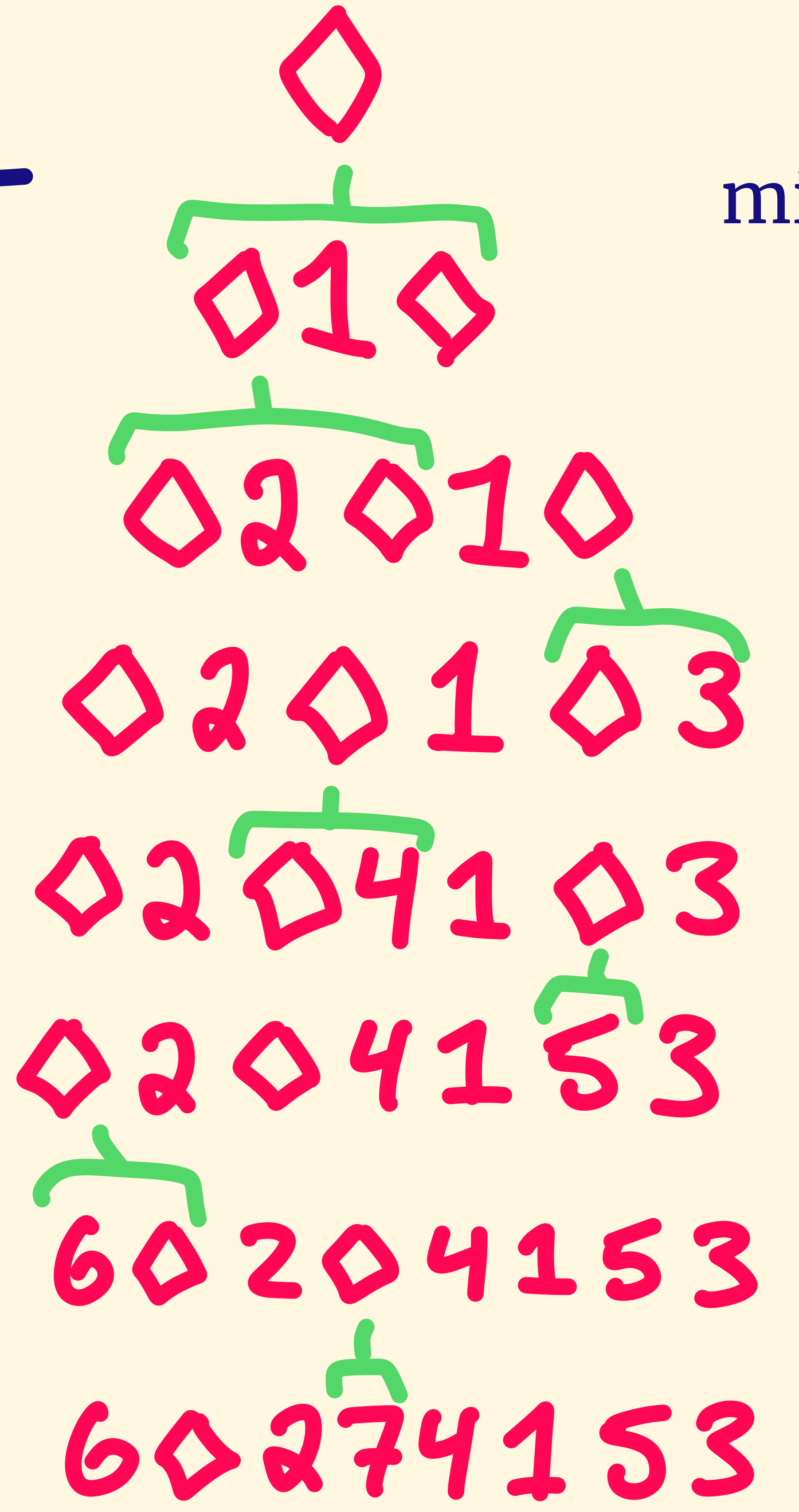
left placement

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

middle placement

right placement

right placement

fill

left placement

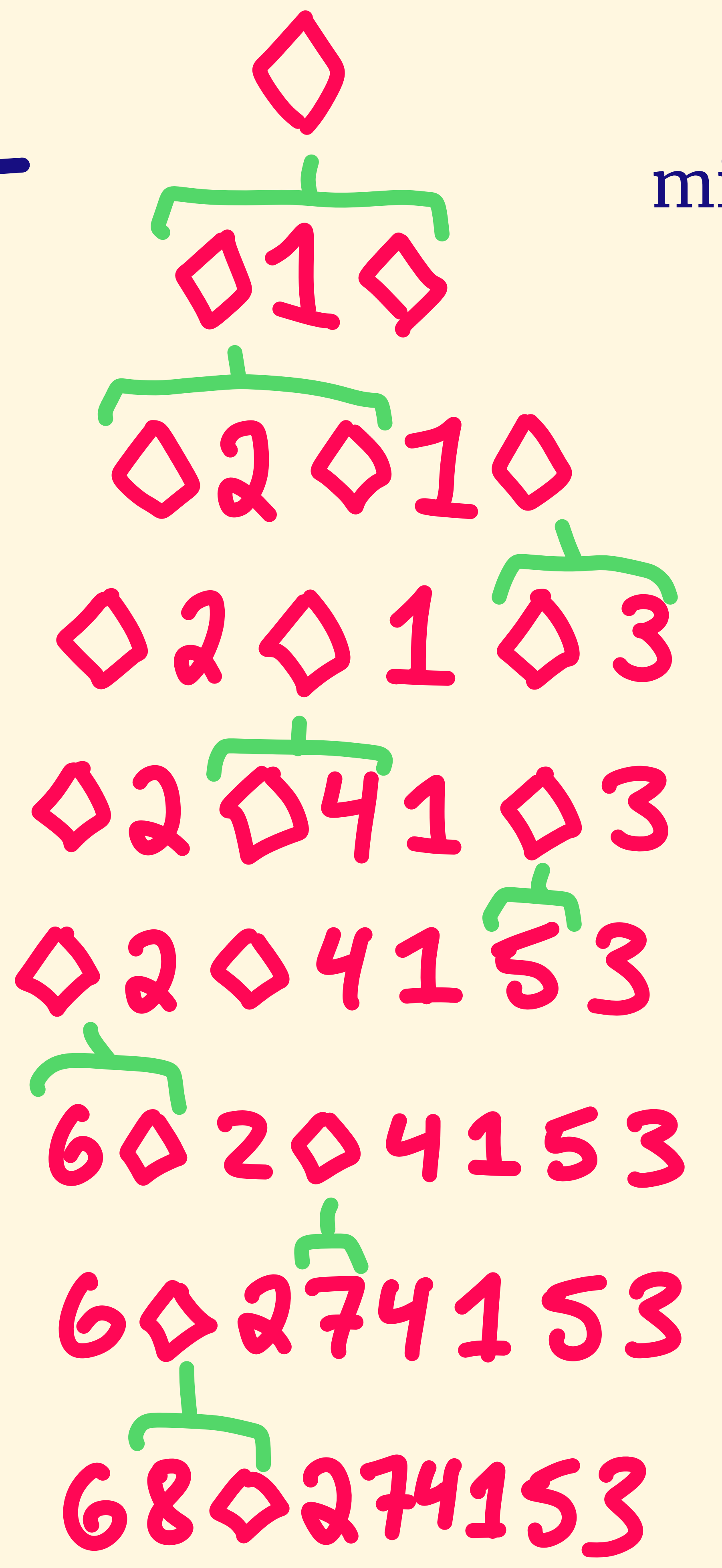
fill

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

middle placement

right placement

right placement

fill

left placement

fill

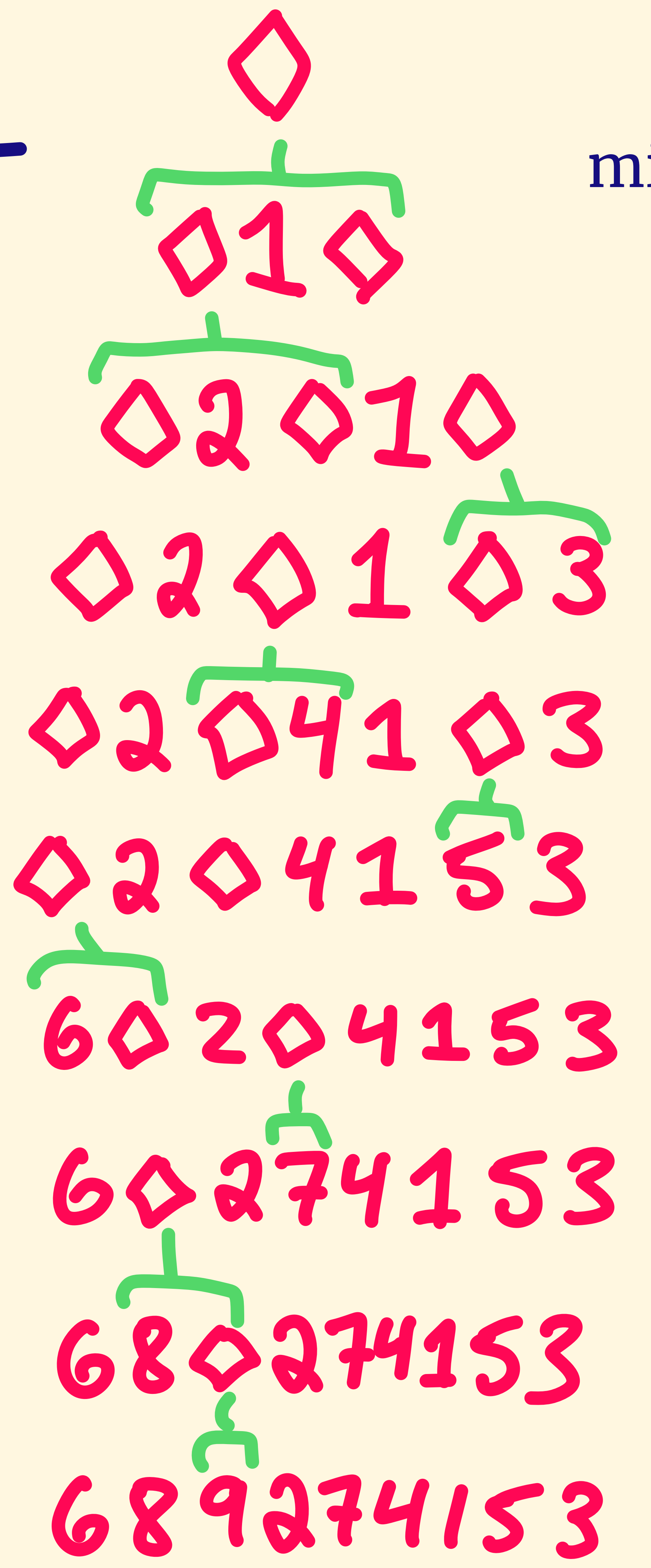
left placement

Insertion Encoding

(The Insertion Encoding of Permutations, Albert, Linton, and Ruškuc, 2005)

Encodes how a permutation can be built from bottom to top.

689274153



middle placement

middle placement

right placement

right placement

fill

left placement

fill

left placement

fill

Insertion Encoding

Let B be a set of permutations.

Define Av(B) to be the set of all permutations that avoid as patterns all of the permutations in B.

6427153 contains 231

and avoids 123

Examples:

Av(1324)
Av(132, 231)
Av(1324, 51234, 654123)

These kinds of sets are called *permutation classes*.

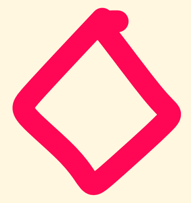
Insertion Encoding

Some permutation classes have a "finite insertion encoding" — if you write down the stages of the insertion encodings of every permutation in the class, and simplify them in certain ways, you end up with a finite set.

(Finding Regular Insertion Encodings for Permutation Classes, Vatter, 2012)

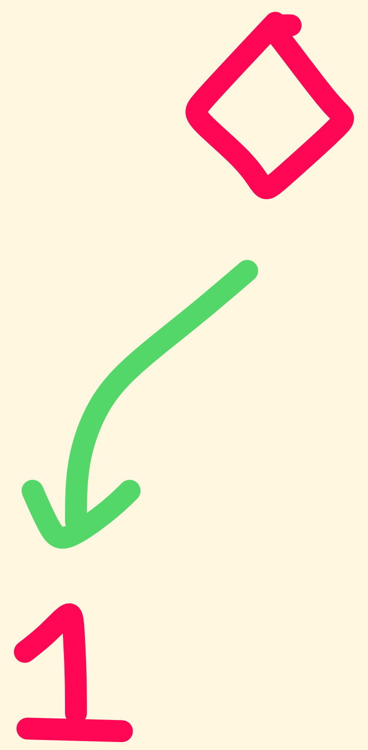
Insertion Encoding

Av(132, 231)



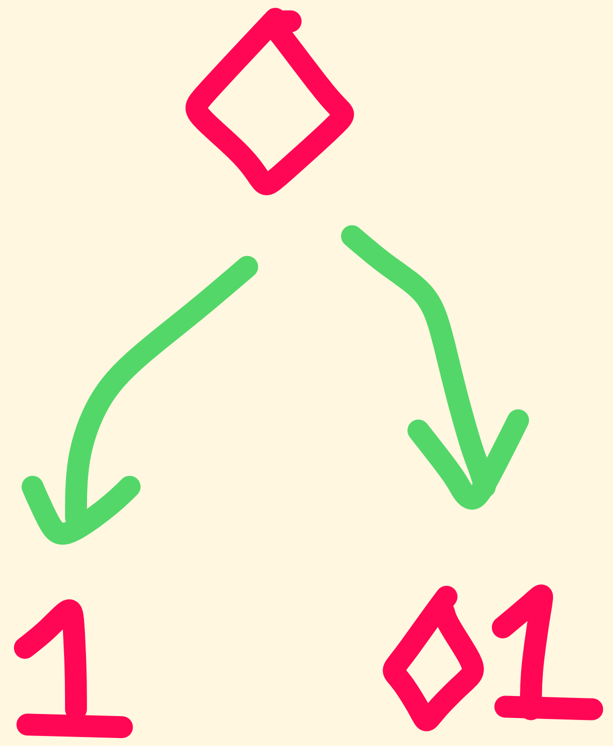
Insertion Encoding

Av(132, 231)



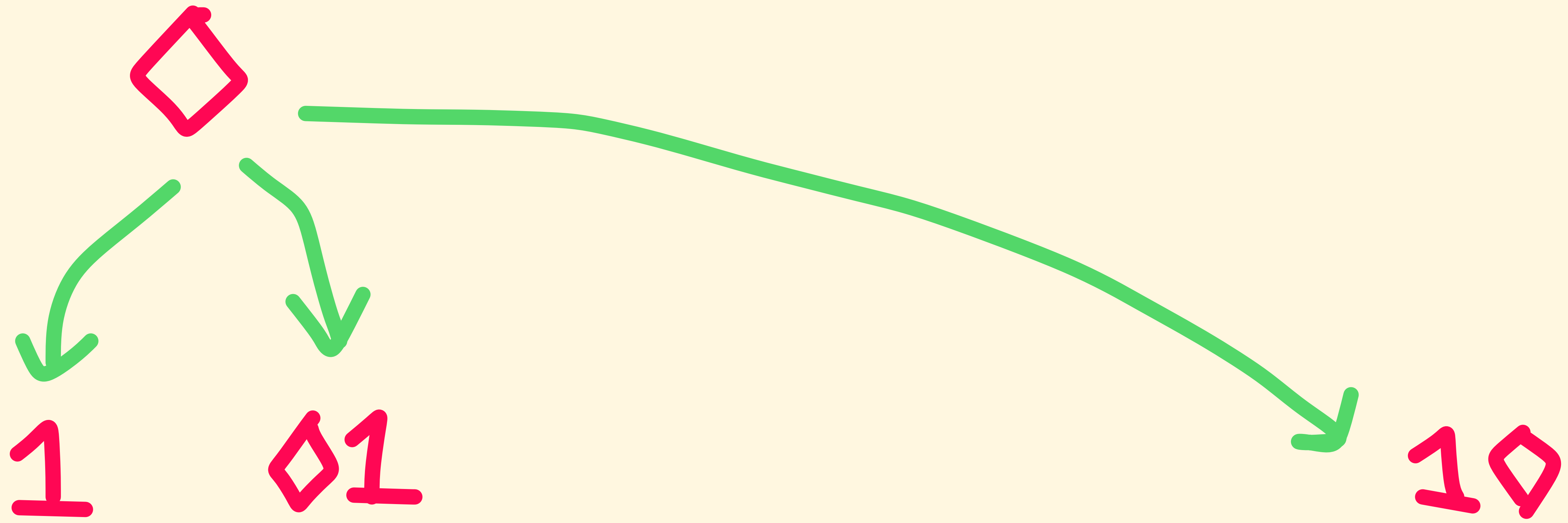
Insertion Encoding

Av(132, 231)



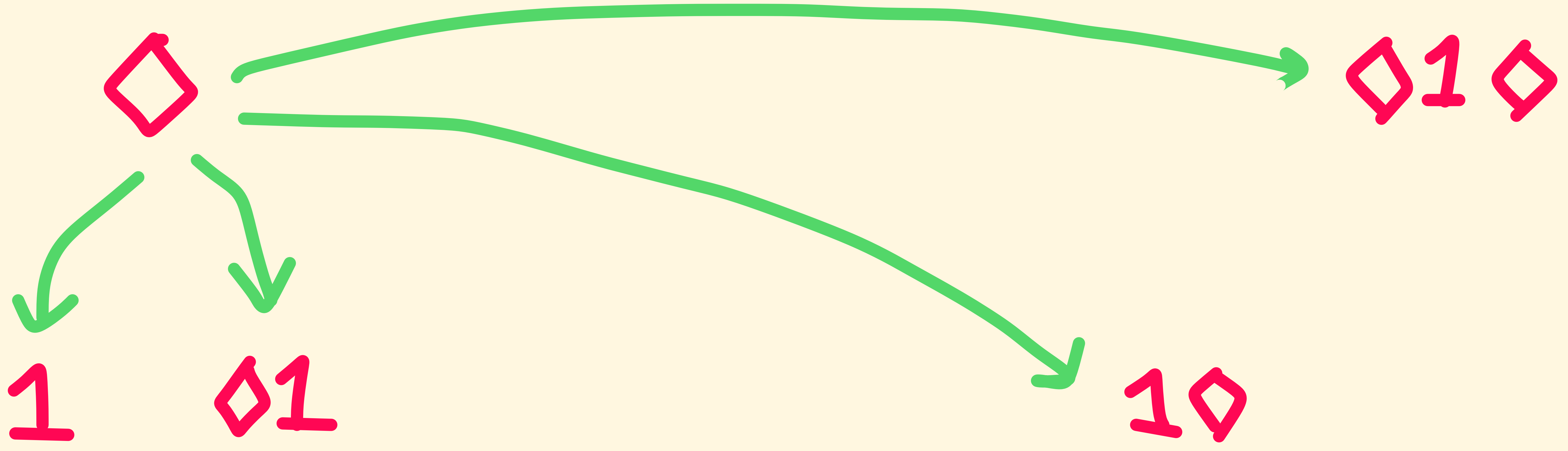
Insertion Encoding

Av(132, 231)



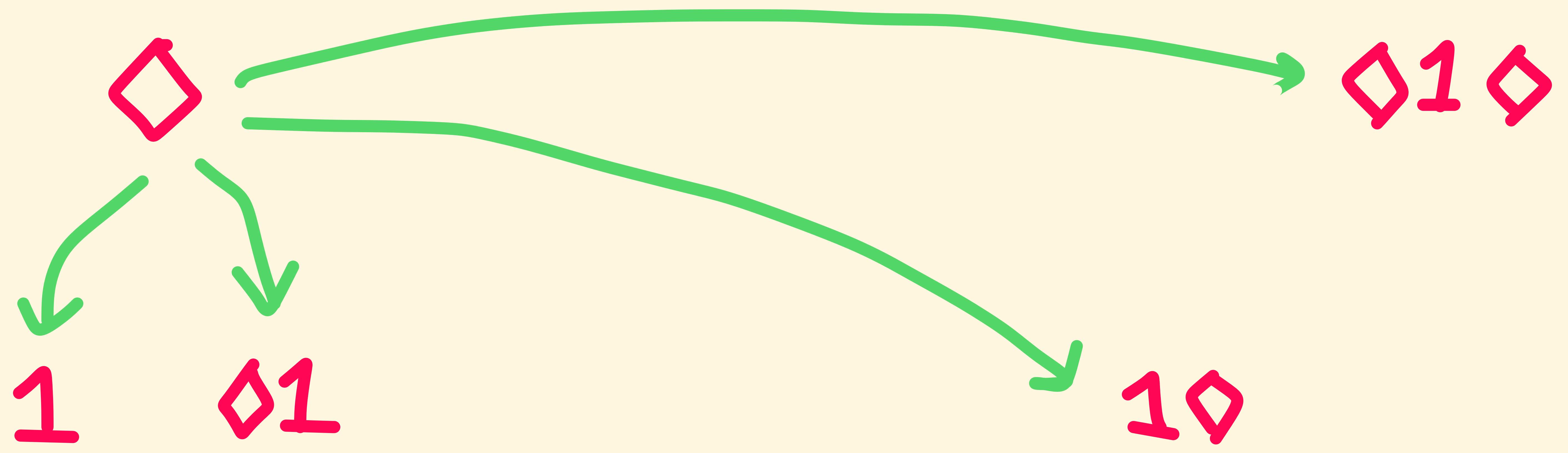
Insertion Encoding

Av(132, 231)



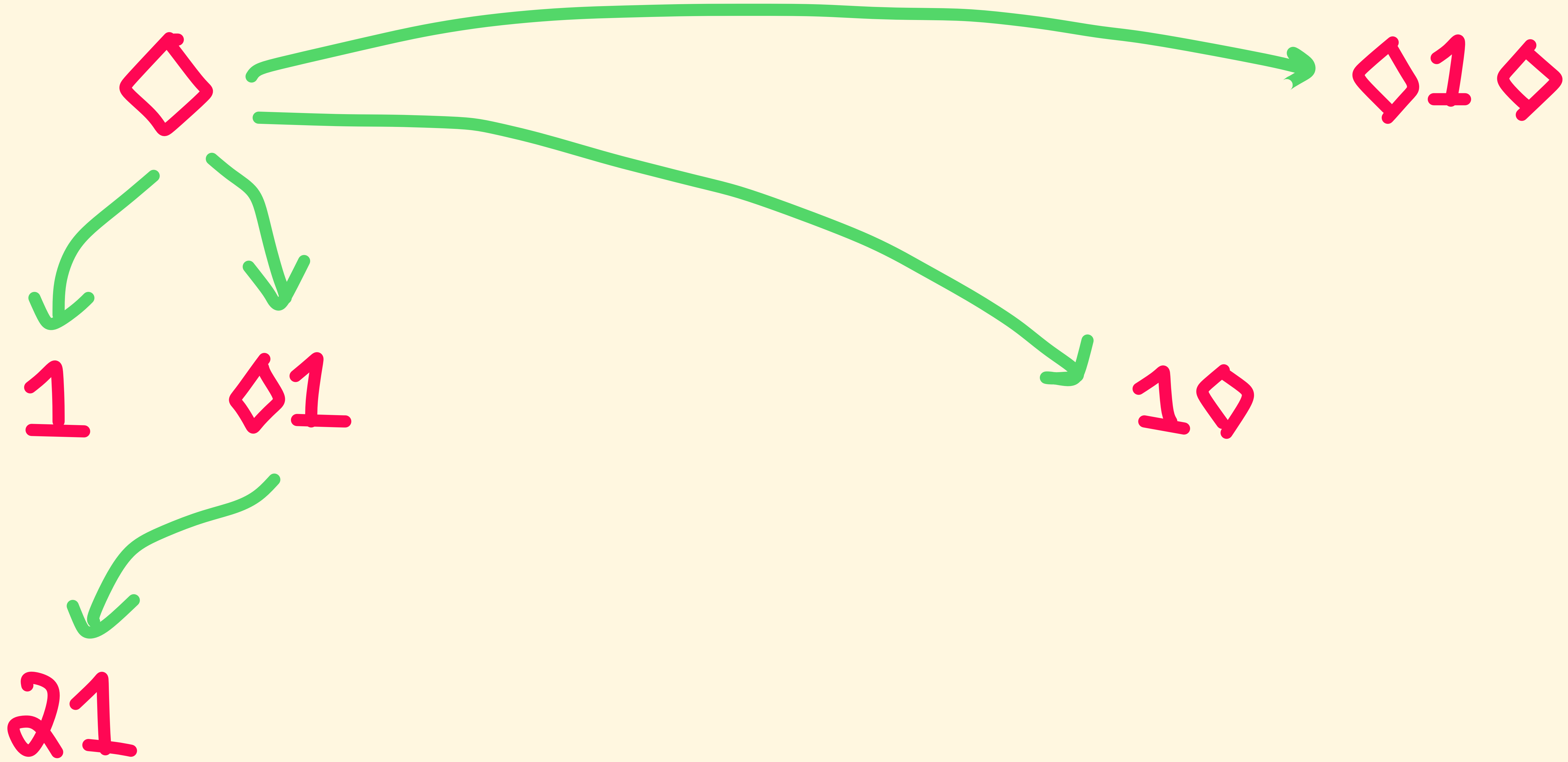
Insertion Encoding

Av(132, 231)



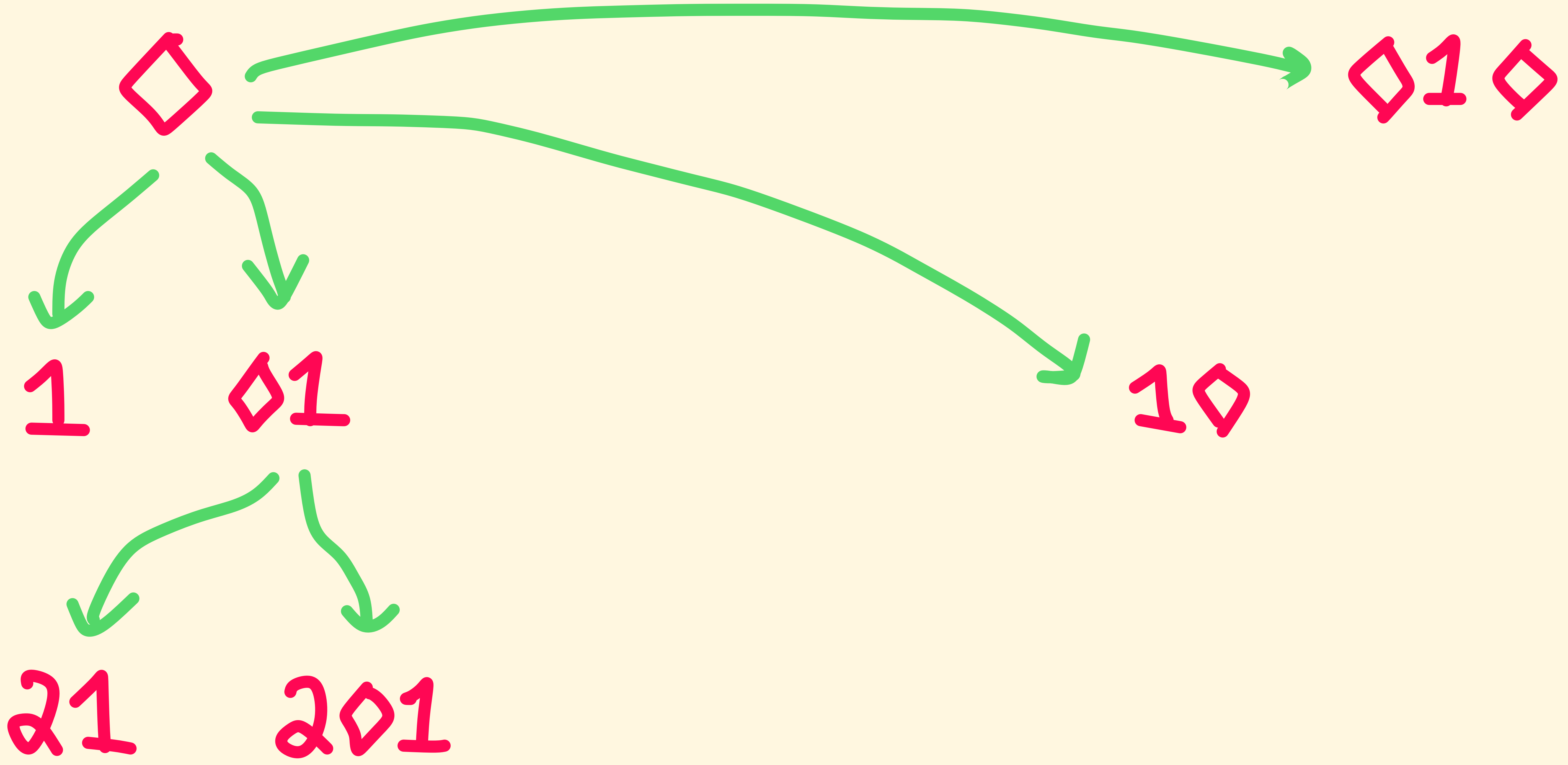
Insertion Encoding

Av(132, 231)



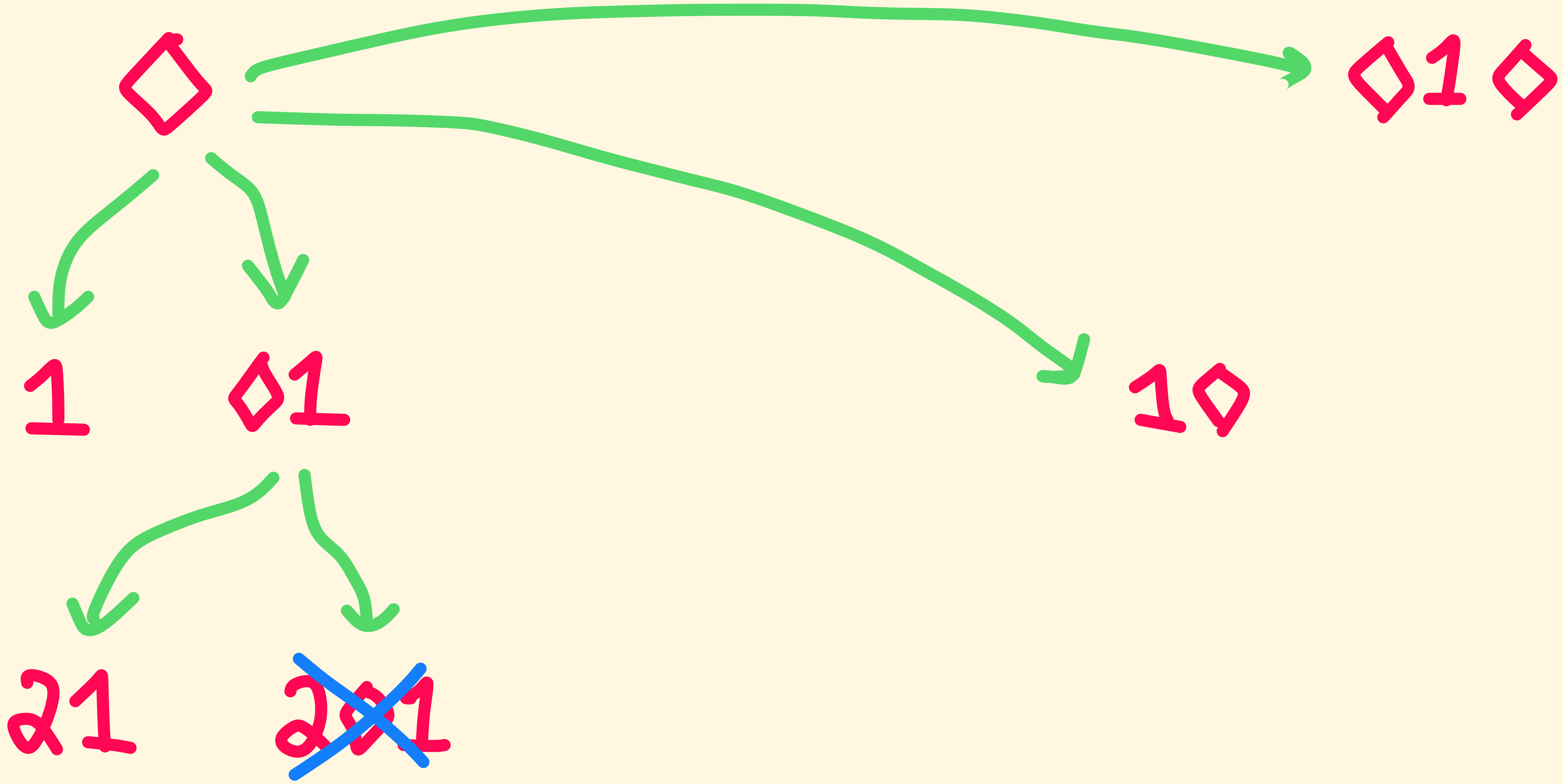
Insertion Encoding

Av(132, 231)



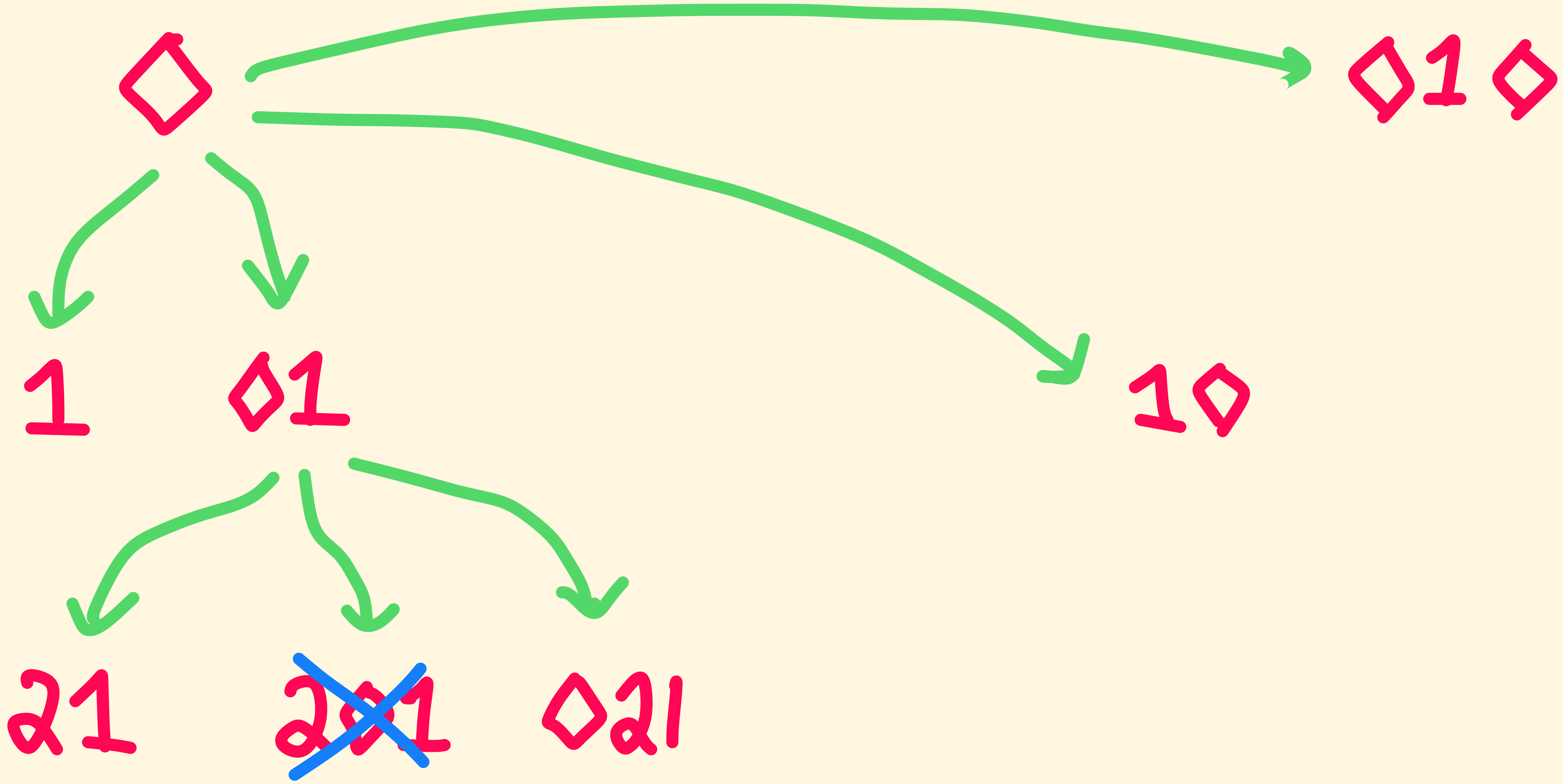
Insertion Encoding

Av(132, 231)



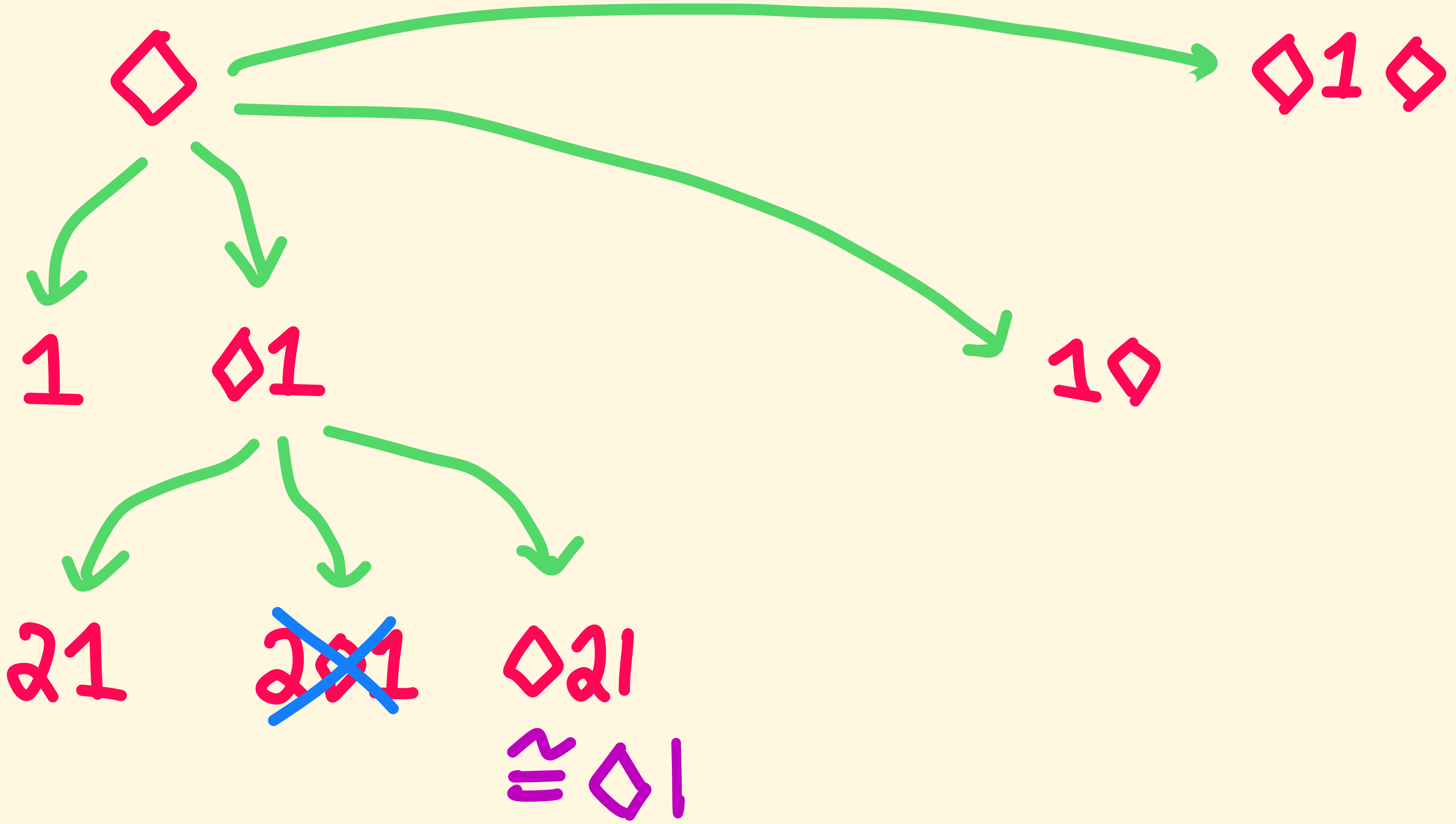
Insertion Encoding

Av(132, 231)



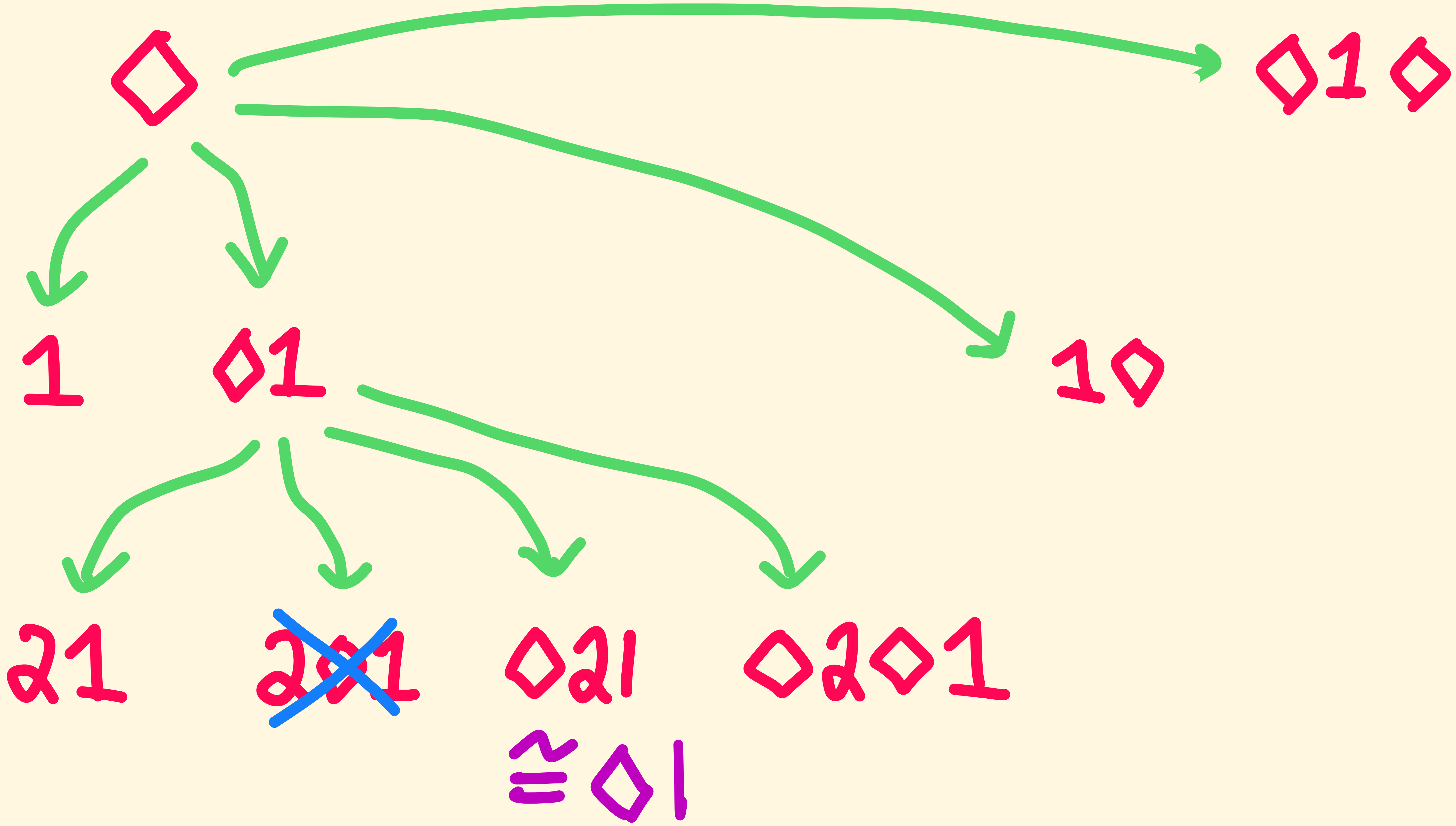
Insertion Encoding

Av(132, 231)



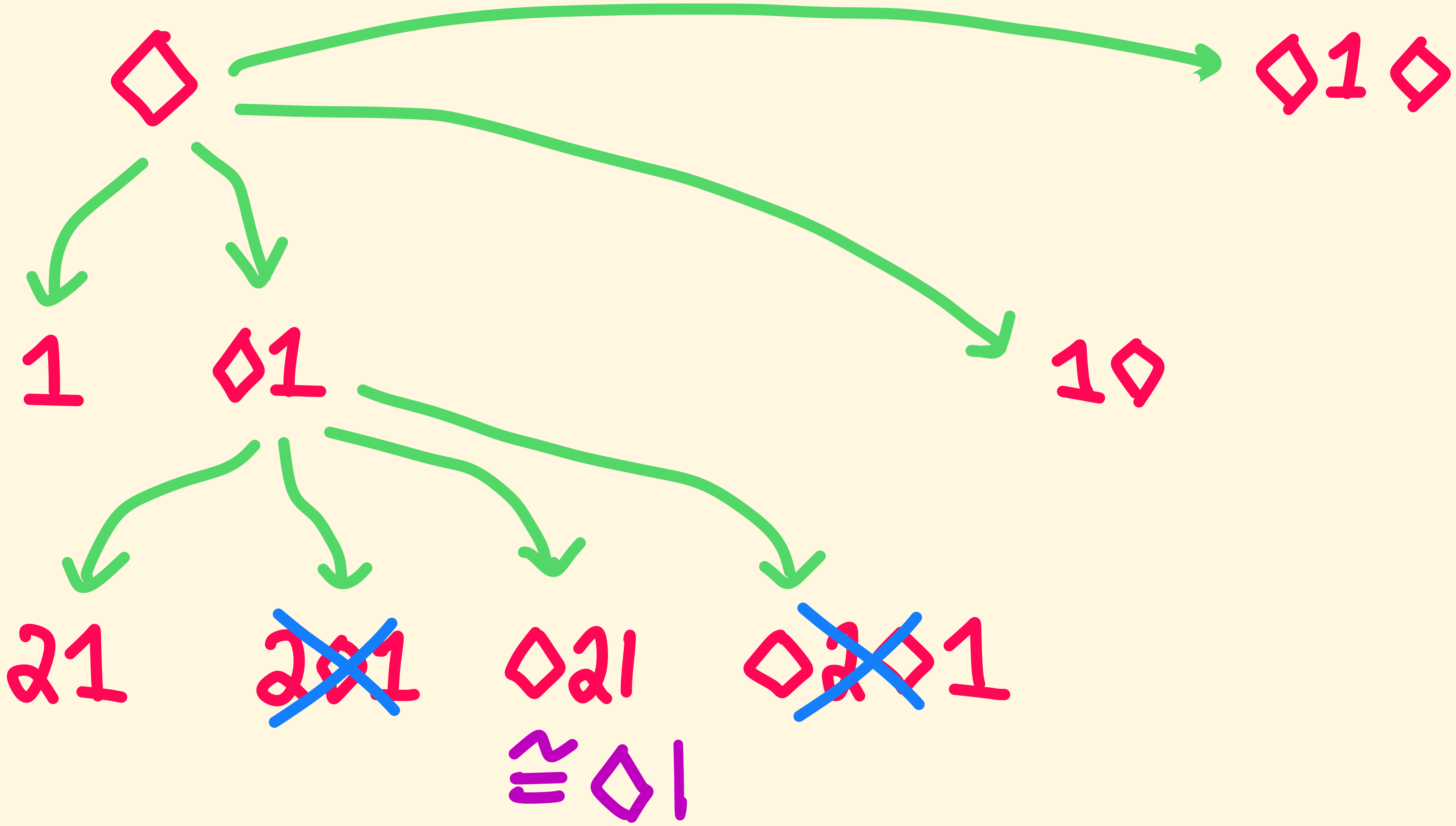
Insertion Encoding

Av(132, 231)



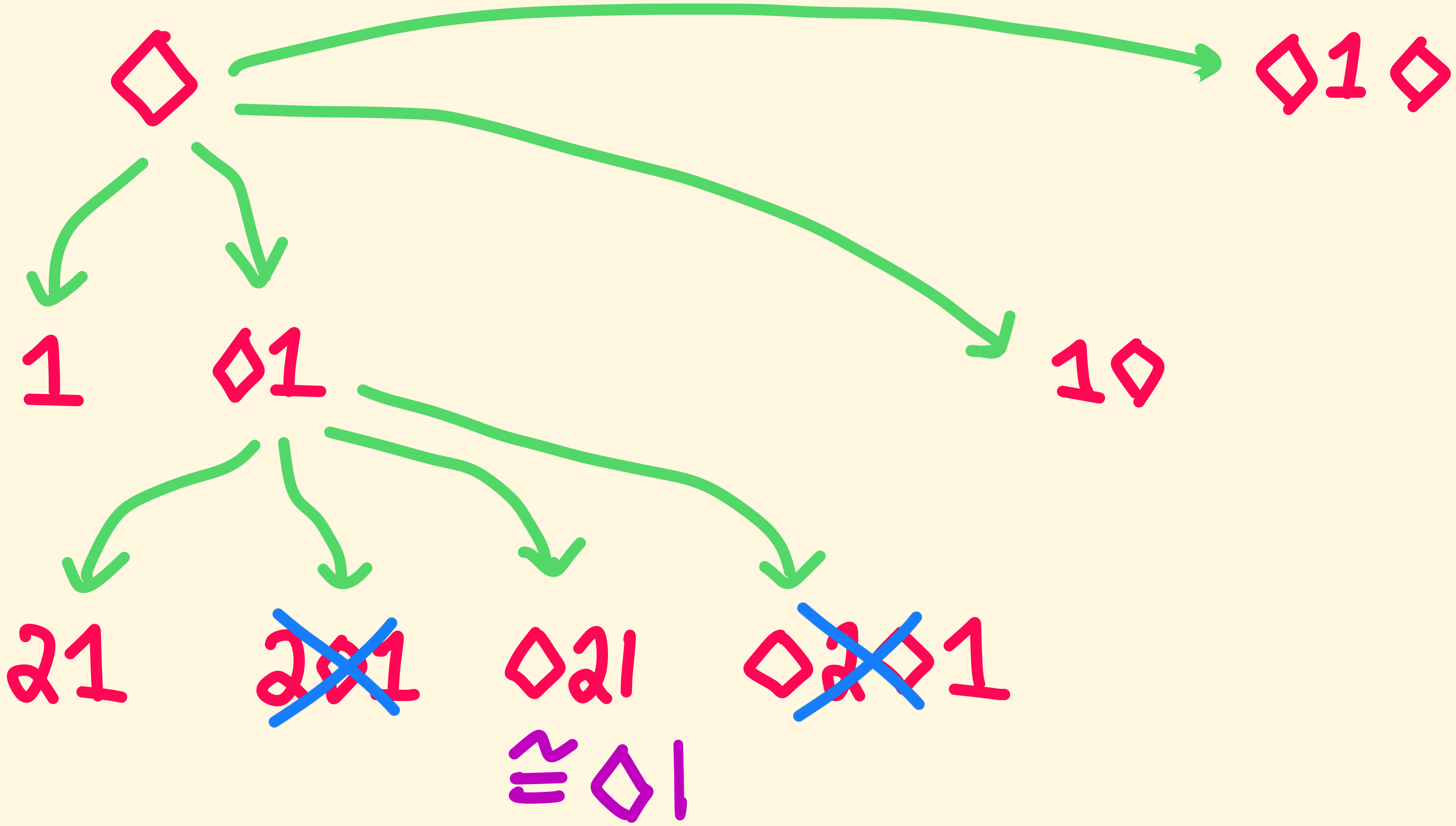
Insertion Encoding

Av(132, 231)



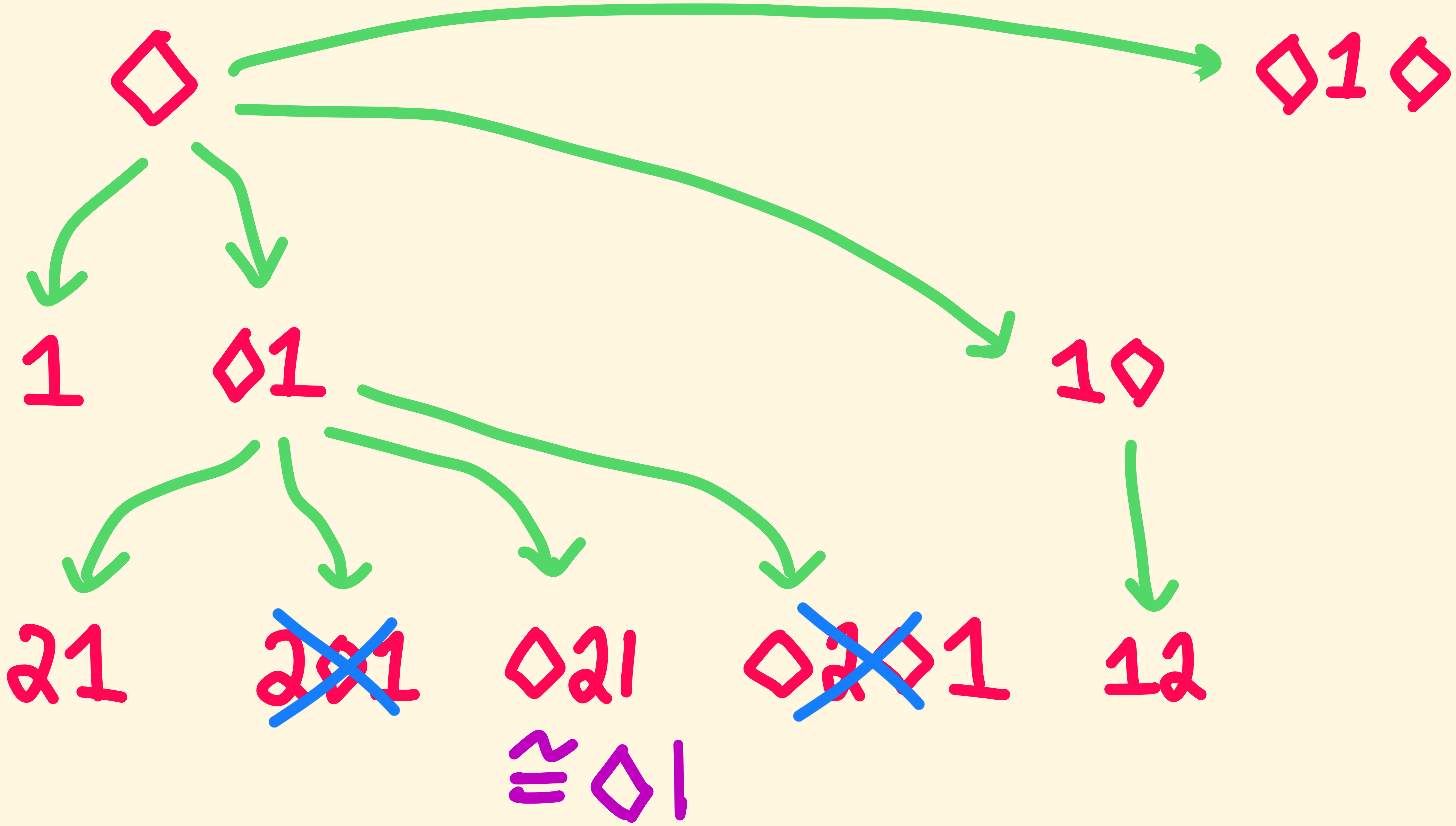
Insertion Encoding

Av(132, 231)



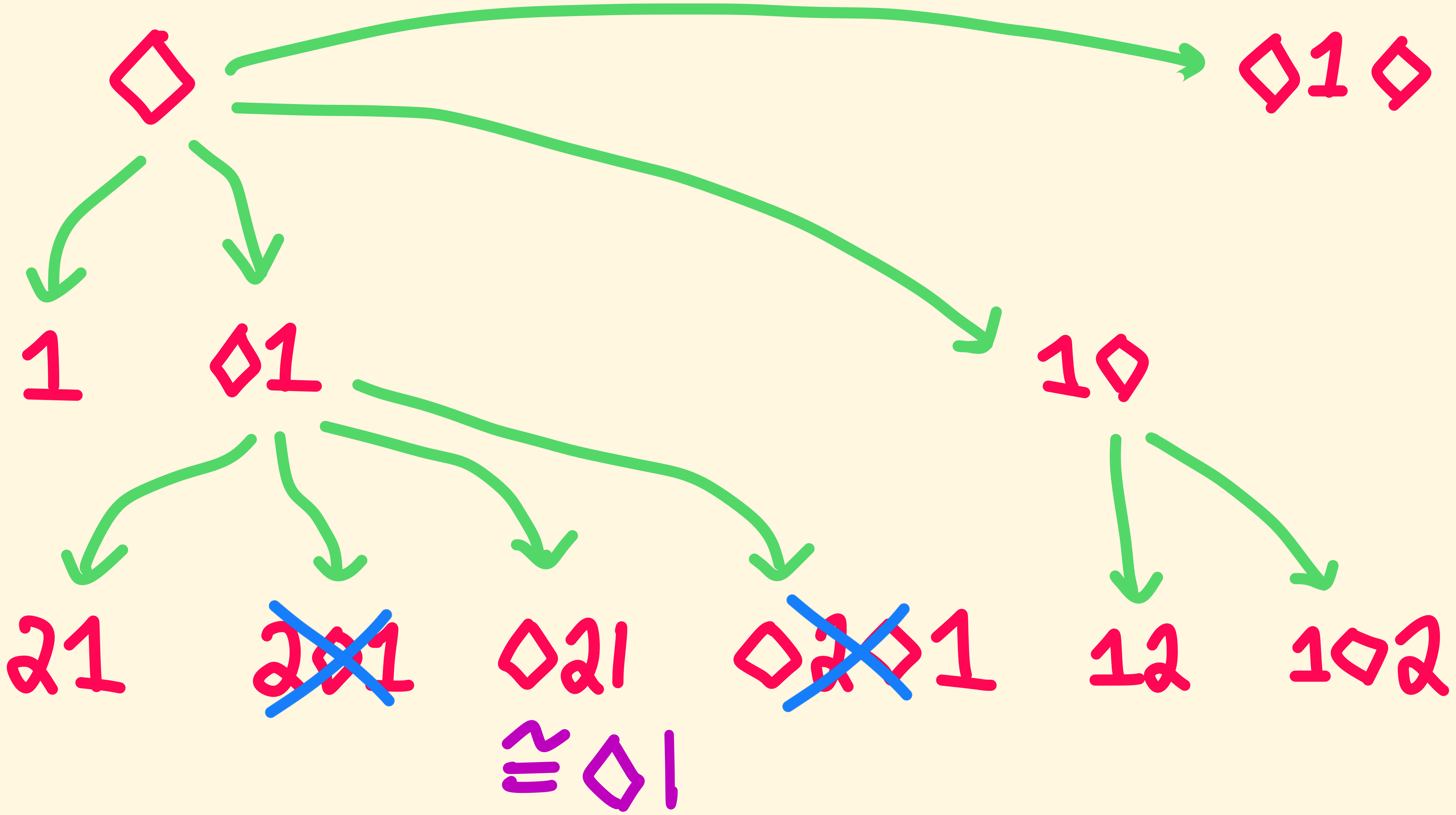
Insertion Encoding

Av(132, 231)



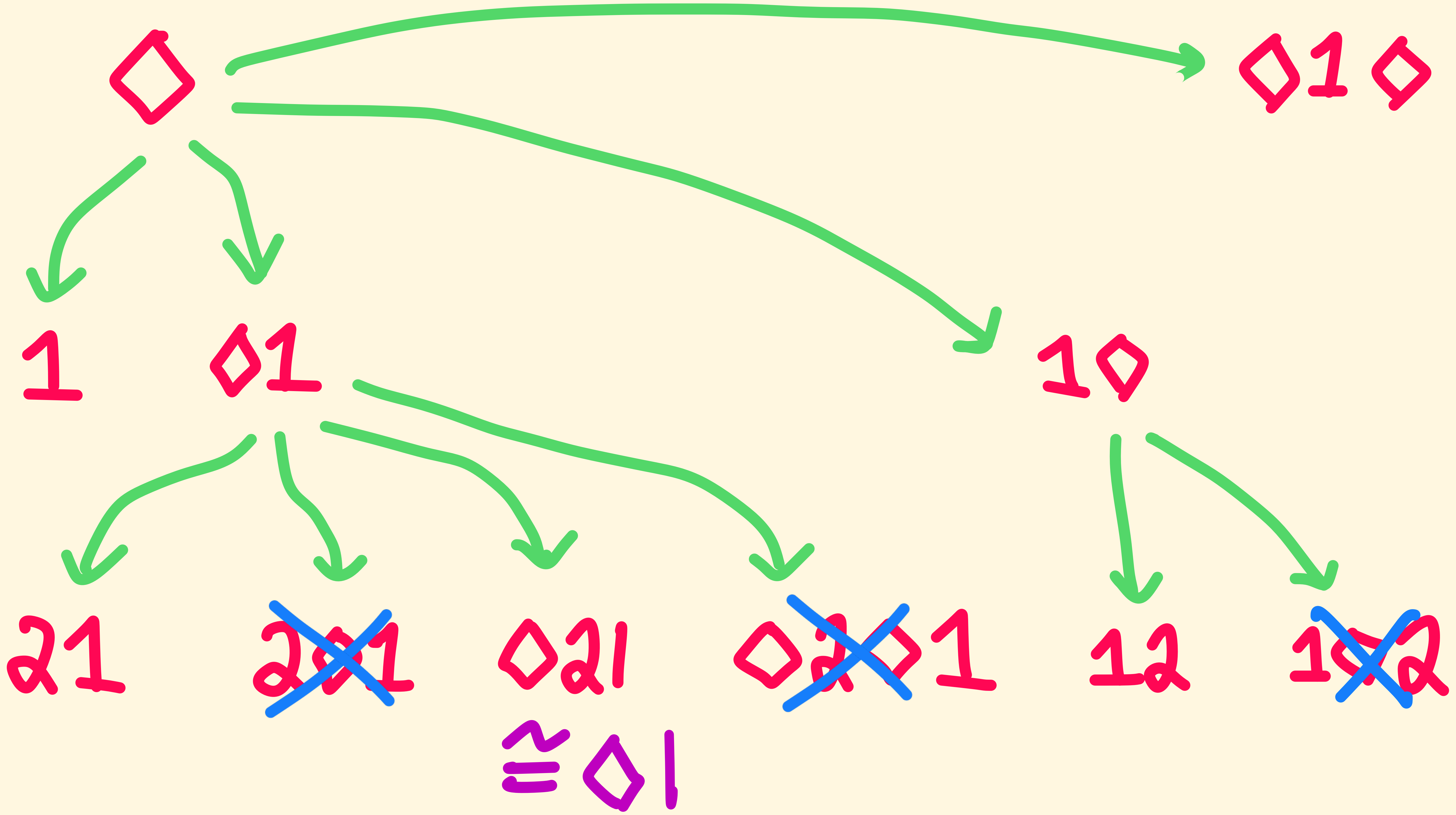
Insertion Encoding

Av(132, 231)



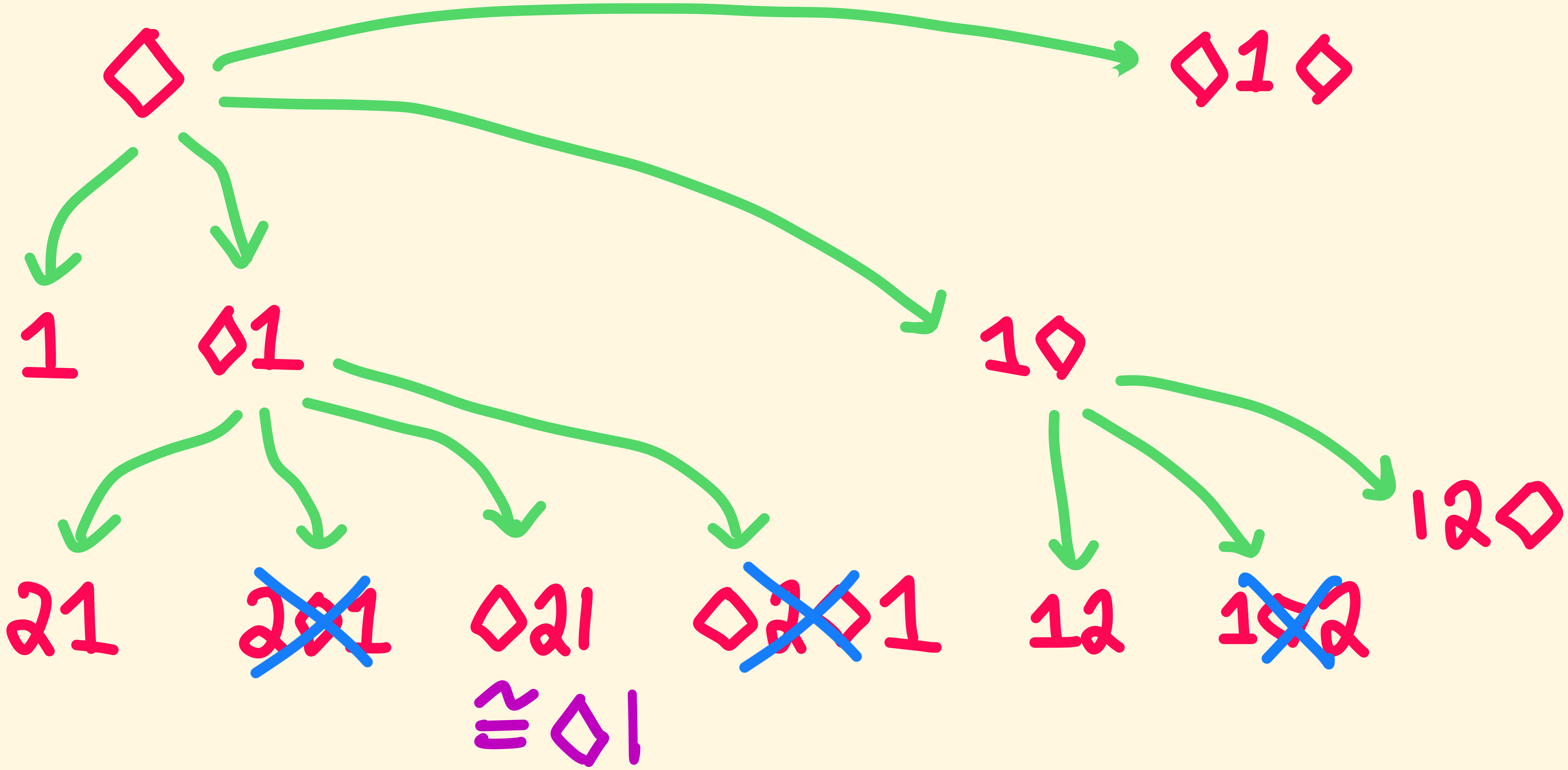
Insertion Encoding

Av(132, 231)



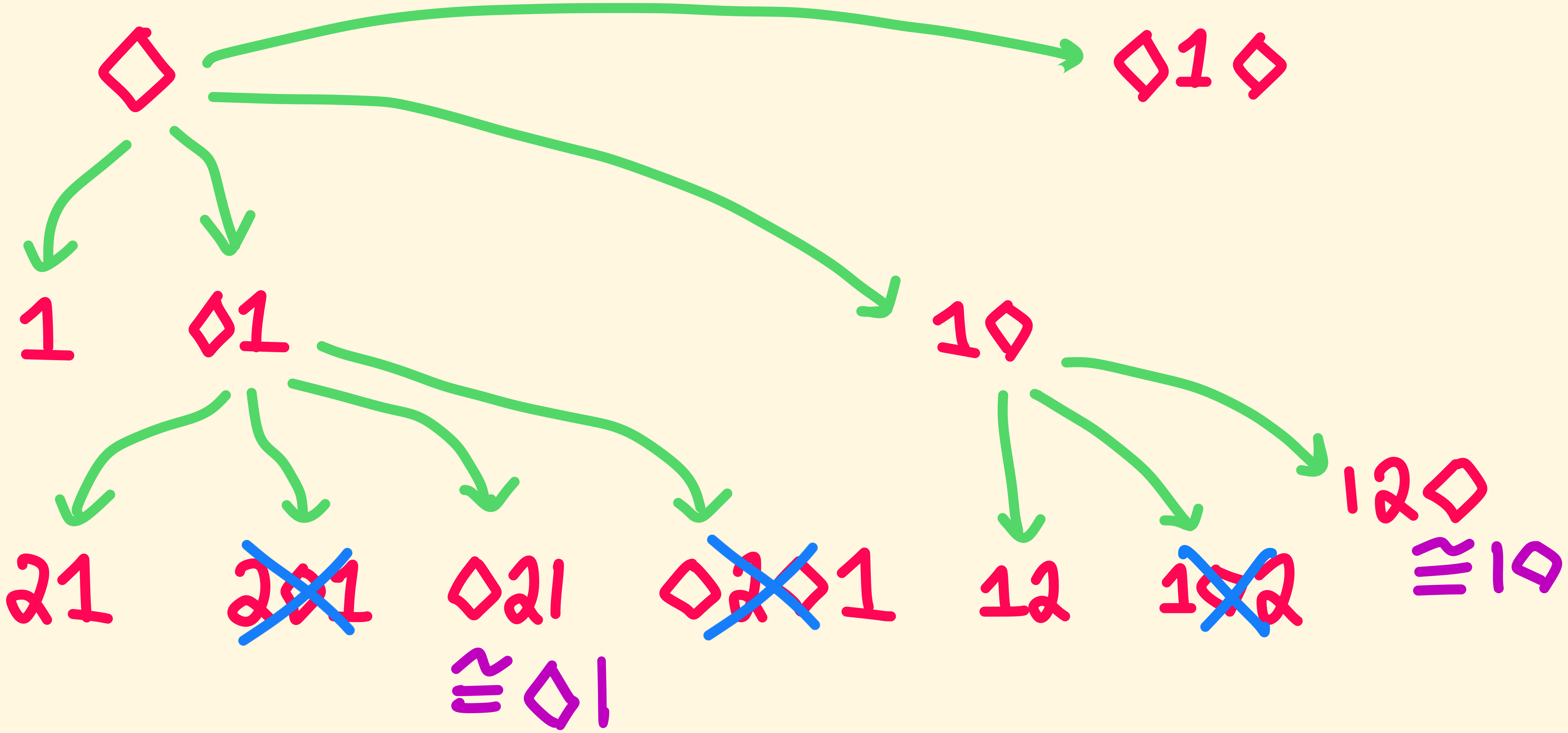
Insertion Encoding

Av(132, 231)



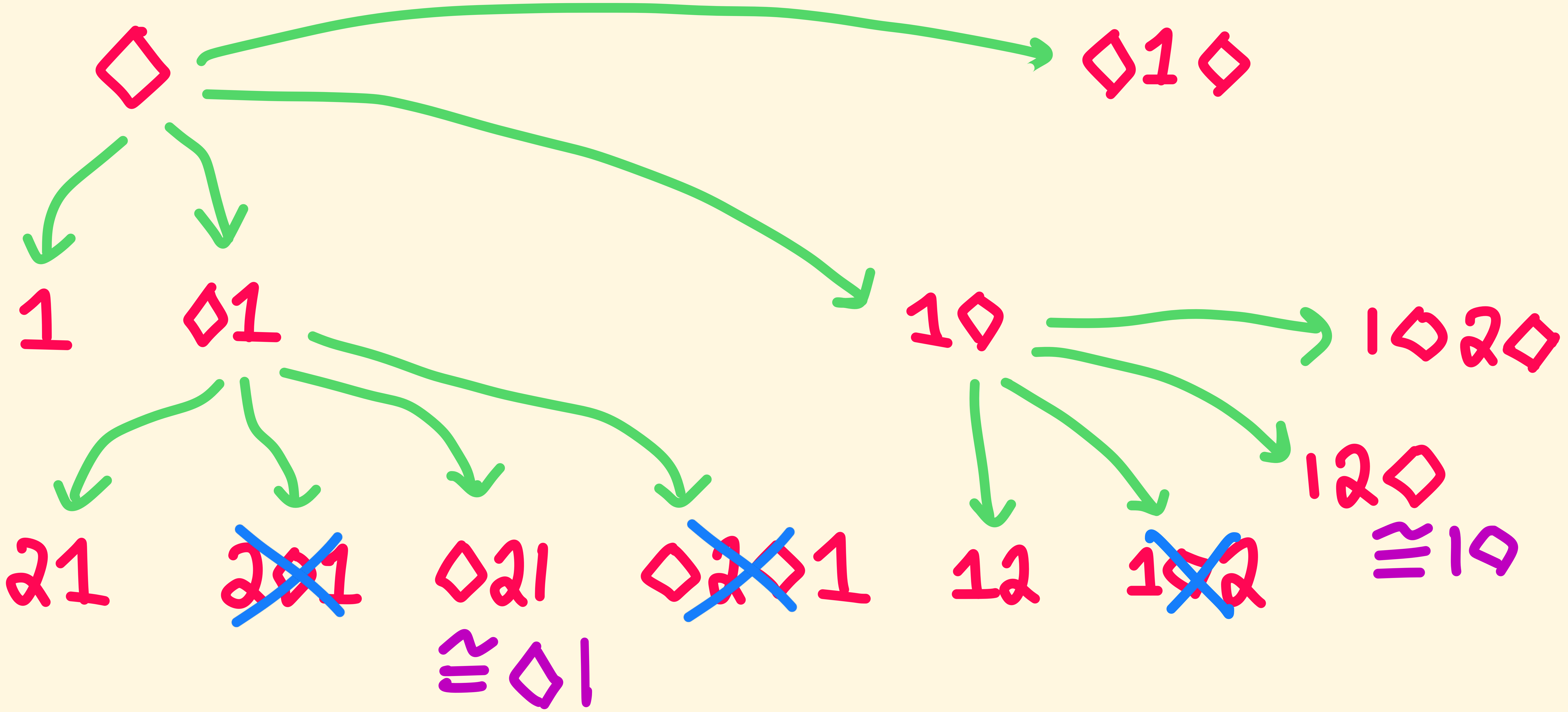
Insertion Encoding

Av(132, 231)



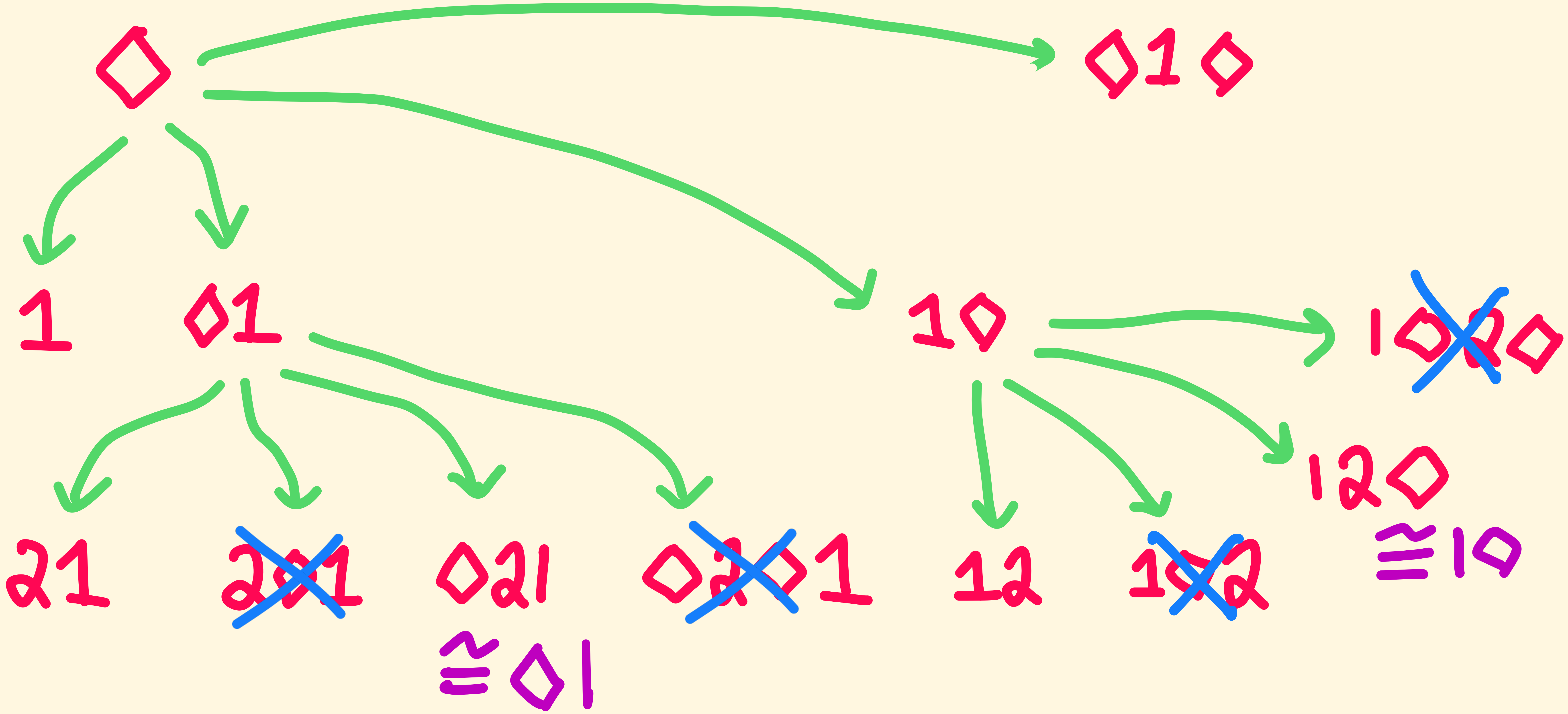
Insertion Encoding

Av(132, 231)



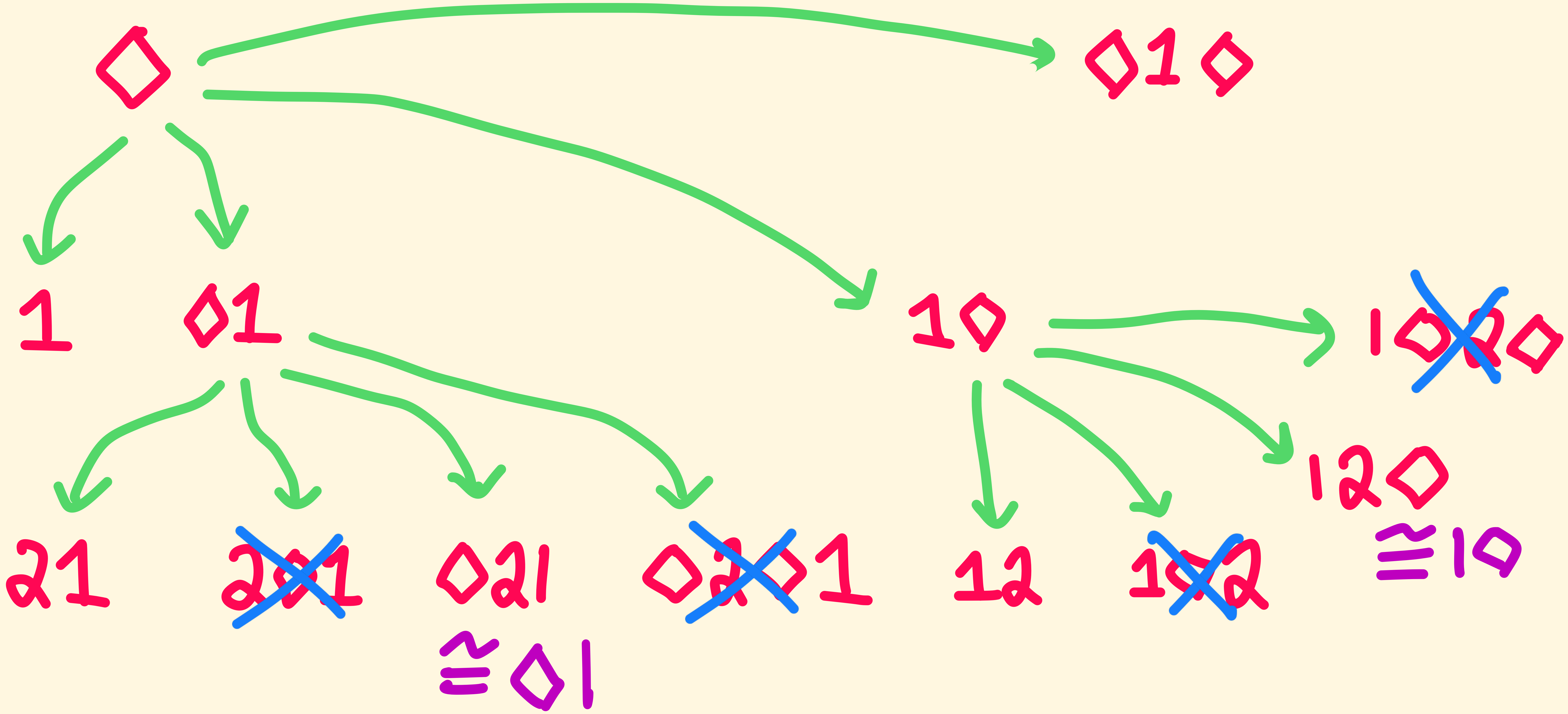
Insertion Encoding

Av(132, 231)



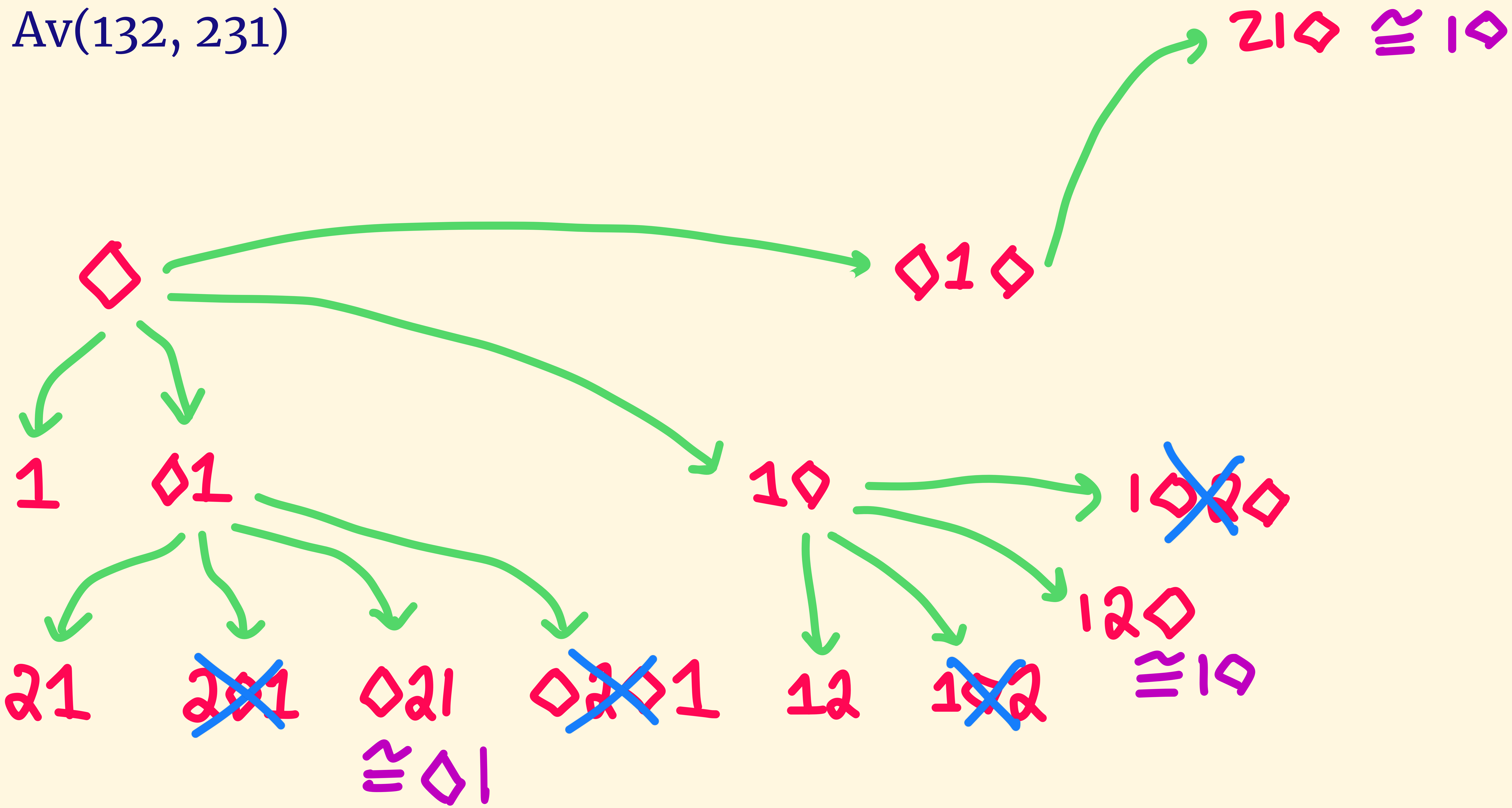
Insertion Encoding

Av(132, 231)



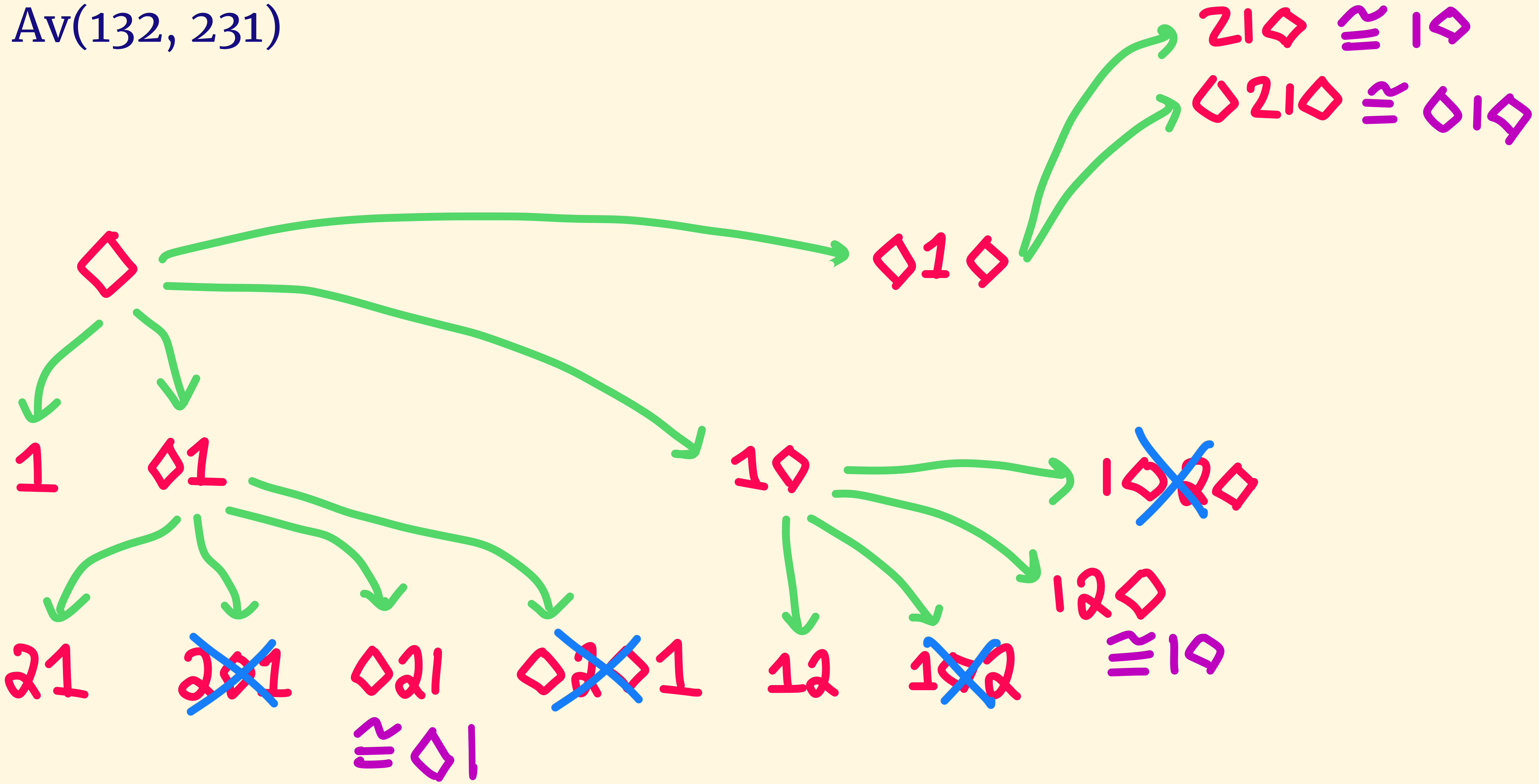
Insertion Encoding

Av(132, 231)



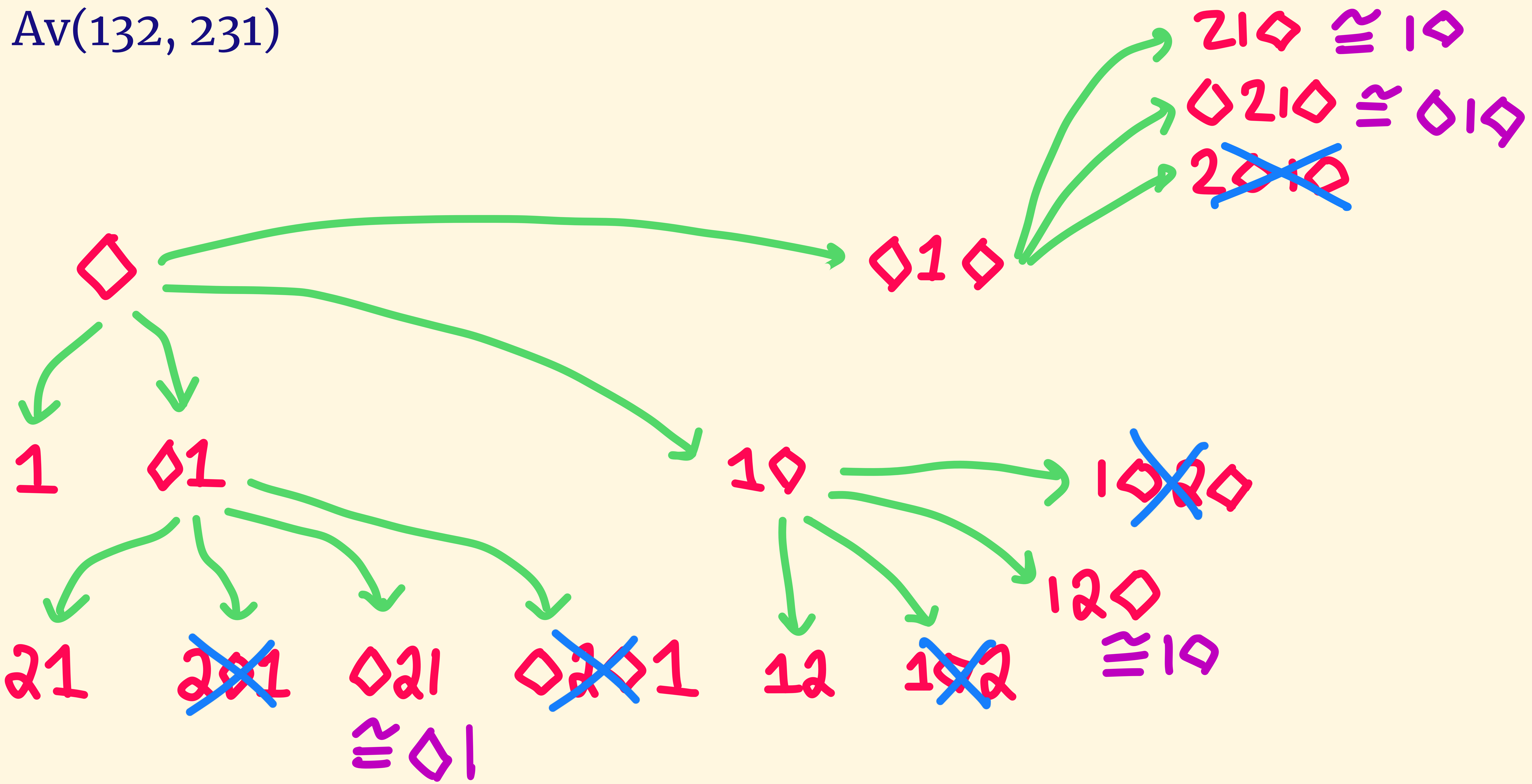
Insertion Encoding

Av(132, 231)



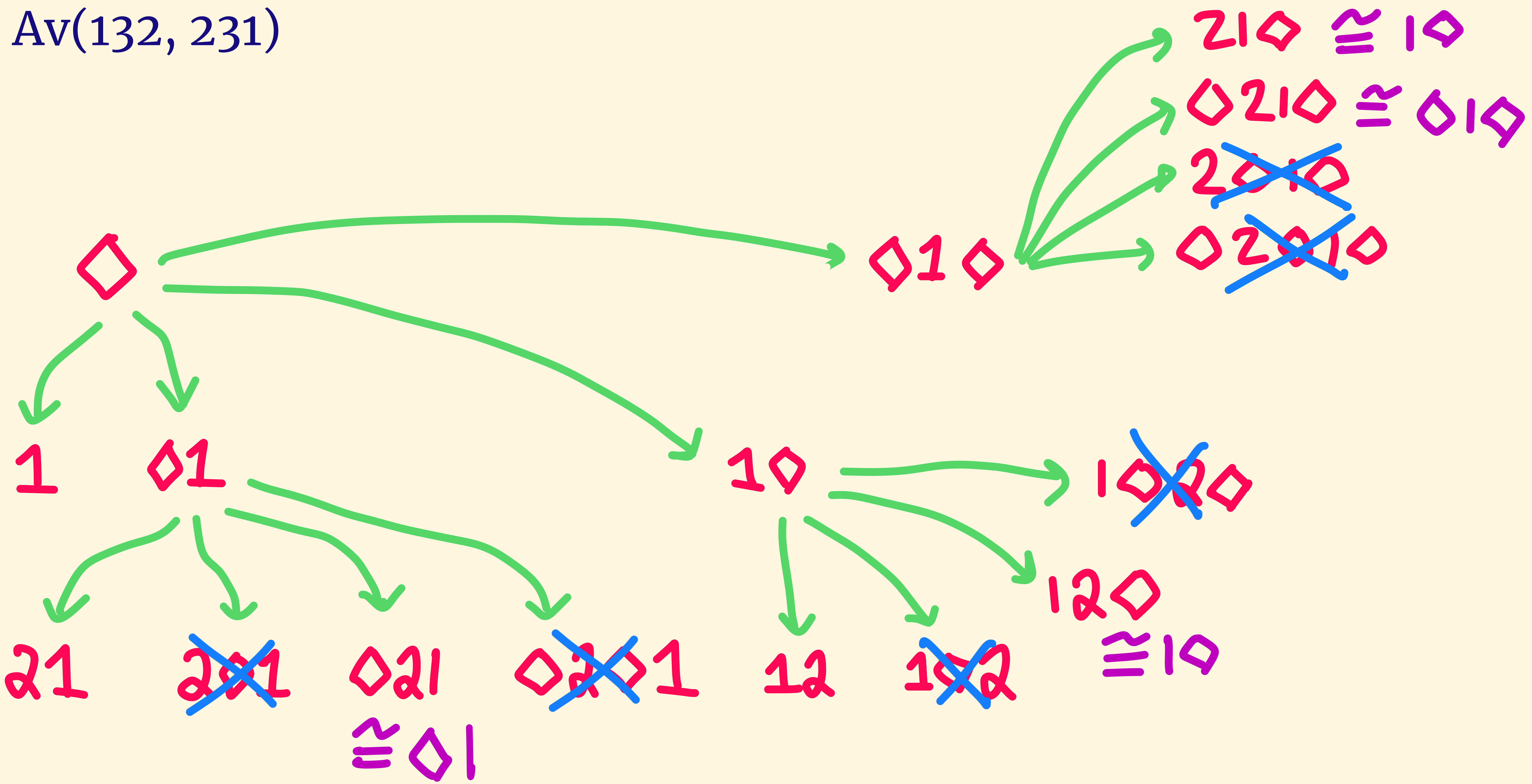
Insertion Encoding

Av(132, 231)



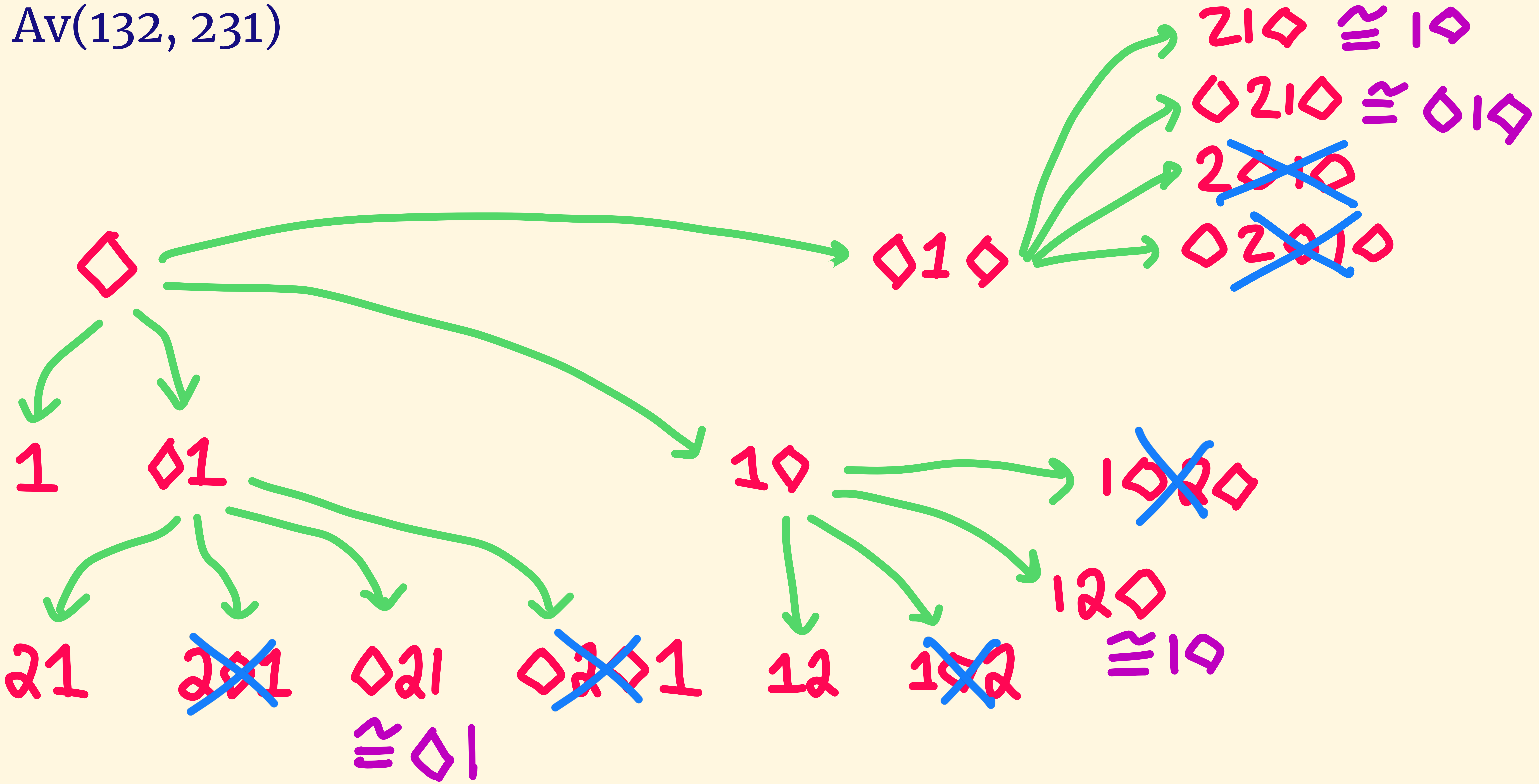
Insertion Encoding

Av(132, 231)



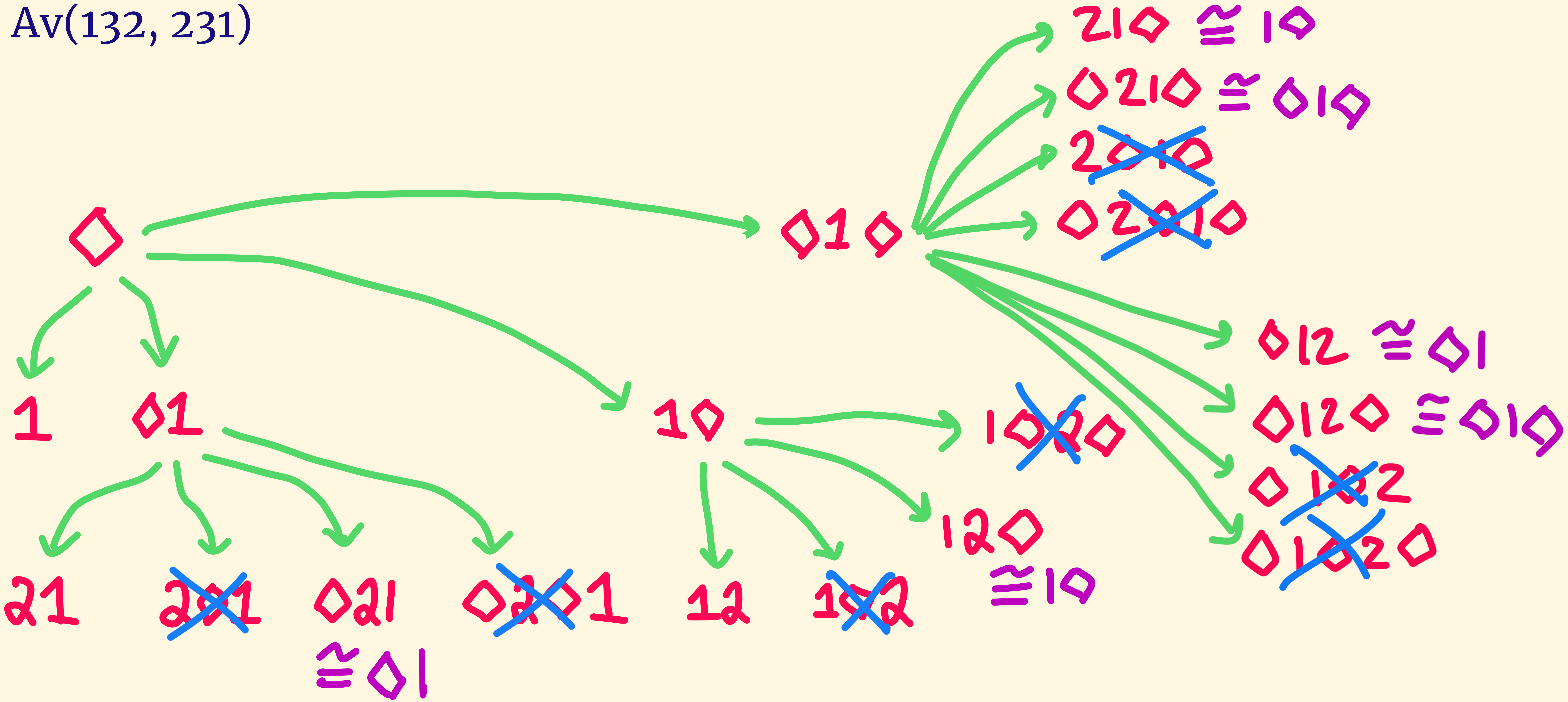
Insertion Encoding

Av(132, 231)



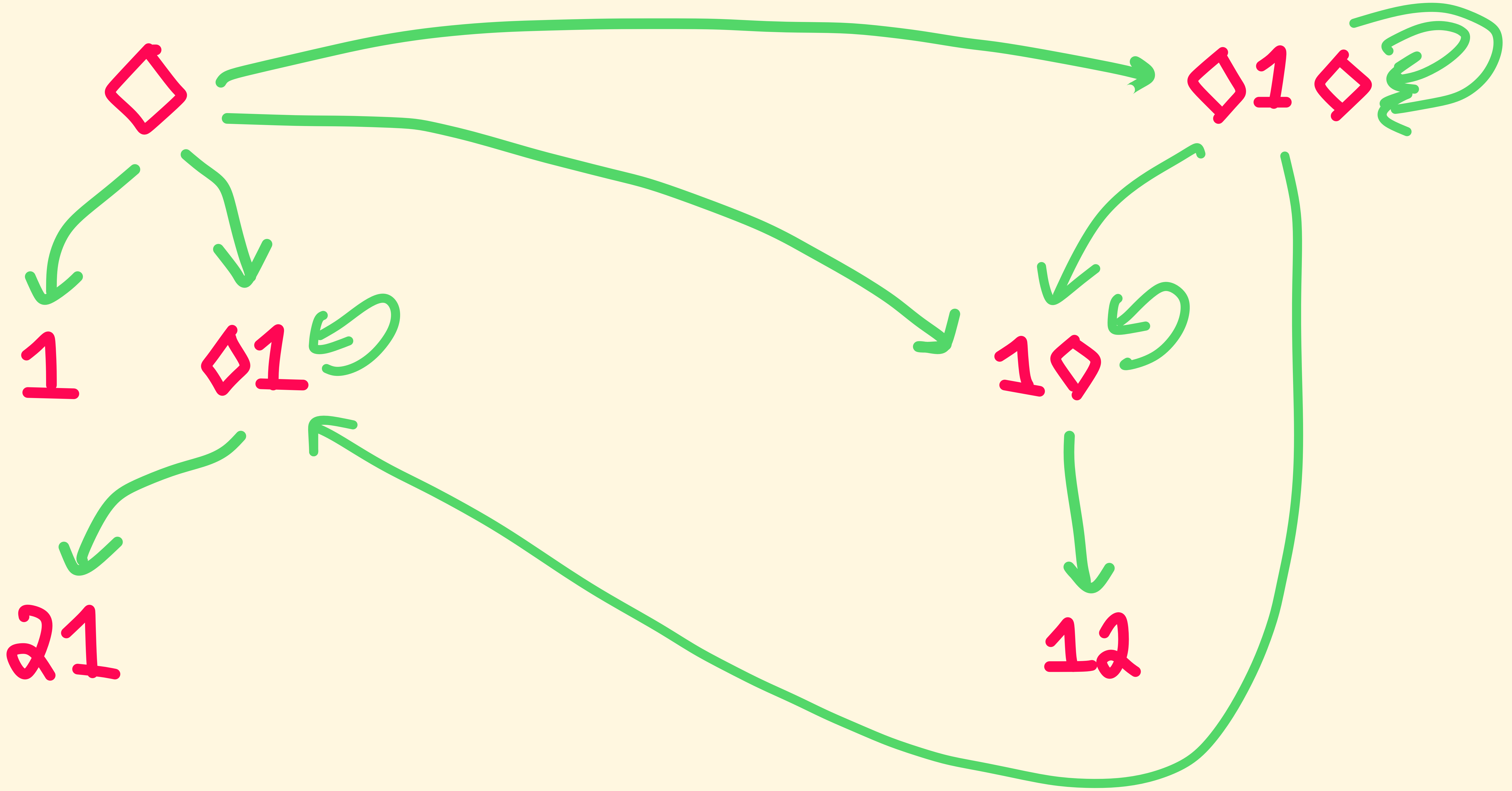
Insertion Encoding

Av(132, 231)



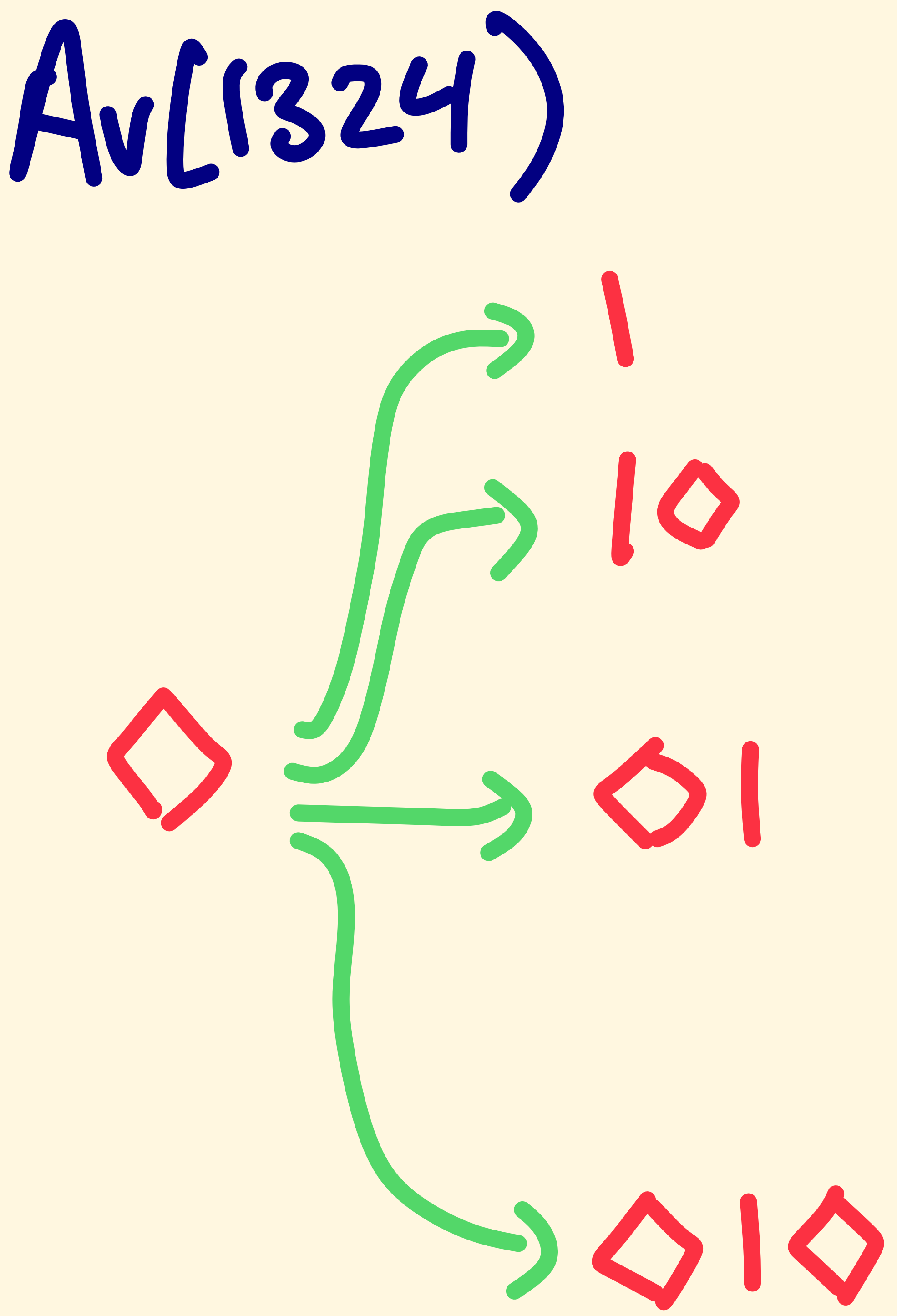
Insertion Encoding

Av(132, 231)



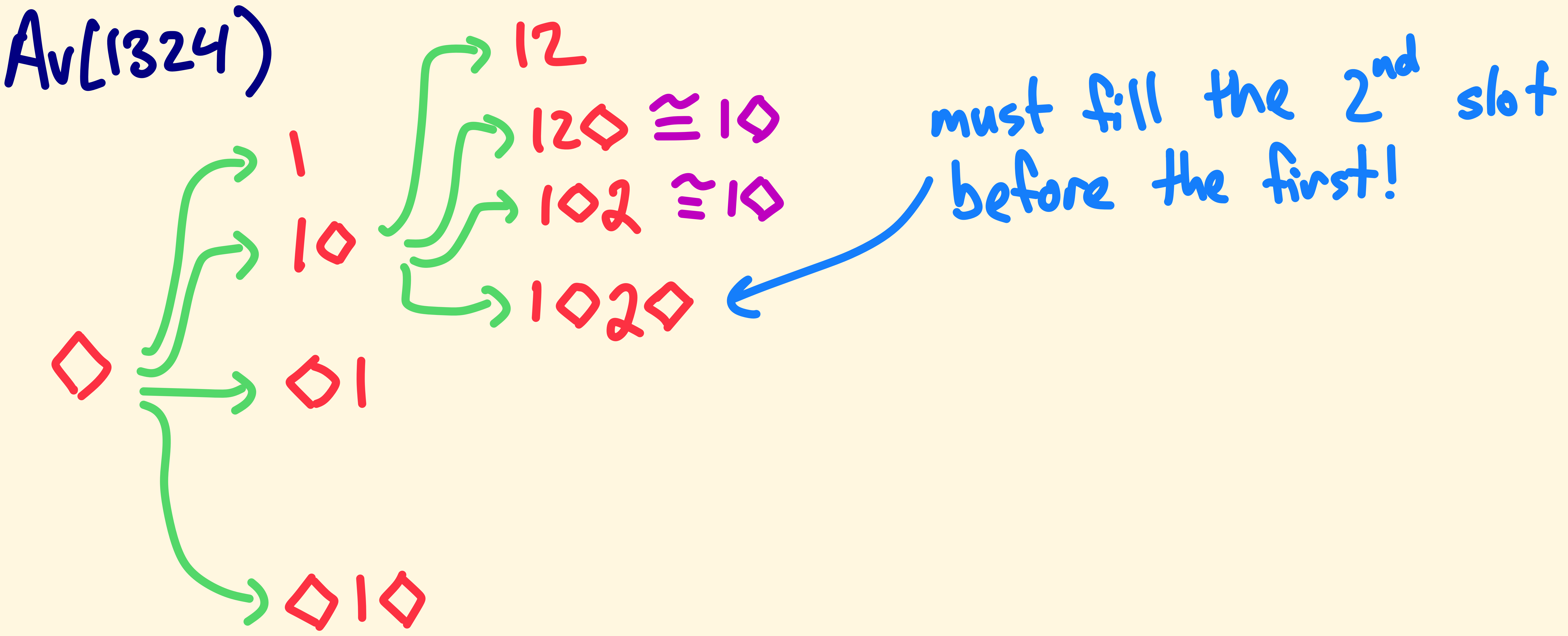
Insertion Encoding

Even if a class has an infinite insertion encoding, you can still use it to count the number of permutations in a class up to some point.



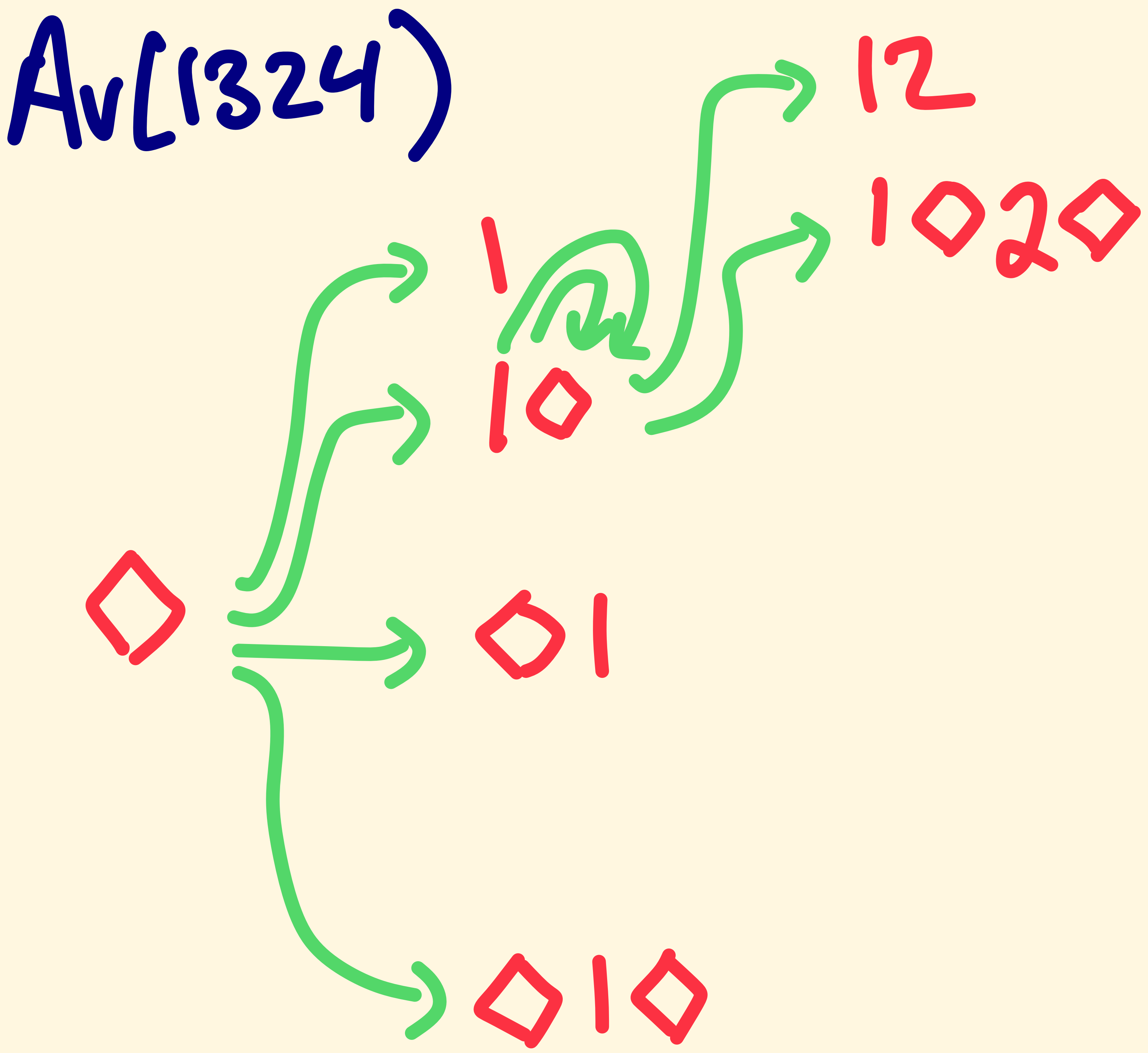
Insertion Encoding

Even if a class has an infinite insertion encoding, you can still use it to count the number of permutations in a class up to some point.



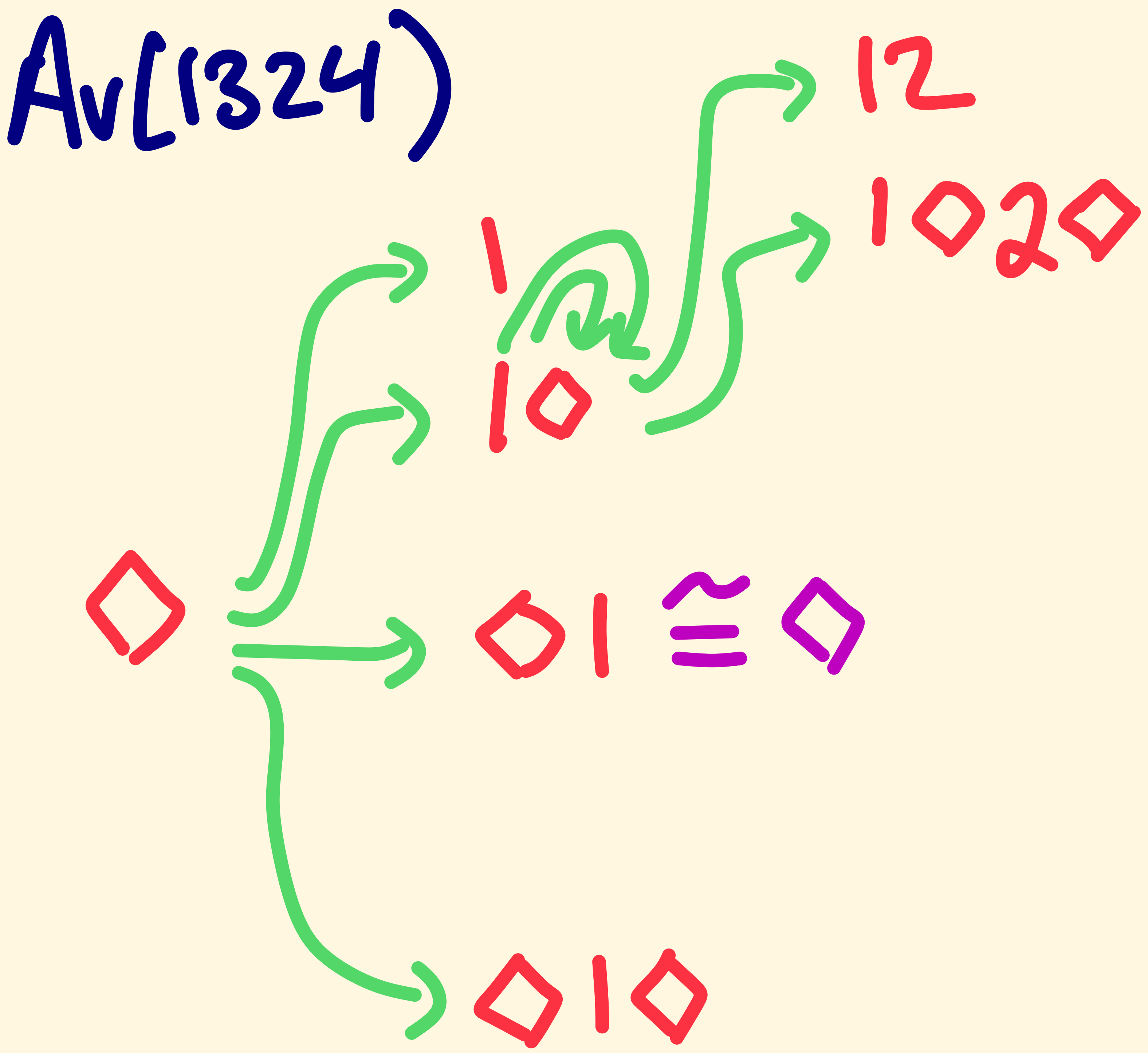
Insertion Encoding

Even if a class has an infinite insertion encoding, you can still use it to count the number of permutations in a class up to some point.



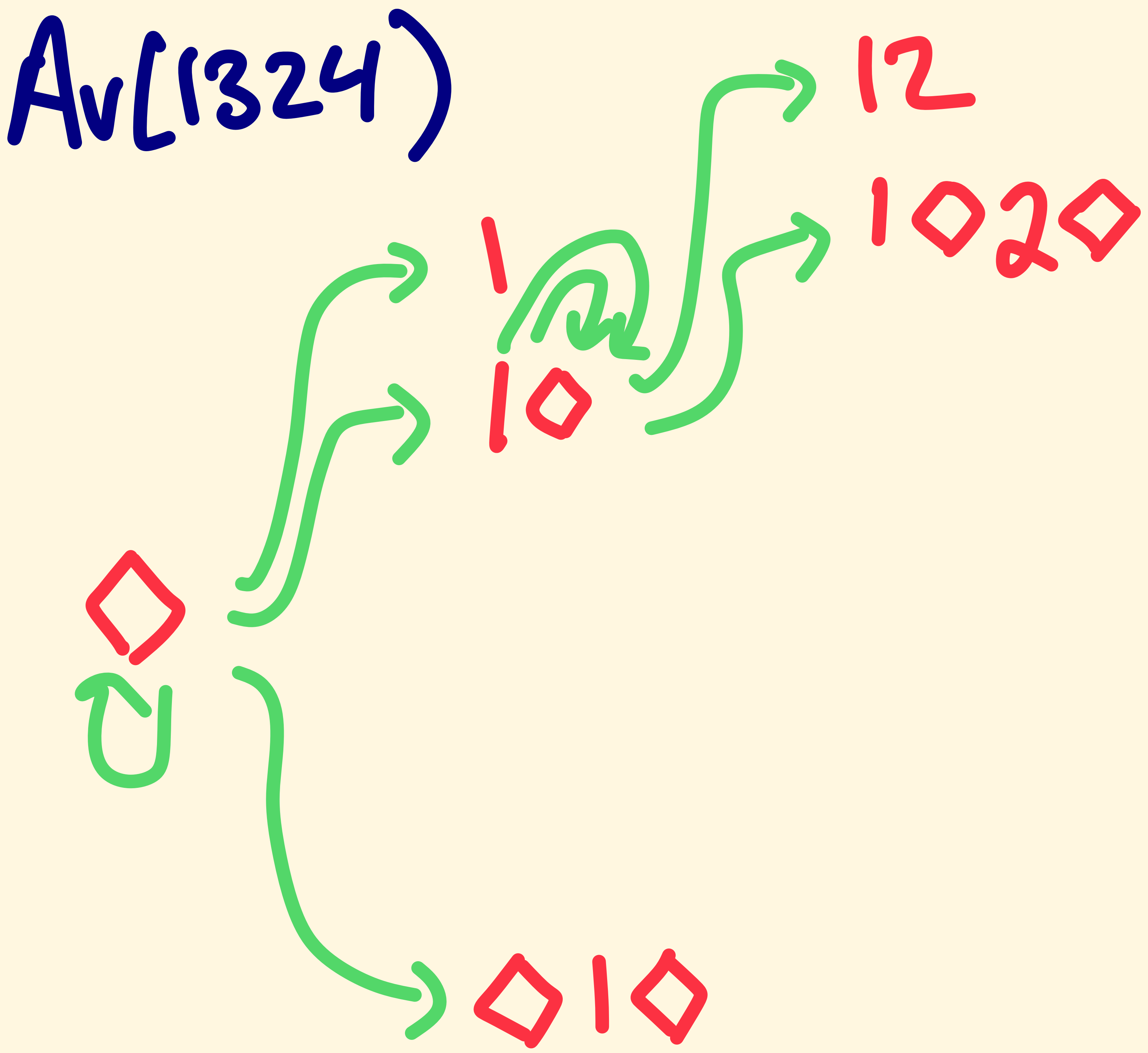
Insertion Encoding

Even if a class has an infinite insertion encoding, you can still use it to count the number of permutations in a class up to some point.



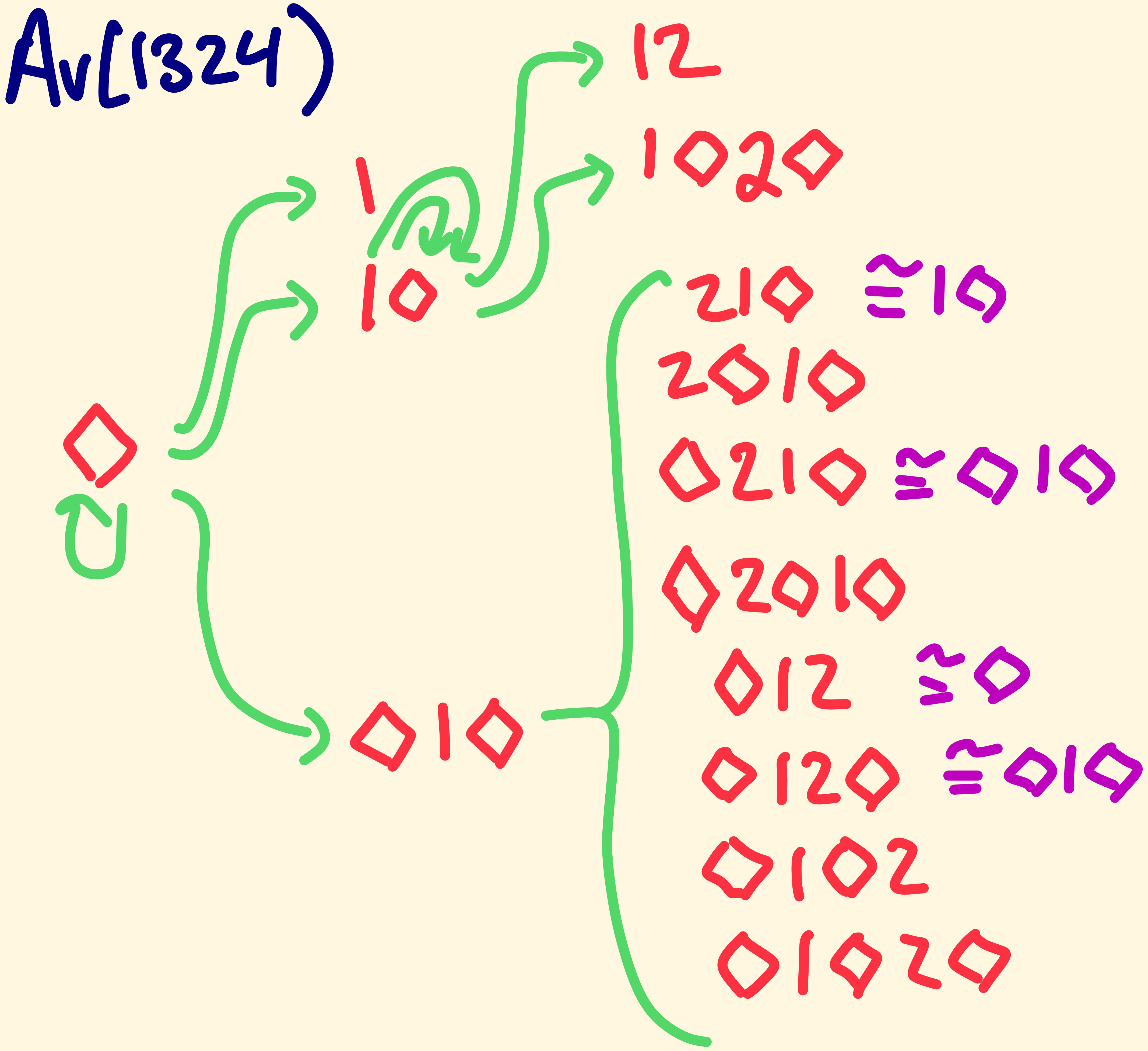
Insertion Encoding

Even if a class has an infinite insertion encoding, you can still use it to count the number of permutations in a class up to some point.



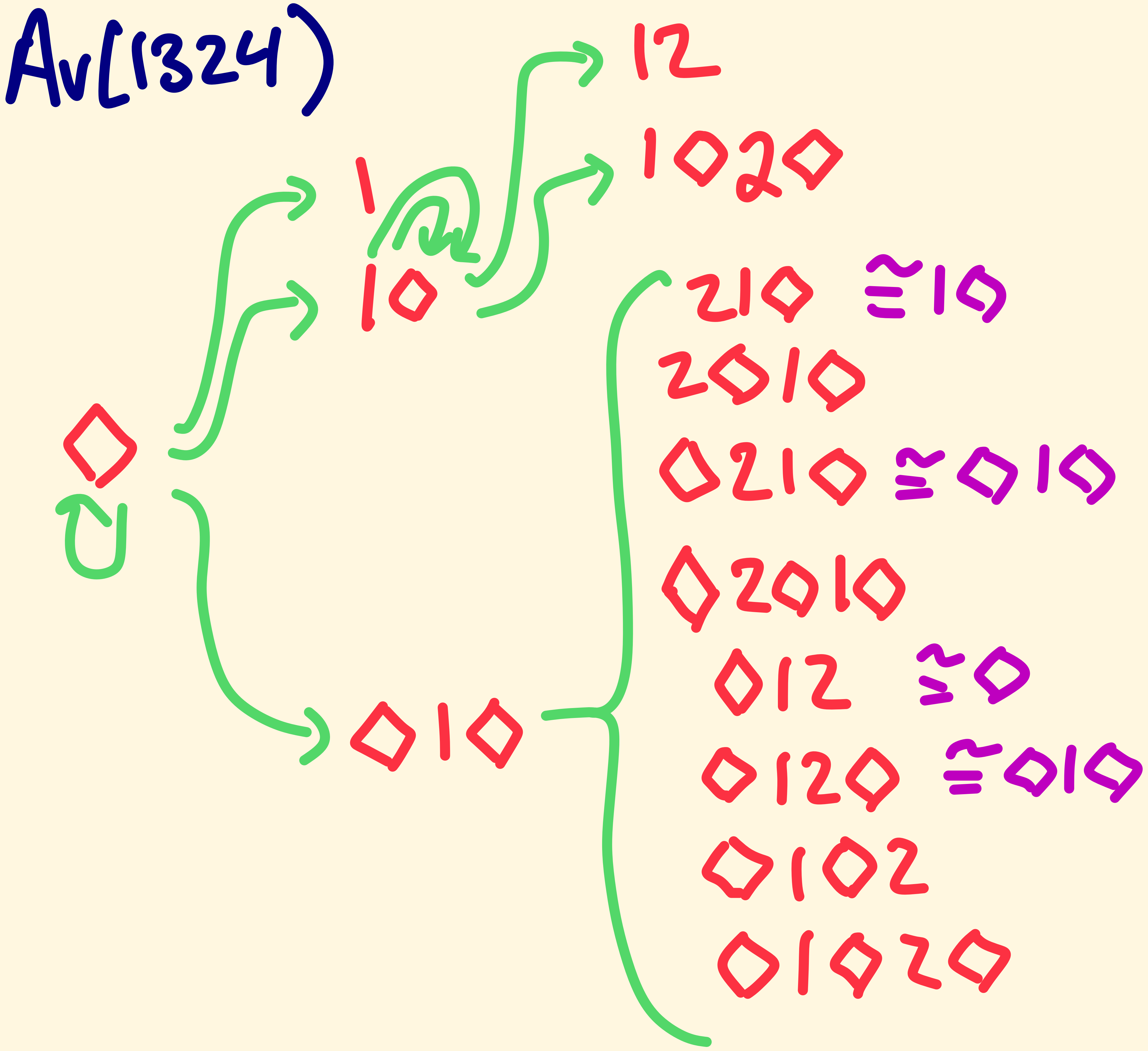
Insertion Encoding

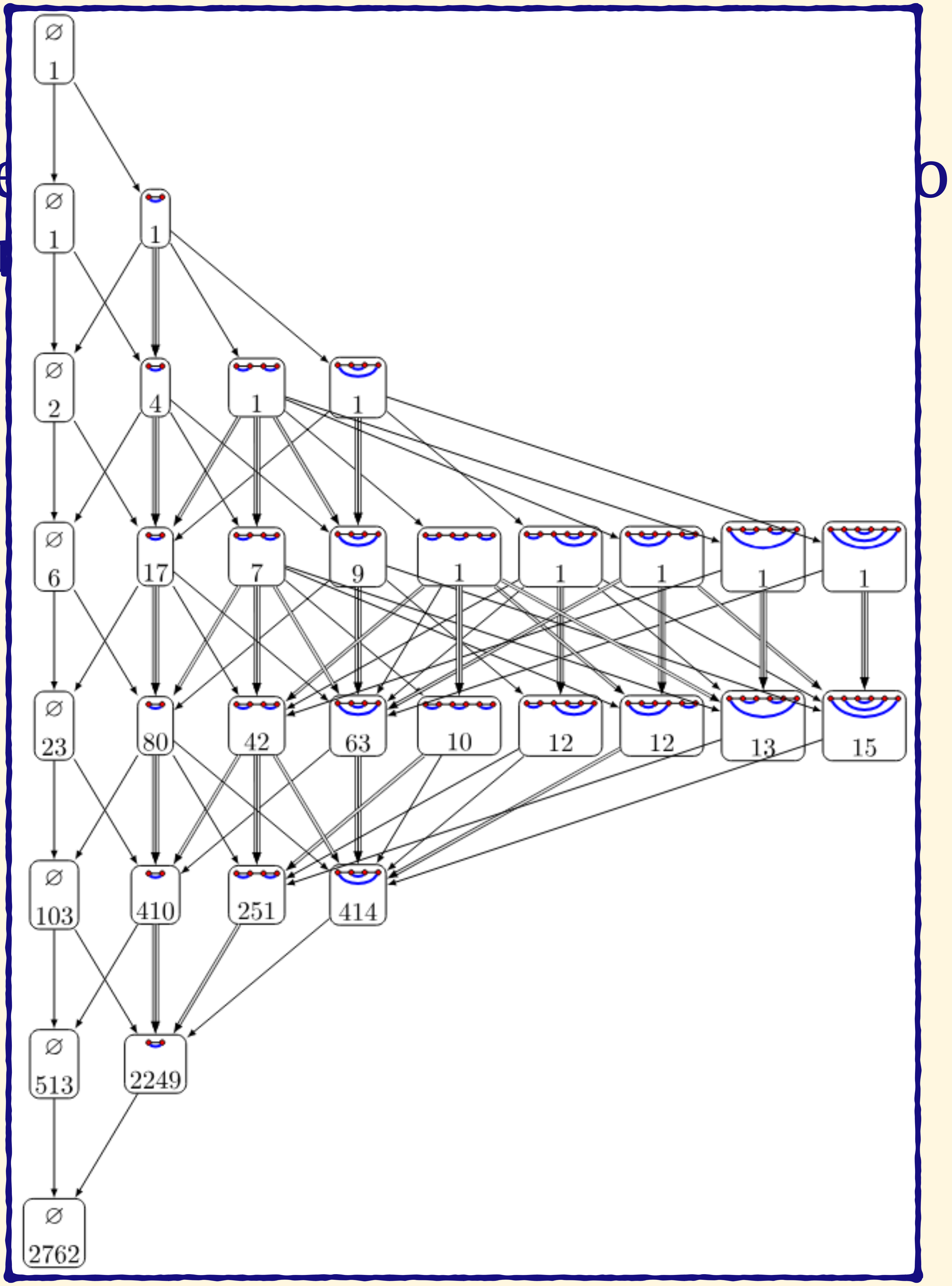
Even if a class has an infinite insertion encoding, you can still use it to count the number of permutations in a class up to some point.



Insertion Encoding

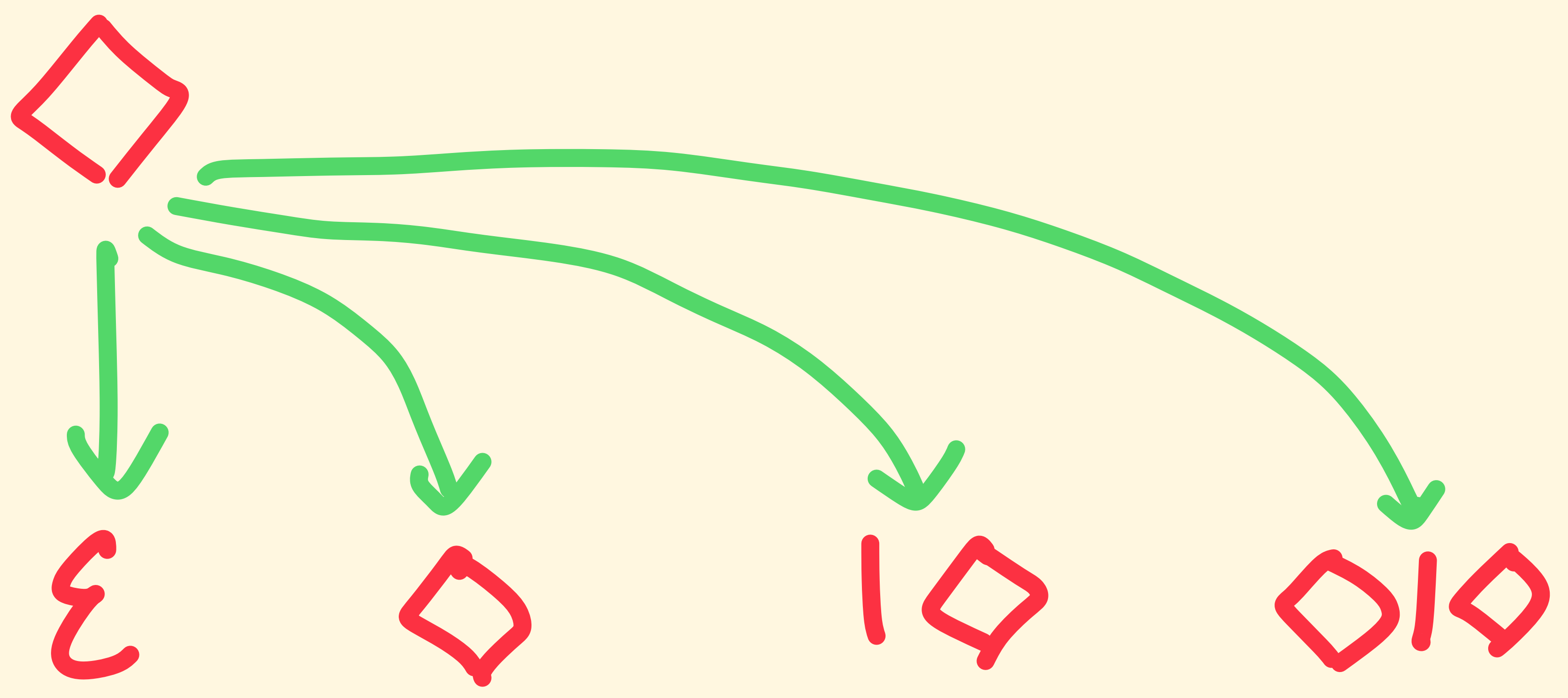
Even if a class has an infinite insertion e… o count
the number of permutations in a class u…





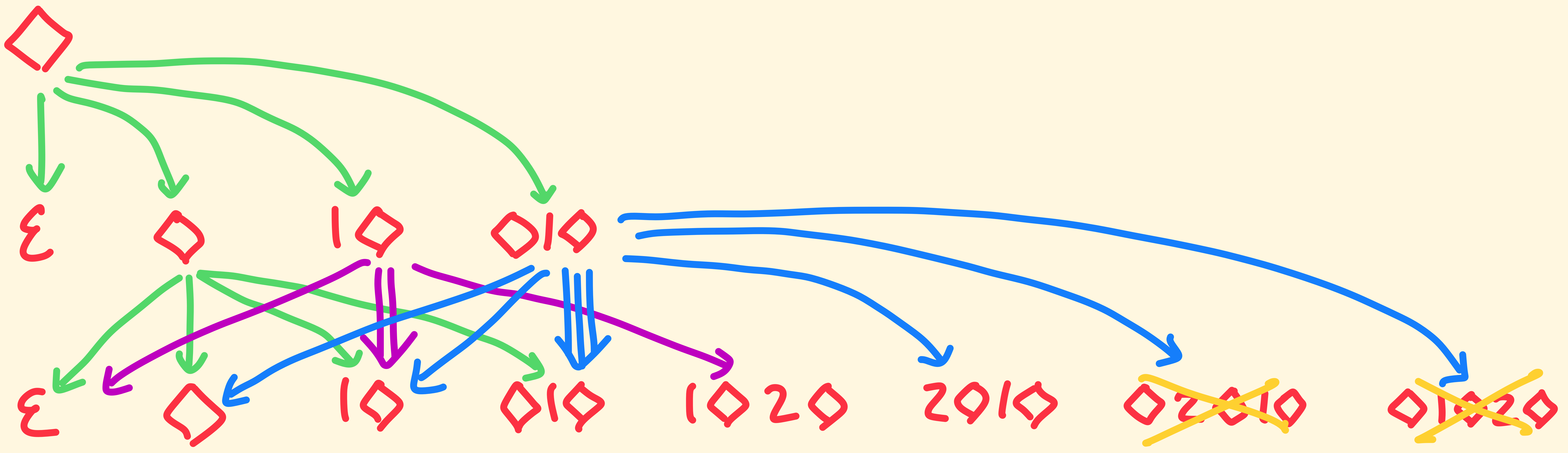
Insertion Encoding

Av(1324)



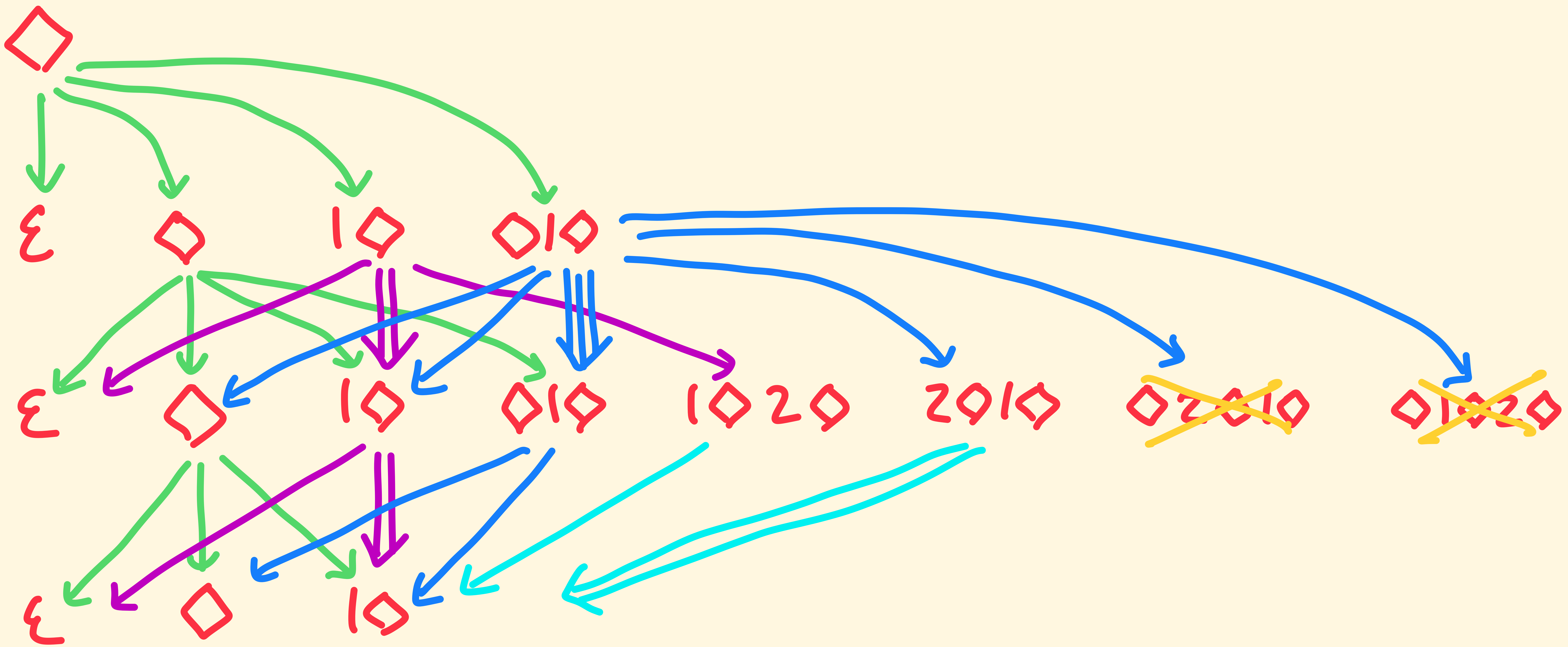
Insertion Encoding

Av(1324)



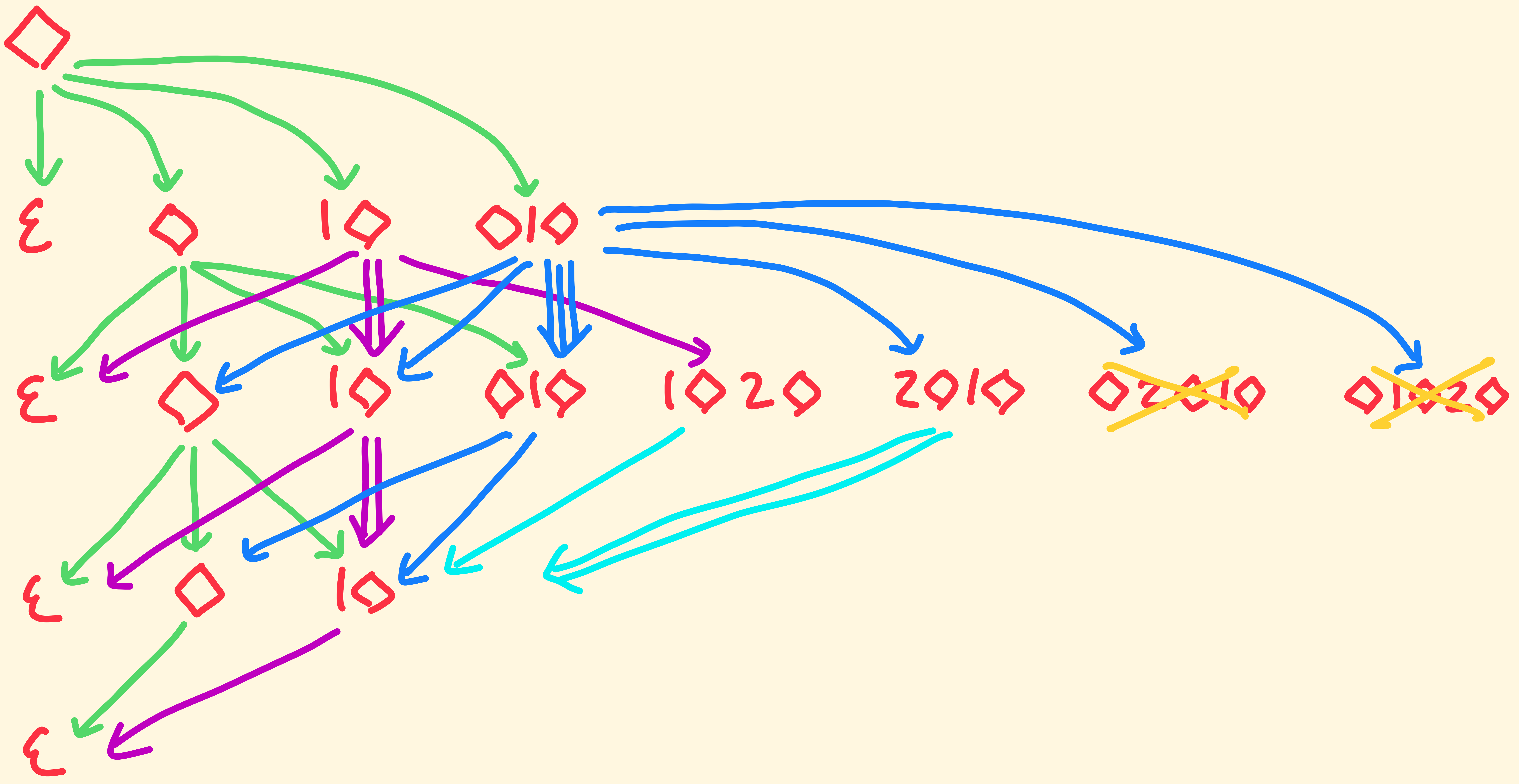
Insertion Encoding

Av(1324)



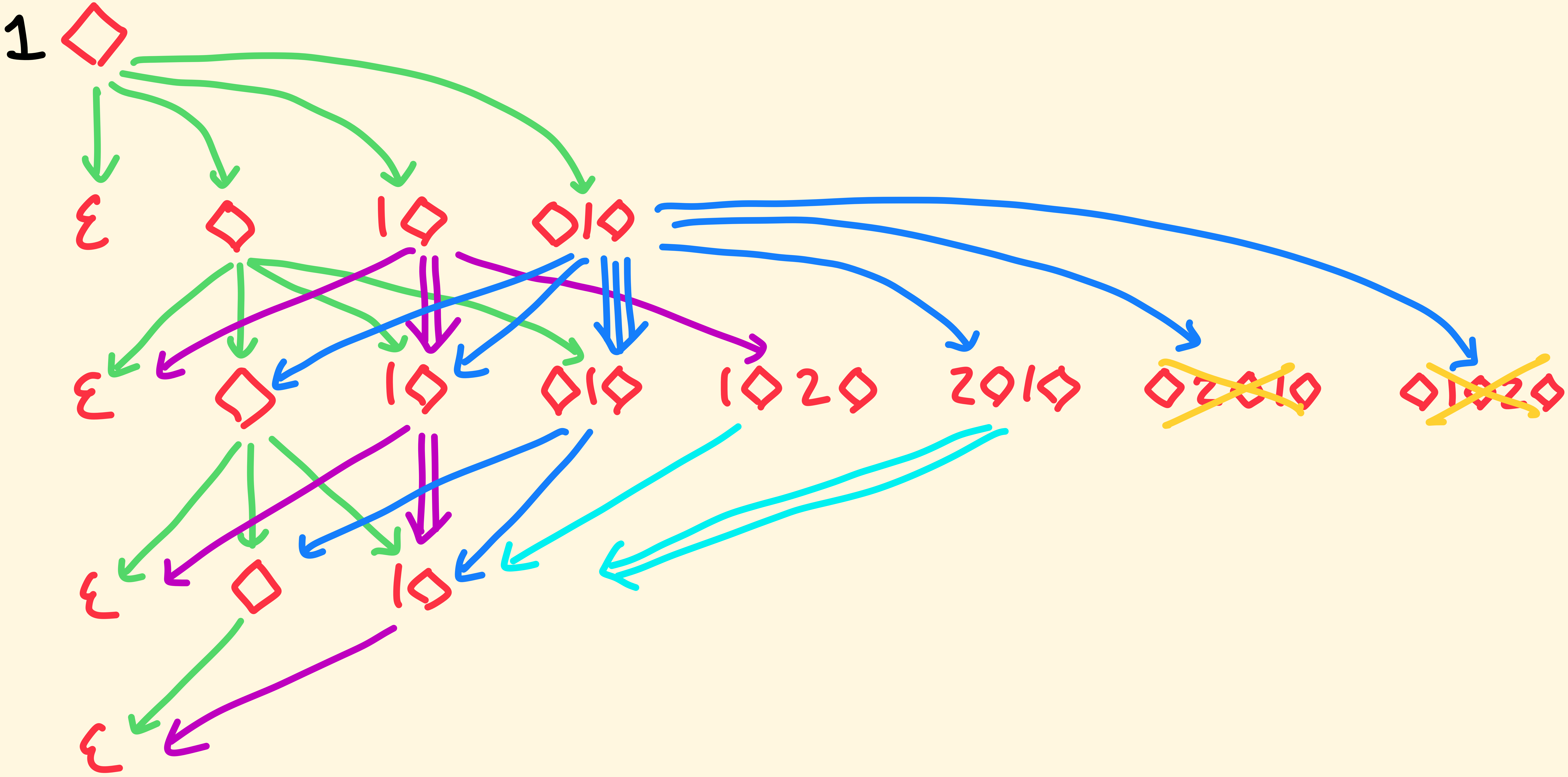
Insertion Encoding

Av(1324)



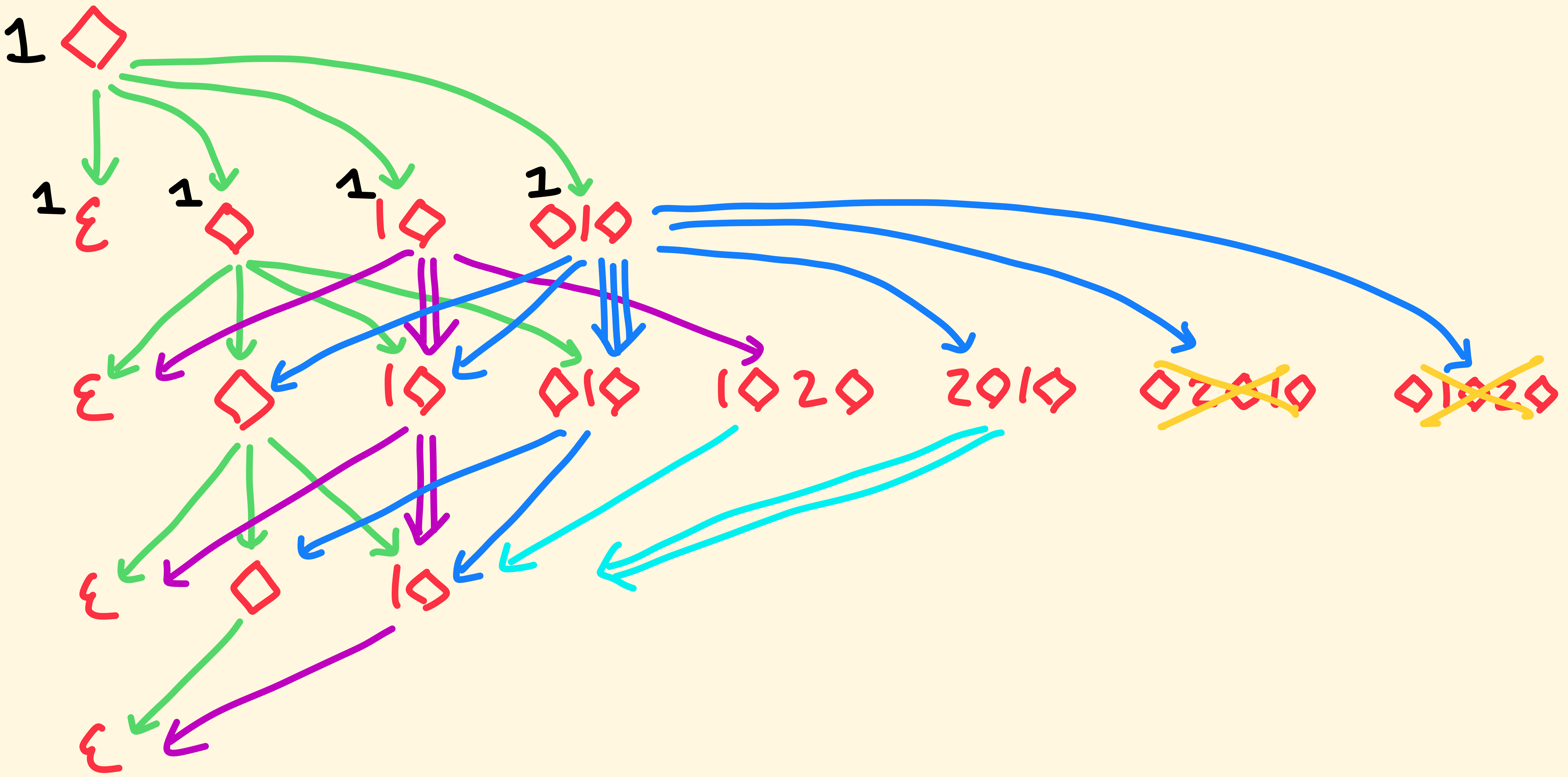
Insertion Encoding

Av(1324)



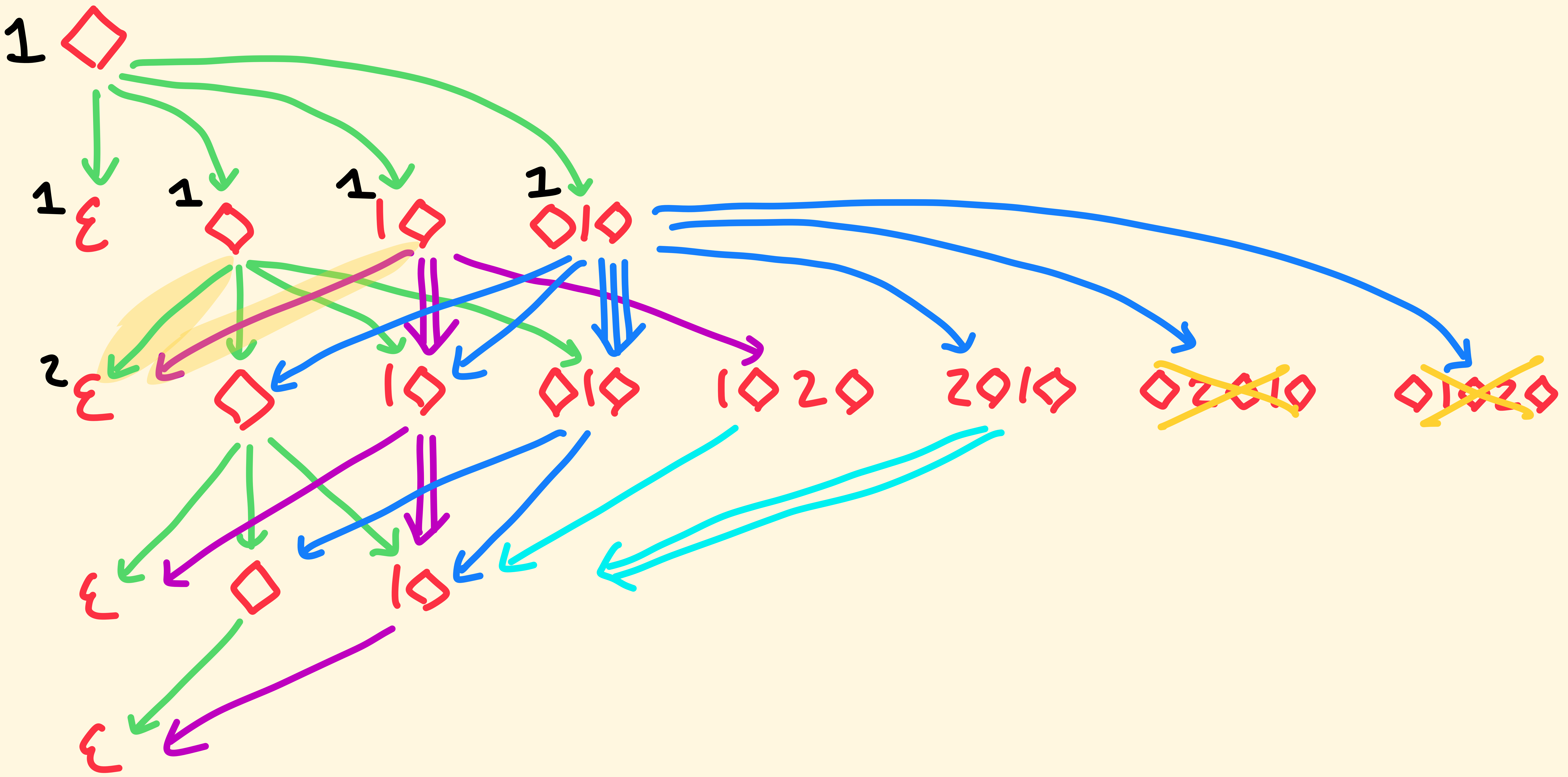
Insertion Encoding

Av(1324)



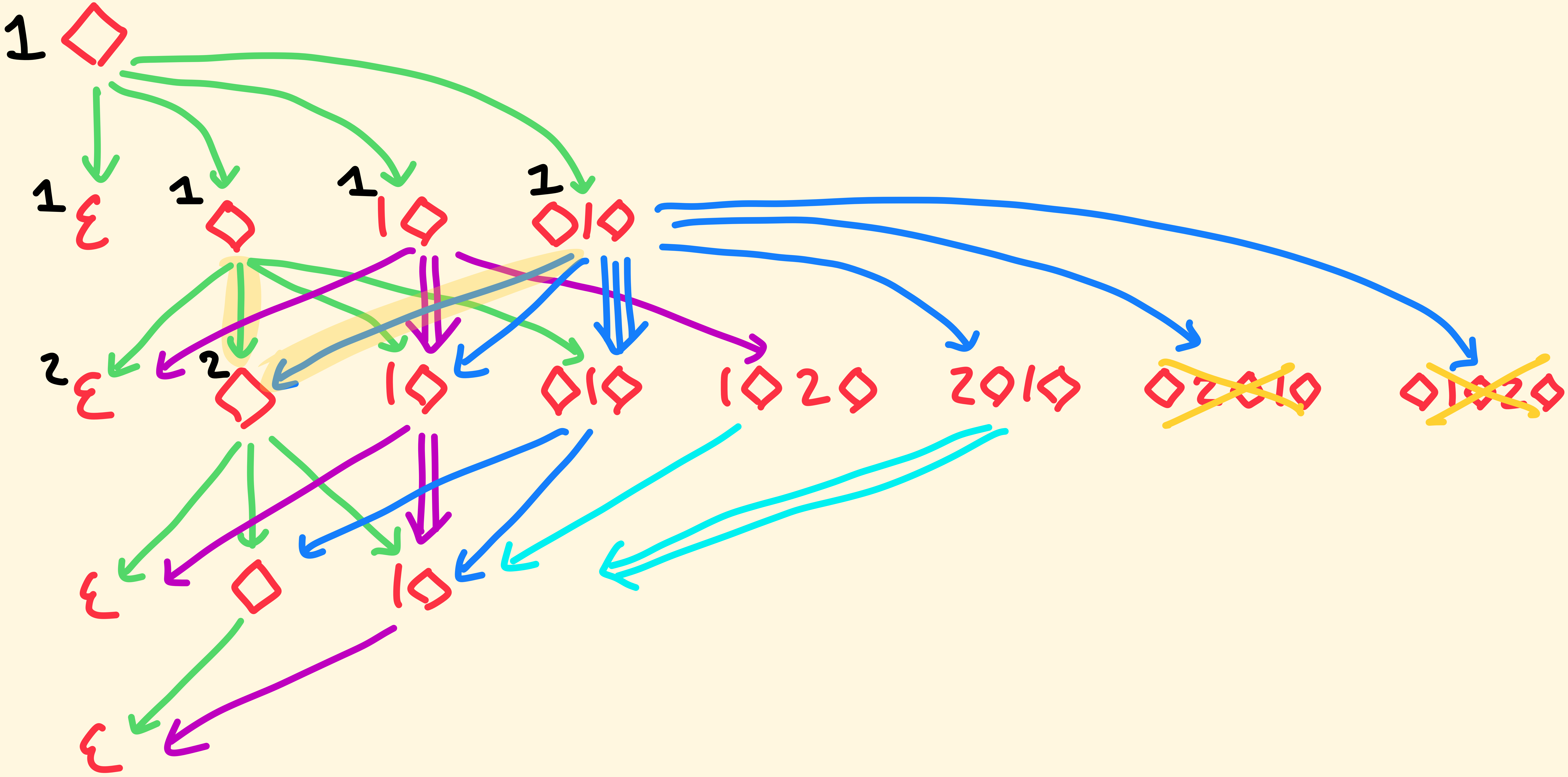
Insertion Encoding

Av(1324)



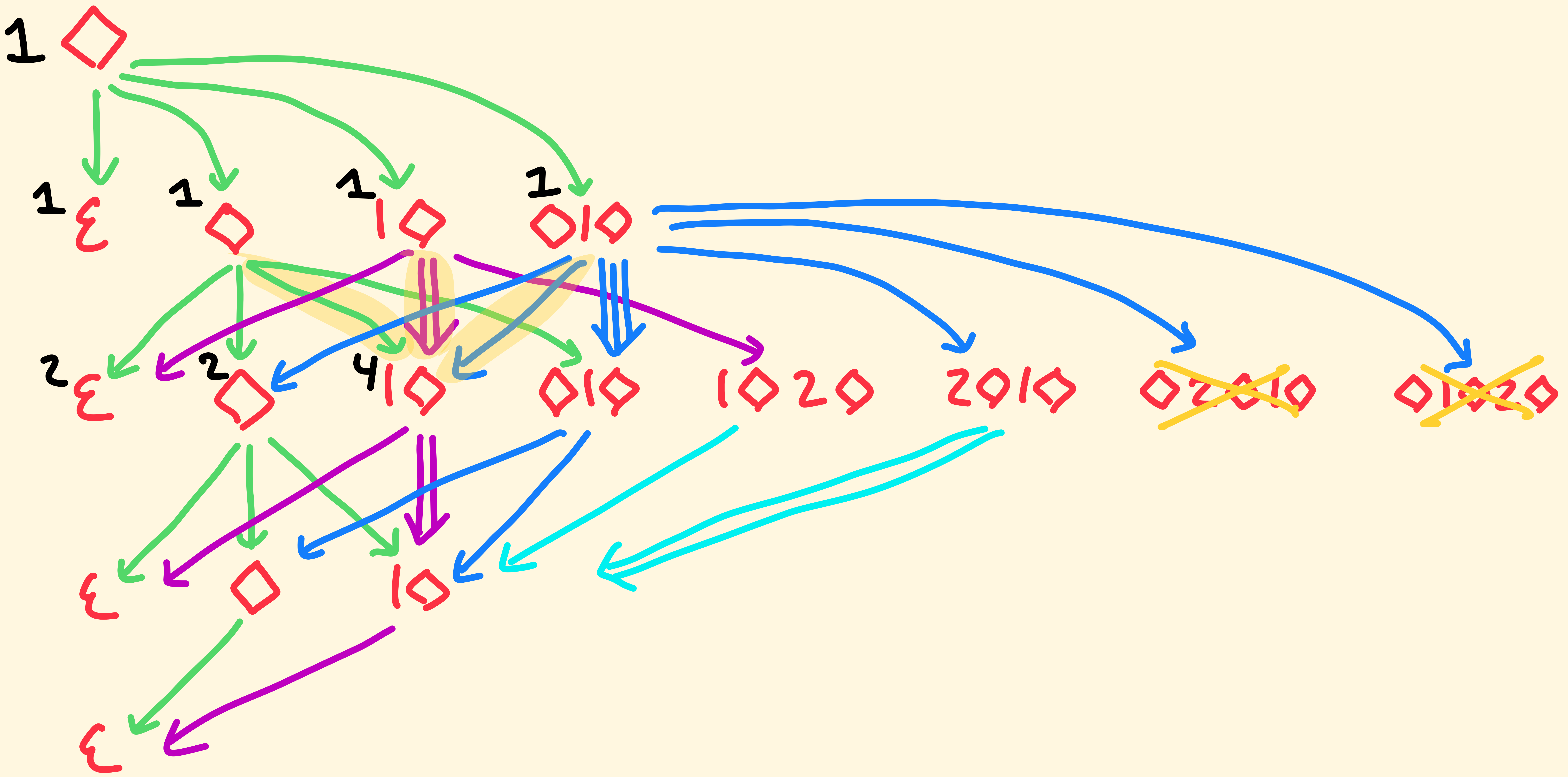
Insertion Encoding

Av(1324)



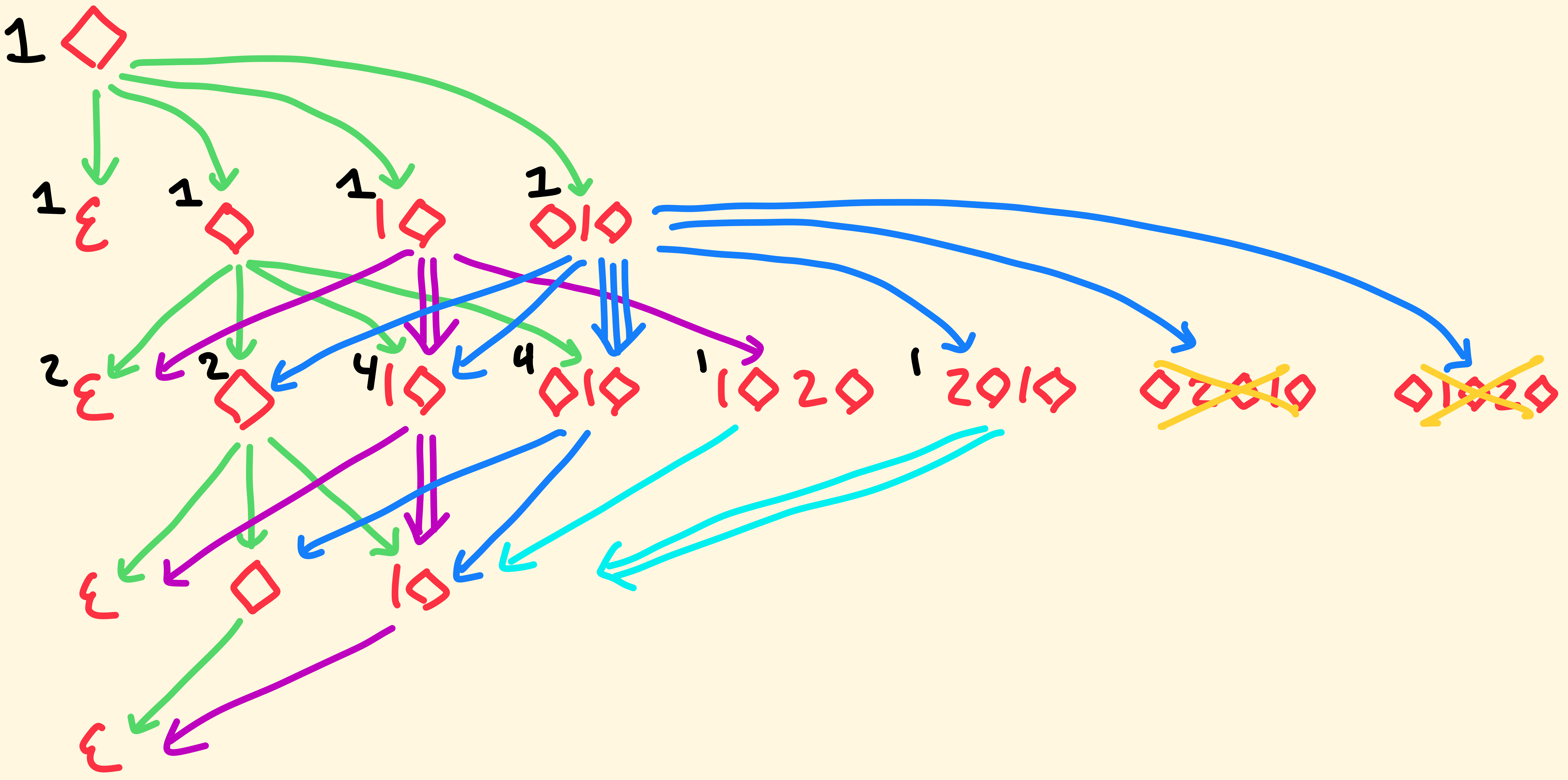
Insertion Encoding

Av(1324)



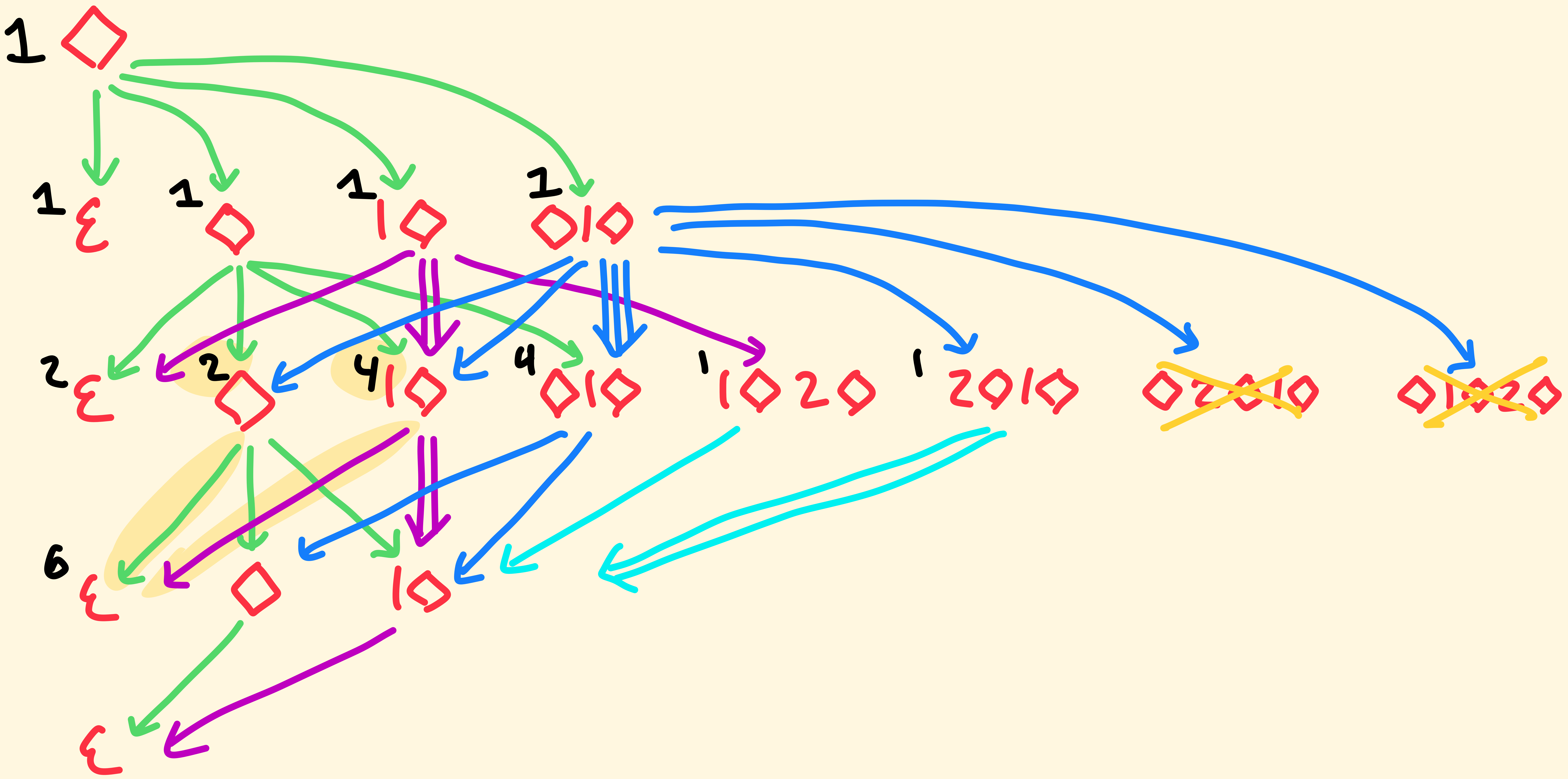
Insertion Encoding

Av(1324)



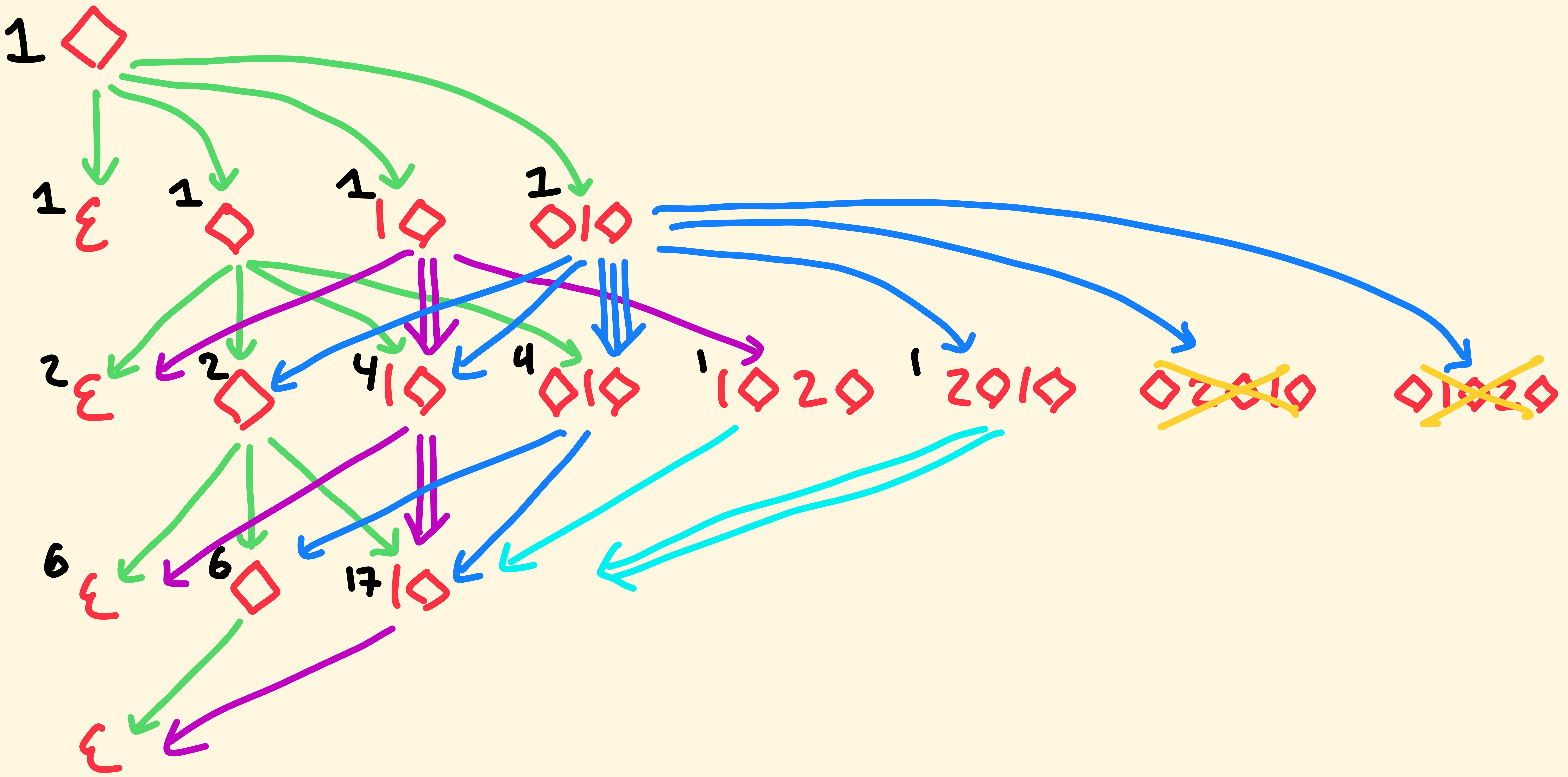
Insertion Encoding

Av(1324)



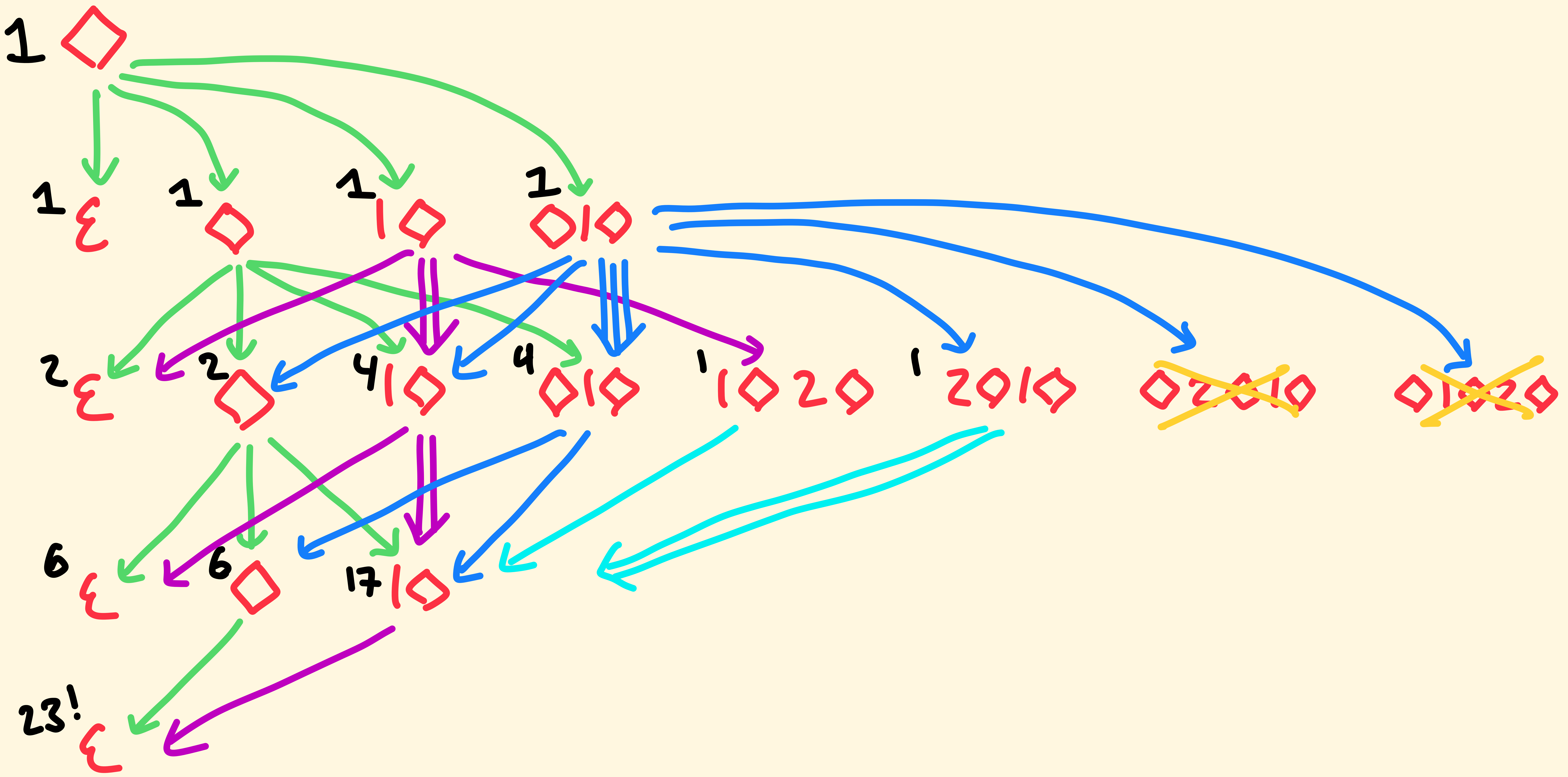
Insertion Encoding

Av(1324)



Insertion Encoding

Av(1324)



Insertion Encoding

The “link diagrams” in the 1324 paper are precisely encoding the relationships between slots — often you cannot fill slot A until after slot B has been closed.

Insertion Encoding

The “link diagrams” in the 1324 paper are precisely encoding the relationships between slots — often you cannot fill slot A until after slot B has been closed.

That means they can follow the link diagrams to know exactly what the transitions between simplified slot configurations are.

Insertion Encoding

The “link diagrams” in the 1324 paper are precisely encoding the relationships between slots — often you cannot fill slot A until after slot B has been closed.

That means they can follow the link diagrams to know exactly what the transitions between simplified slot configurations are.

Huge computational savings because the simplification is an expensive operation in the original insertion encoding.

Insertion Encoding

So the “big picture”, translated into the insertion encoding, is that the paper uses a very efficient construction to generate the insertion encoding finite state machine for all states with up to 25 slots.

Insertion Encoding

So the “big picture”, translated into the insertion encoding, is that the paper uses a very efficient construction to generate the insertion encoding finite state machine for all states with up to 25 slots.

It also uses some extremely clever theoretical and optimization tricks to reach length 50!

Table 2
The first 50 terms of the $Av(1324)$ series.

1
2
6
23
103
513
2762
15793
94776
591950
3824112
25431452
173453058
1209639642
8604450011
62300851632
458374397312
3421888118907
25887131596018
198244731603623
1535346218316422
12015325816028313
94944352095728825
757046484552152932
6087537591051072864
49339914891701589053
402890652358573525928
3313004165660965754922
27424185239545986820514
228437994561962363104048
1914189093351633702834757
16130725510342551986540152
136664757387536091240503406
1163812341034817216384582333
9959364766841851088593974979
85626551244475524038311935717
739479176041581588794042743521
6413612398452364144369673970347
55855094052029166019855630997080
488354507551082299792086219184434
4286013140398612535730177106798038
37753338738386034300928290519149333
333720028221302436110132711265898937
2959914488410727889919188039470296624
26338690757116988316771828238926079326
235113956679181729949424482617740434207
2105162587512716675745868833684827184388
18904804517351837590874336467009693522354
170253750251391700942449152528030601519757
1537516984674177479234766336099763469212469

Generalizing to Any Permutation Class

In the rest of this talk, I'll explain how to generalize this to any permutation class.

Generalizing to Any Permutation Class

In the rest of this talk, I'll explain how to generalize this to any permutation class.

Big idea: We use a structure that automatically discovers and tracks the relationships between slots.

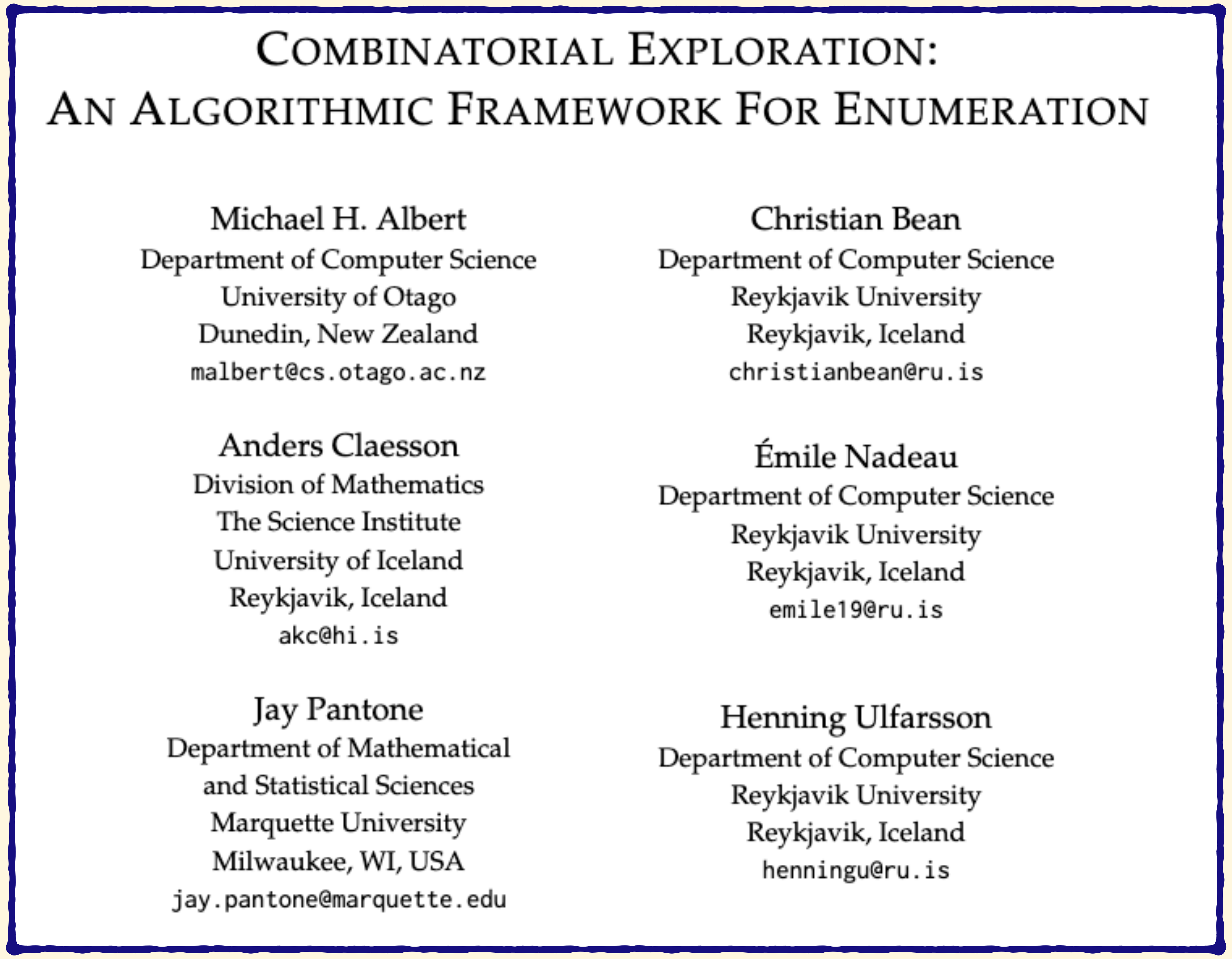
Generalizing to Any Permutation Class

In the rest of this talk, I'll explain how to generalize this to any permutation class.

Big idea: We use a structure that automatically discovers and tracks the relationships between slots.

It simultaneously derives the right "link pattern" analogues and uses them to count.

Tilings



COMBINATORIAL EXPLORATION:
AN ALGORITHMIC FRAMEWORK FOR ENUMERATION

Michael H. Albert
Department of Computer Science
University of Otago
Dunedin, New Zealand
malbert@cs.otago.ac.nz

Christian Bean
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
christianbean@ru.is

Anders Claesson
Division of Mathematics
The Science Institute
University of Iceland
Reykjavik, Iceland
akc@hi.is

Émile Nadeau
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
emile19@ru.is

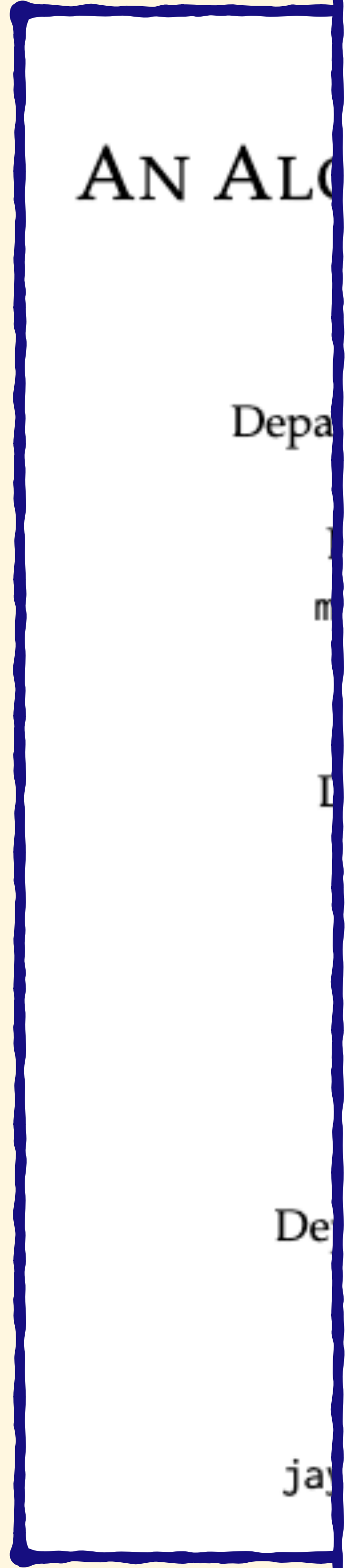
Jay Pantone
Department of Mathematical
and Statistical Sciences
Marquette University
Milwaukee, WI, USA
jay.pantone@marquette.edu

Henning Ulfarsson
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
henningu@ru.is

Automatic enumeration of permutation classes (and other objects)

Discovers (rigorously) combinatorial specifications, which can be turned into generating functions and polynomial-time counting algorithms.

Tili



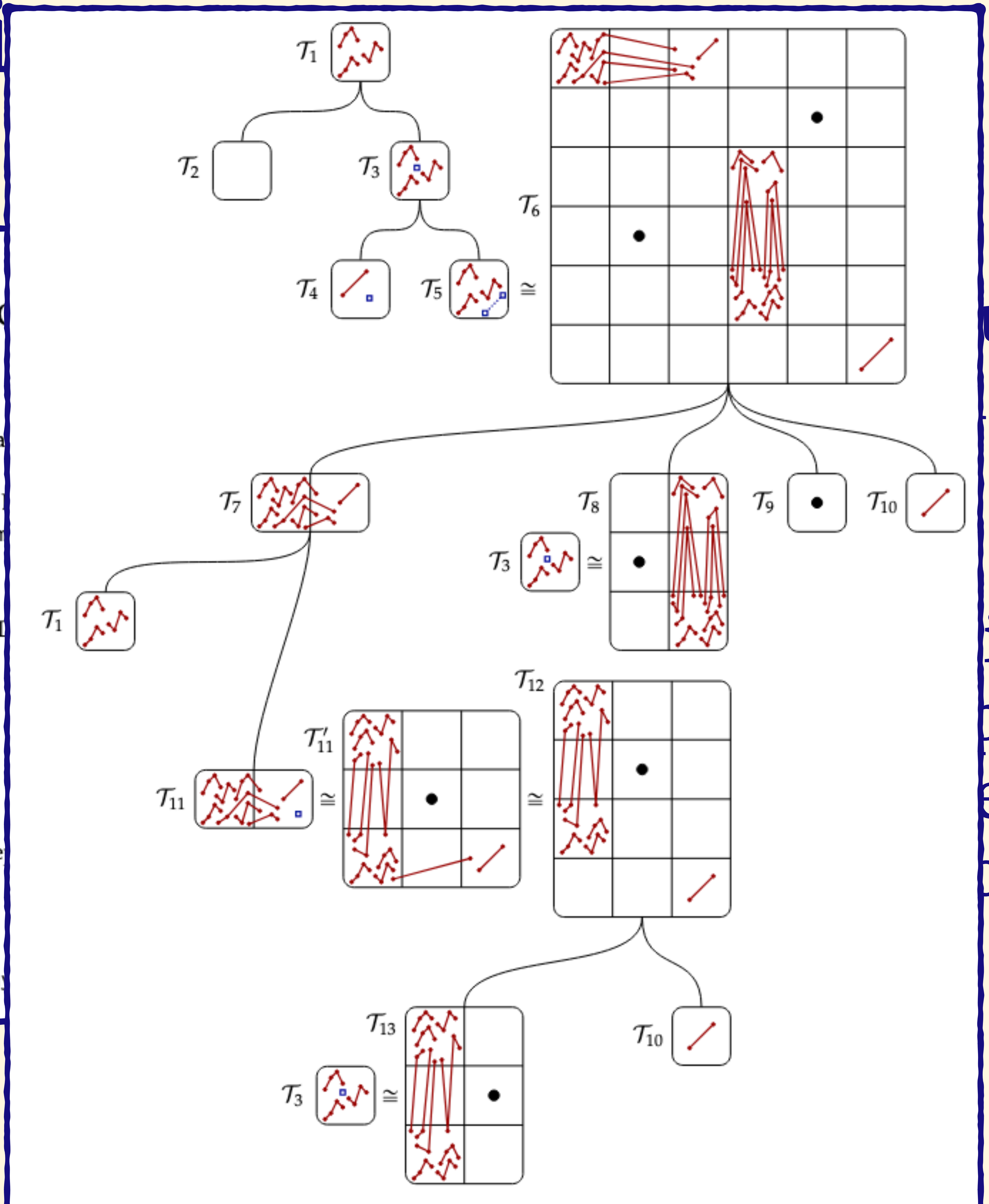


Figure 24: A pictorial representation of the combinatorial specification found by Combinatorial Exploration for Av(1243, 1342, 2143).

utomatic enumeration of permutation lasses (and other objects)

iscovers (rigorously) combinatorial pecifications, which can be turned into enerating functions and polynomial-time ounting algorithms.

Tili

AN AL

Depa

De

jay

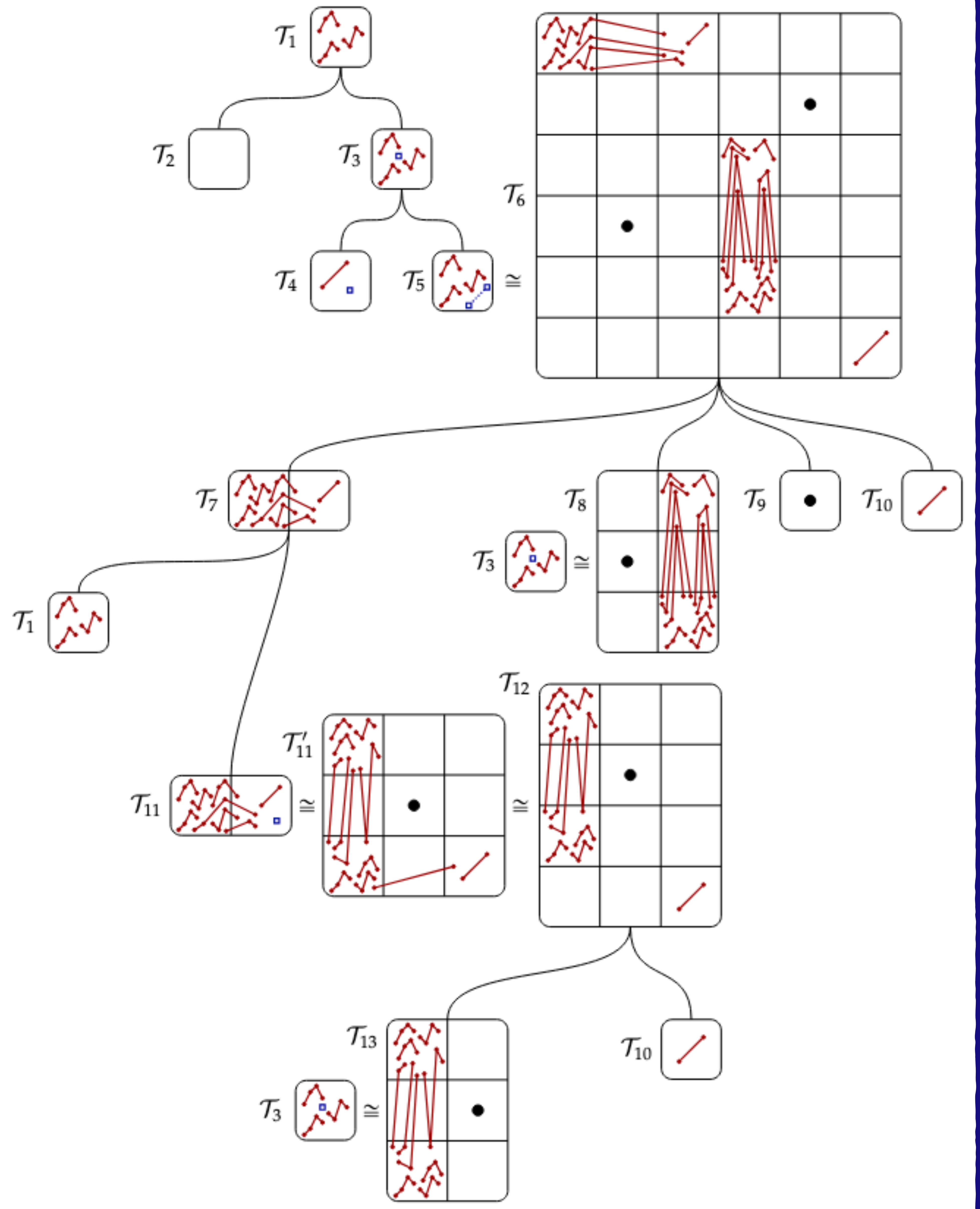


Figure 24: A pictorial representation of the combinatorial specification found by Combinatorial Exploration for Av(1243, 1342, 2143).

$$T_1(x) = T_2(x) + E_3(x)$$
$$T_2(x) = 1$$
$$E_3(x) = T_4(x) + E_5(x)$$
$$T_4(x) = x/(1-x)$$
$$E_5(x) = T_7(x) \cdot E_3(x) \cdot T_9(x) \cdot T_{10}(x)$$
$$T_7(x) = T_1(x) + E_{11}(x)$$
$$T_9(x) = x$$
$$T_{10}(x) = 1/(1-x)$$
$$E_{11}(x) = E_3(x) \cdot T_{10}(x)$$

$$T_1(x) = \frac{1 + x - \sqrt{1 - 6x + 5x^2}}{2x(2-x)}.$$

- u … ion of permutation
  a … jects)
- iscovers (rigorously) combinatorial
  pecifications, which can be turned into
  enerating functions and polynomial-time
  ounting algorithms.

Tili

An Al

Depa

T_1

De

ja

Fig
Exp

PermPAL Home Examples Search Random

The Permutation Pattern Avoidance Library (PermPAL)

PermPAL is a database of algorithmically-derived theorems about permutation classes.

The Combinatorial Exploration framework produces rigorously verified combinatorial specifications for families of combinatorial objects. These specifications then lead to generating functions, counting sequence, polynomial-time counting algorithms, random sampling procedures, and more.

This database contains 24,454 permutation classes for which specifications have been automatically found. This includes many classes that have been previously enumerated by other means and many classes that have not been previously enumerated.

Some Notables Successes:

- 6 out of 7 of the principal classes of length 4
- all 56 symmetry classes avoiding two patterns of length 4
- all 317 symmetry classes avoiding three patterns of length 4
- the "domino set" used by Bevan, Brignall, Elvey Price, and Pantone to investigate Av(1324)
- the class Av(3412, 52341, 635241) of Alland and Richmond corresponding a type of Schubert variety
- the class Av(2341, 3421, 4231, 52143) equal to the (Av(12), Av(21))-staircase (see Albert, Pantone, and Vatter), which appears to be non-D-finite
- all of the permutation classes counted by the Schröder numbers conjectured by Eric Egge
- the class Av(34251, 35241, 45231), equal to the preimage of Av(321) under the West-stack-sorting operation (see Defant)

Section 2.4 of the article Combinatorial Exploration: An Algorithmic Framework for Enumeration gives a more comprehensive list of notable results.

The comb_spec_searcher github repository contains the open-source python

$(x) \cdot T_{10}(x)$

$6x + 5x^2$

$x)$

ion of permutation
jects)

rously) combinatorial
which can be turned into
ctions and polynomial-time
ithms.

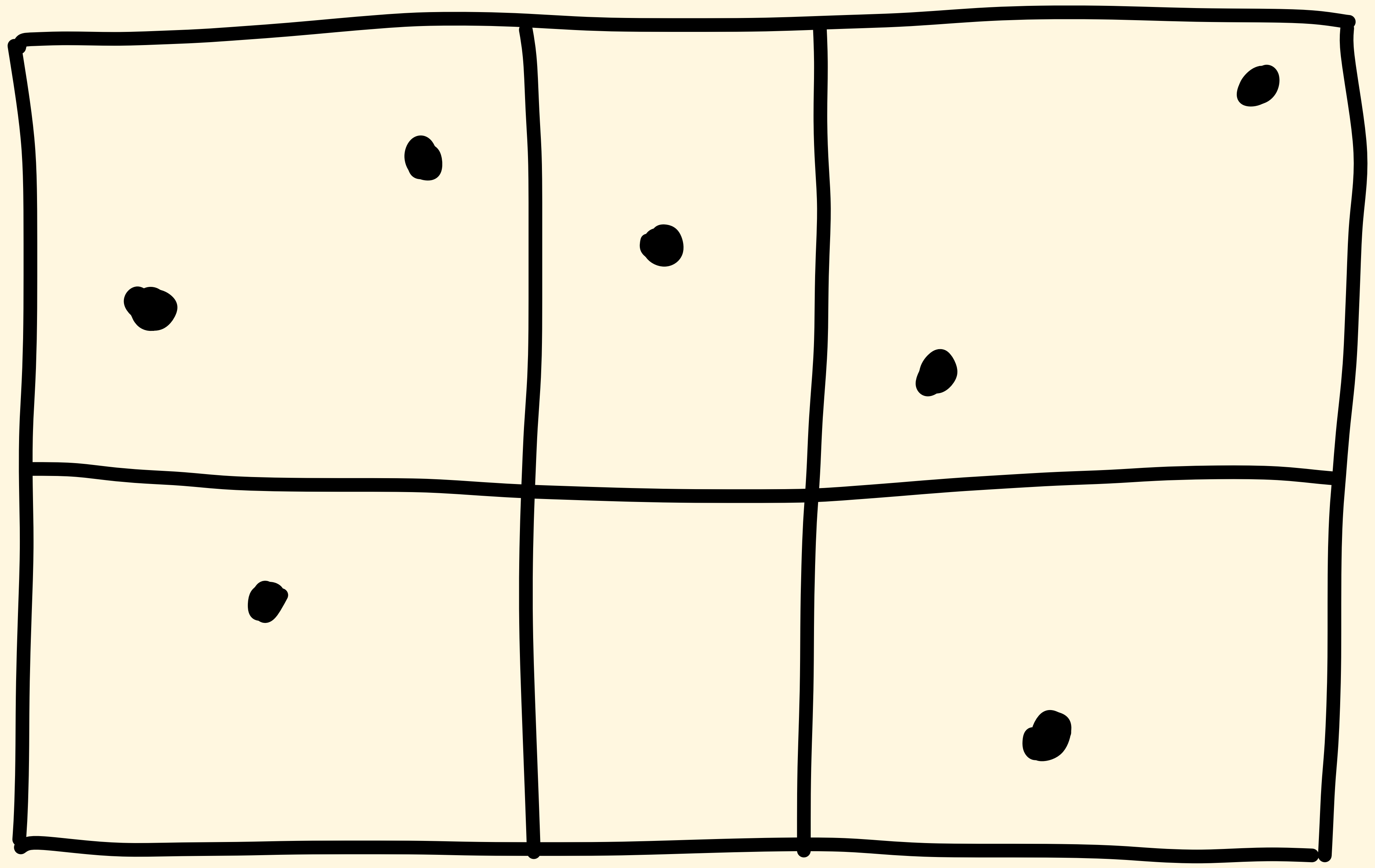
Tilings

One of the fundamental tools for Combinatorial Exploration is the *tiling.* It's essentially a data structure that represents a set of (gridded) permutations.

Tilings

One of the fundamental tools for Combinatorial Exploration is the *tiling*. It's essentially a data structure that represents a set of (gridded) permutations.

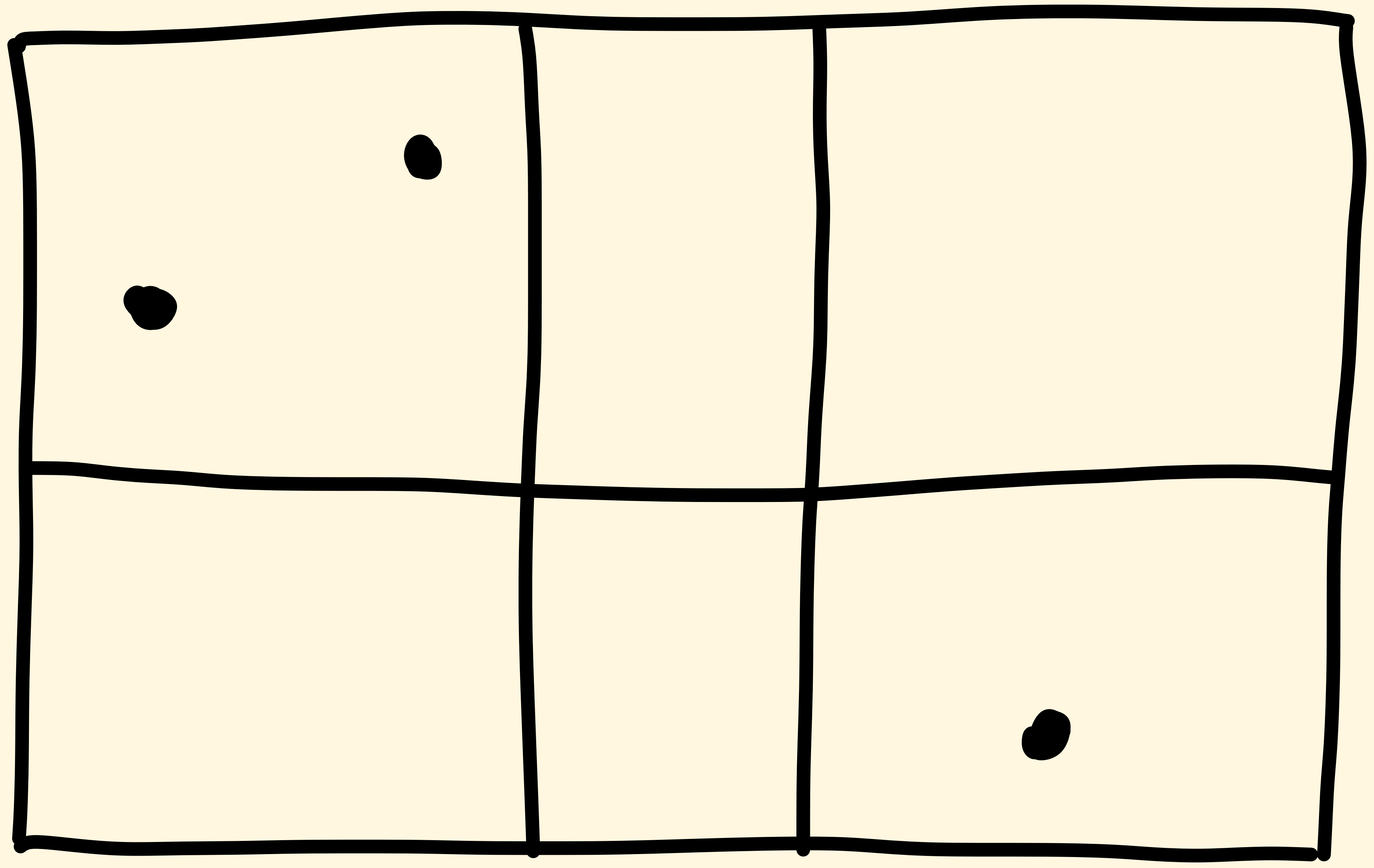
Gridded permutation = a permutation with grid lines draw so that entries are split into cells of a grid



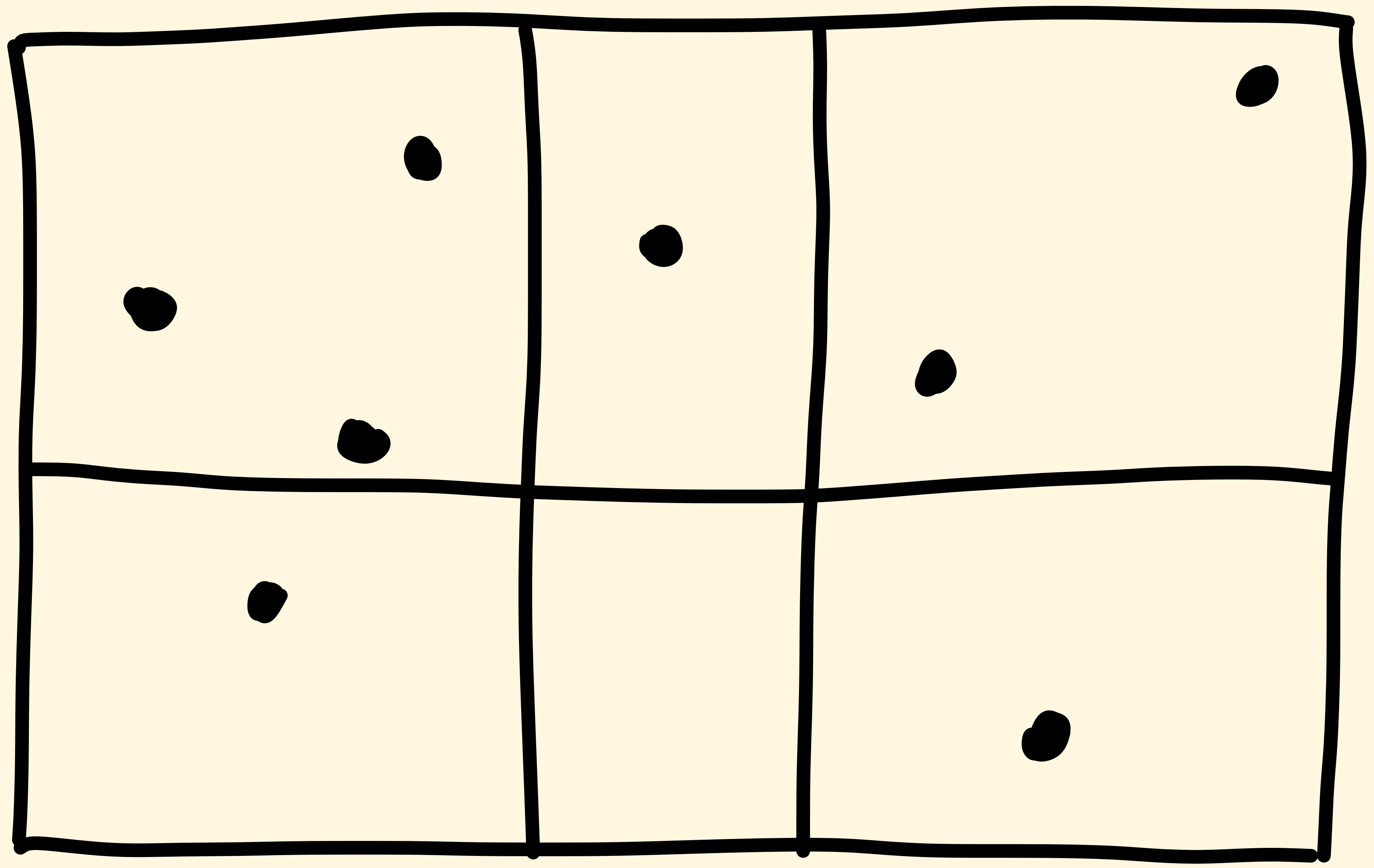
underlying permutation:
4265317

Tilings

A gridded permutation *p* contains a gridded permutation *q* as a pattern if there is a subsequence of entries of *p* that are order-isomorphic to *q* and <u>in the same cells.</u>

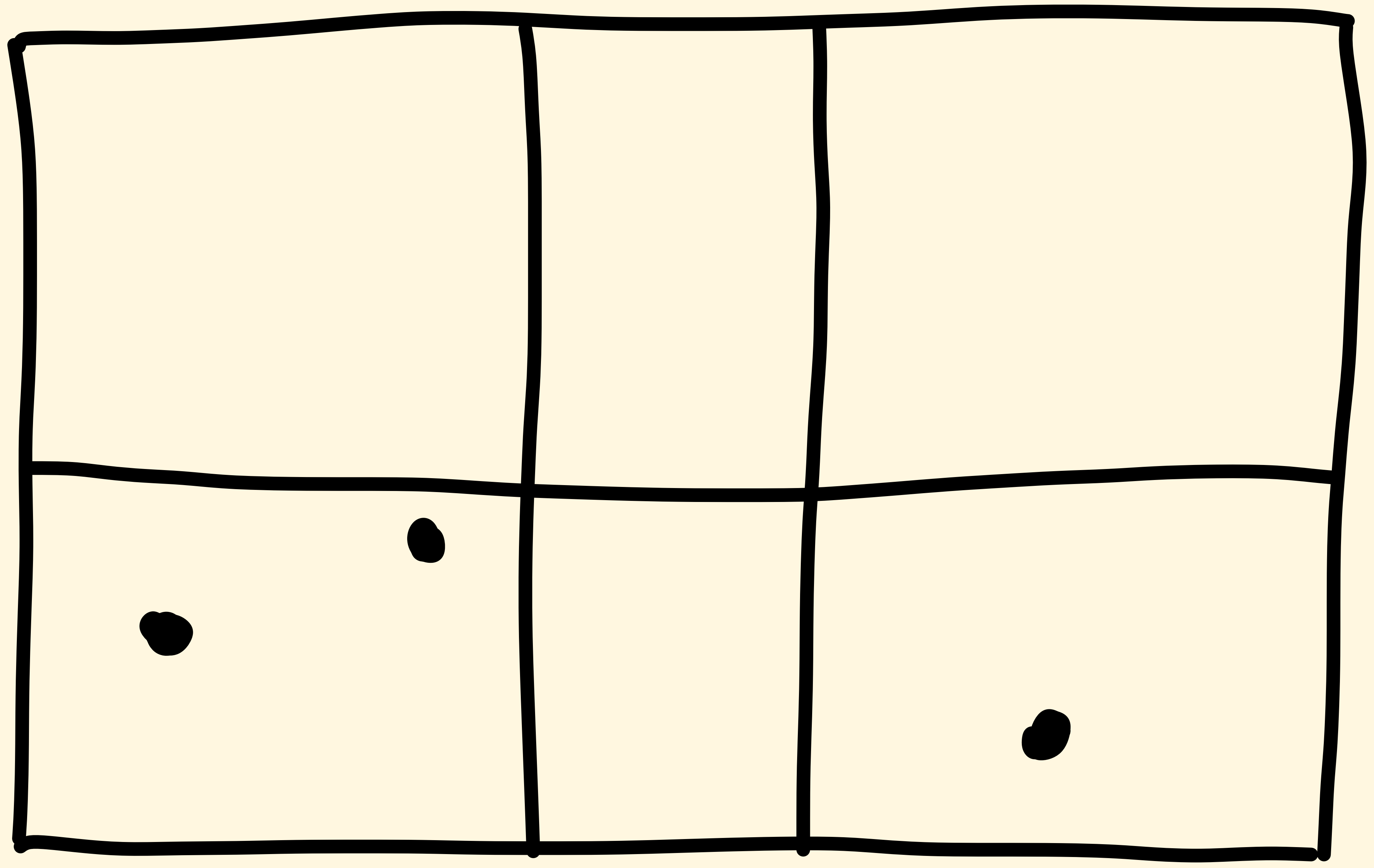


$\leq$

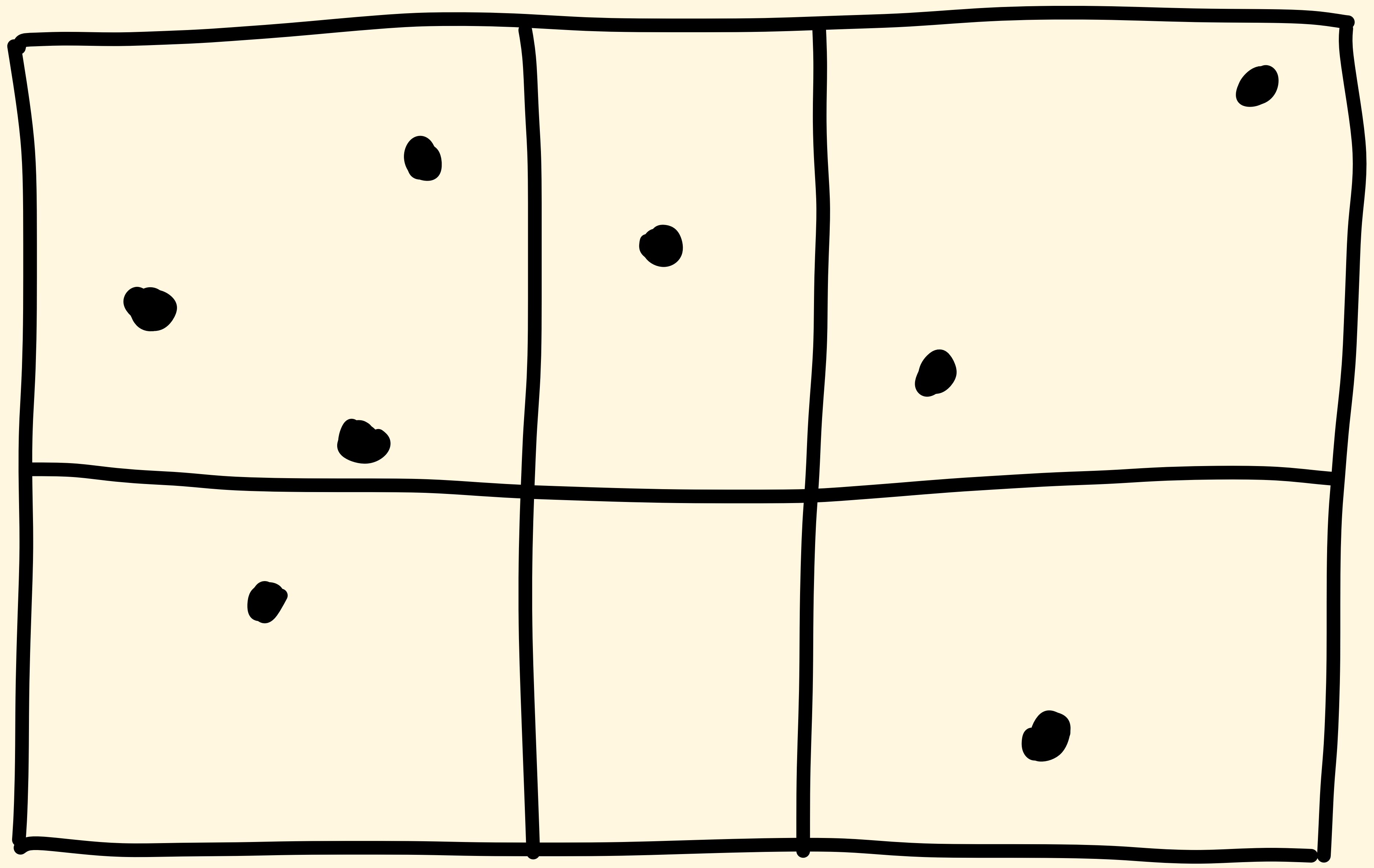


Tilings

A gridded permutation p contains a gridded permutation q as a pattern if there is a subsequence of entries of p that are order-isomorphic to q and in the same cells.



$\not\leq$

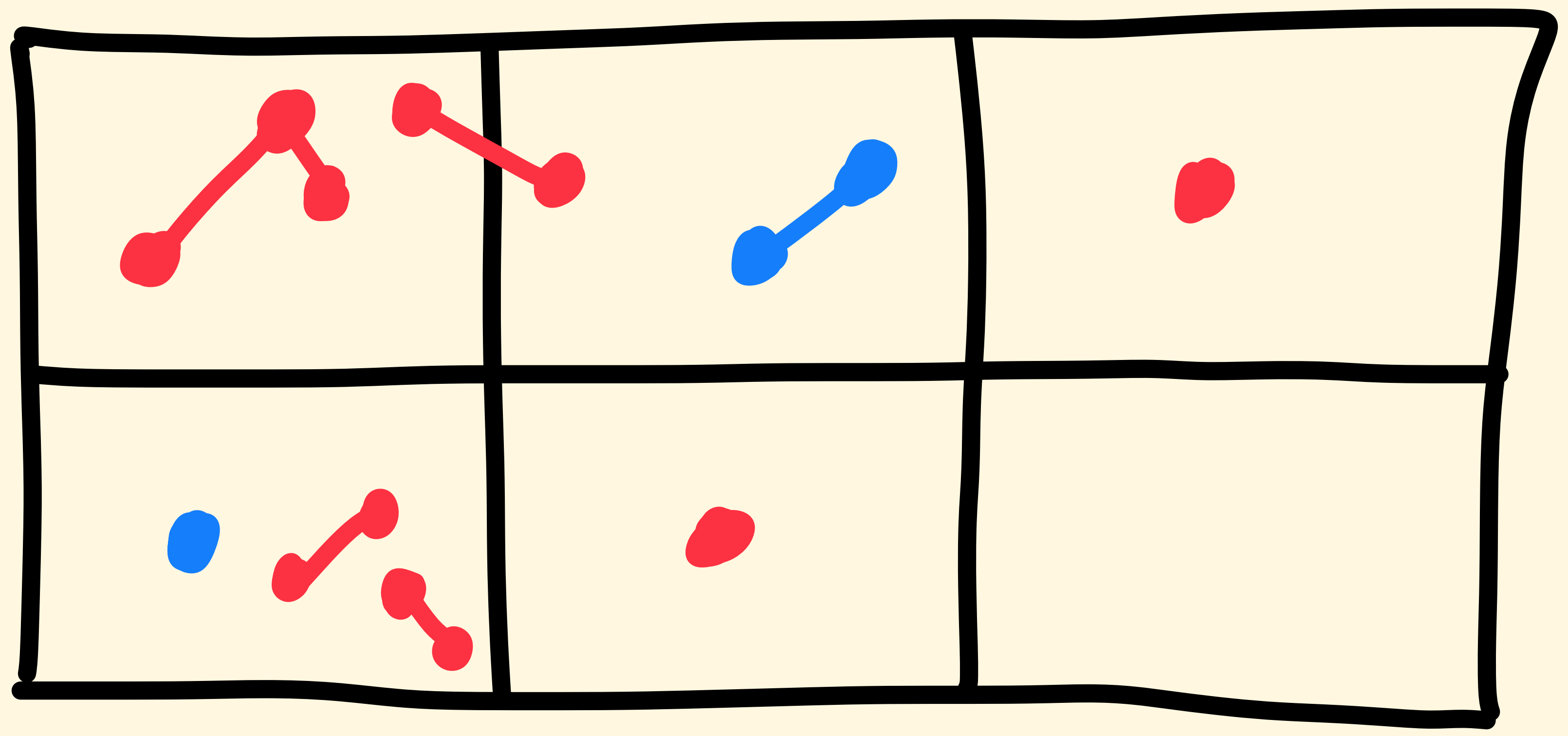


Tilings

A *tiling* is a grid with
obstructions: gridded permutations that must be avoided
requirements: gridded permutations that must be contained

A tiling represents the set of all gridded permutations that can be drawn on that grid that avoid all of the obstructions and contain all of the requirements.

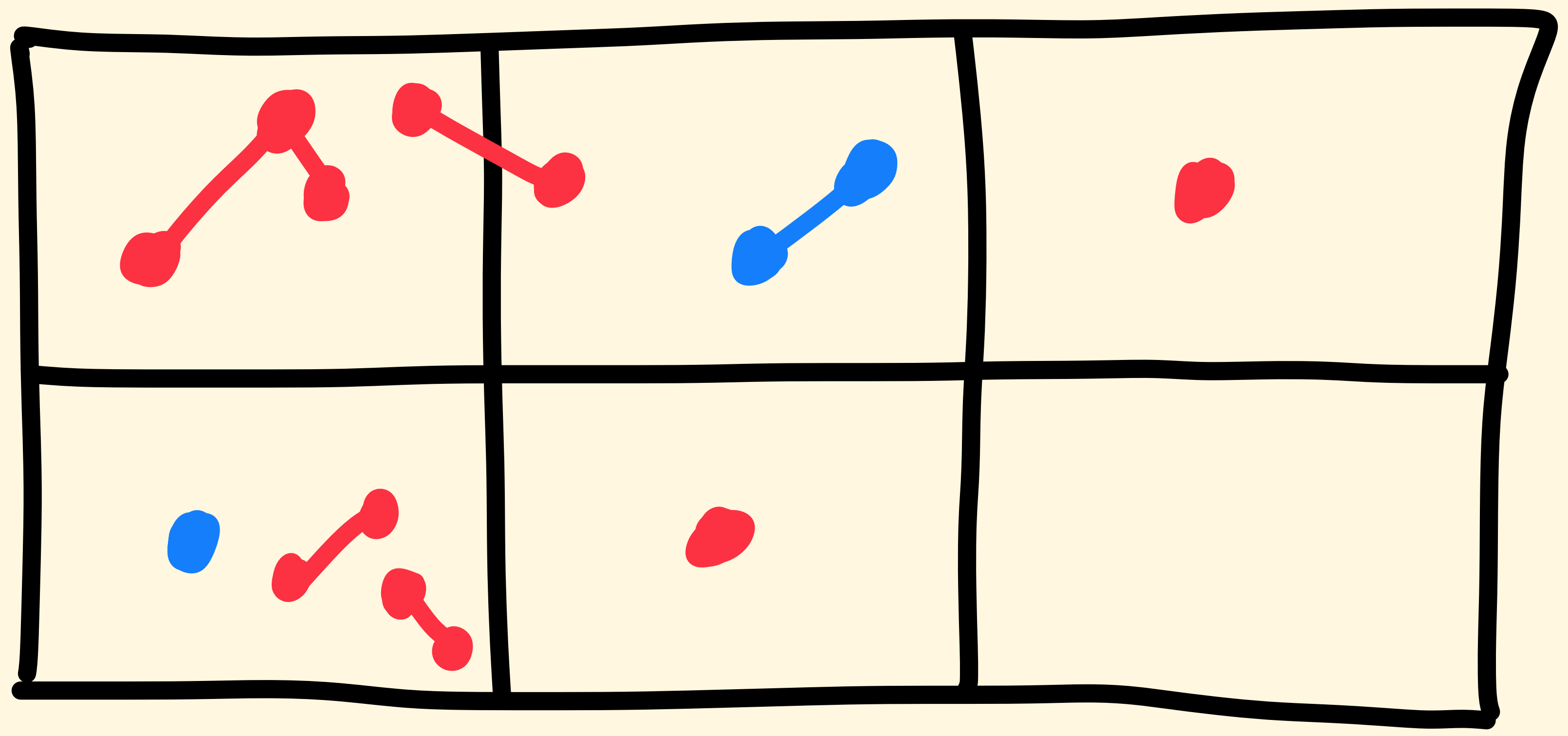
Tilings

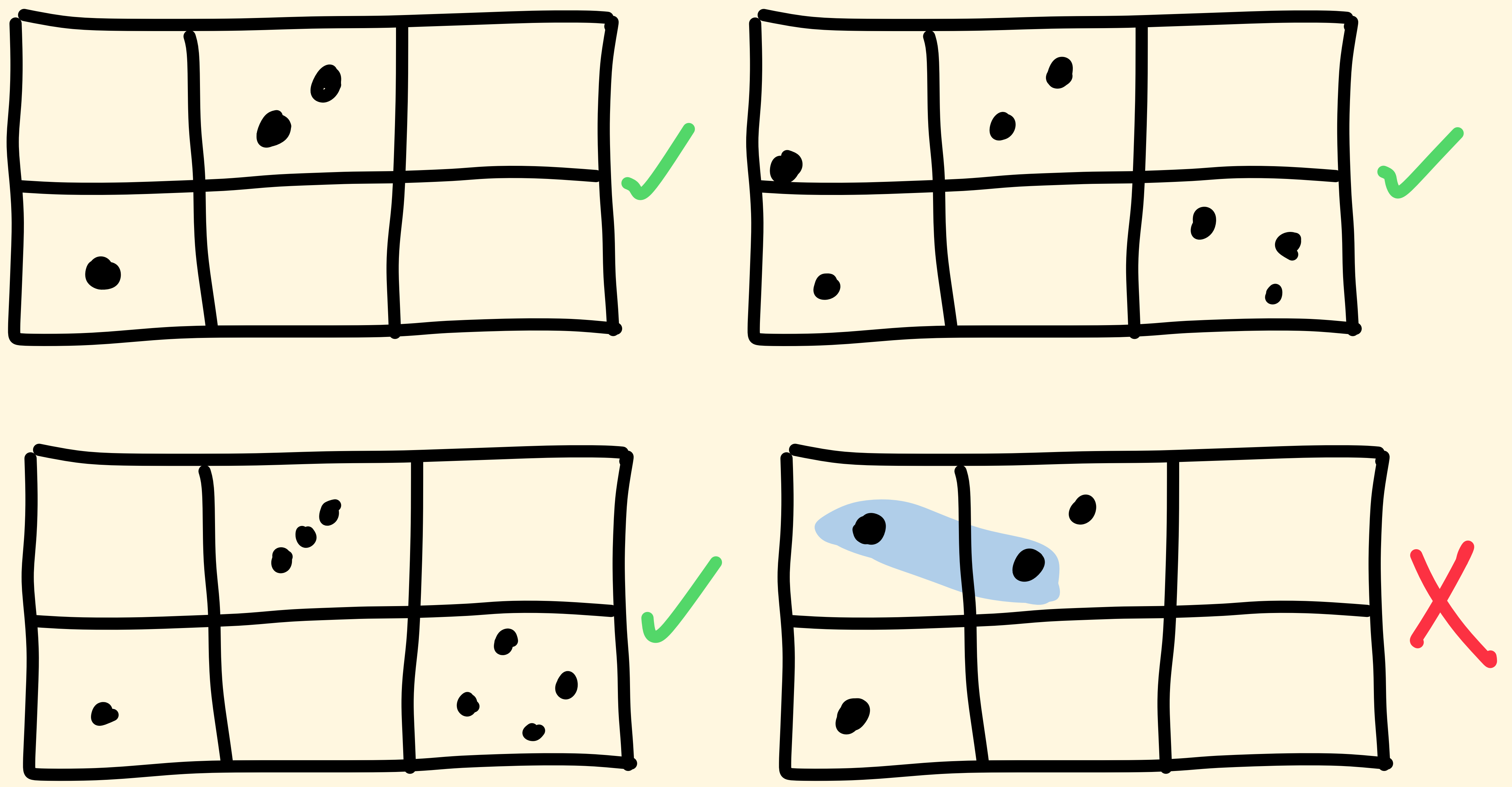


The tiling represents all gridded permutations on a 2x3 grid with:

- exactly one point in the bottom-left cell
- no points in the bottom-middle or top-right cells
- no 132 pattern in the top left cell
- no crossing 21 pattern between the top-left and top-middle cells
- contains a 12 pattern in the top-middle cell

Tilings

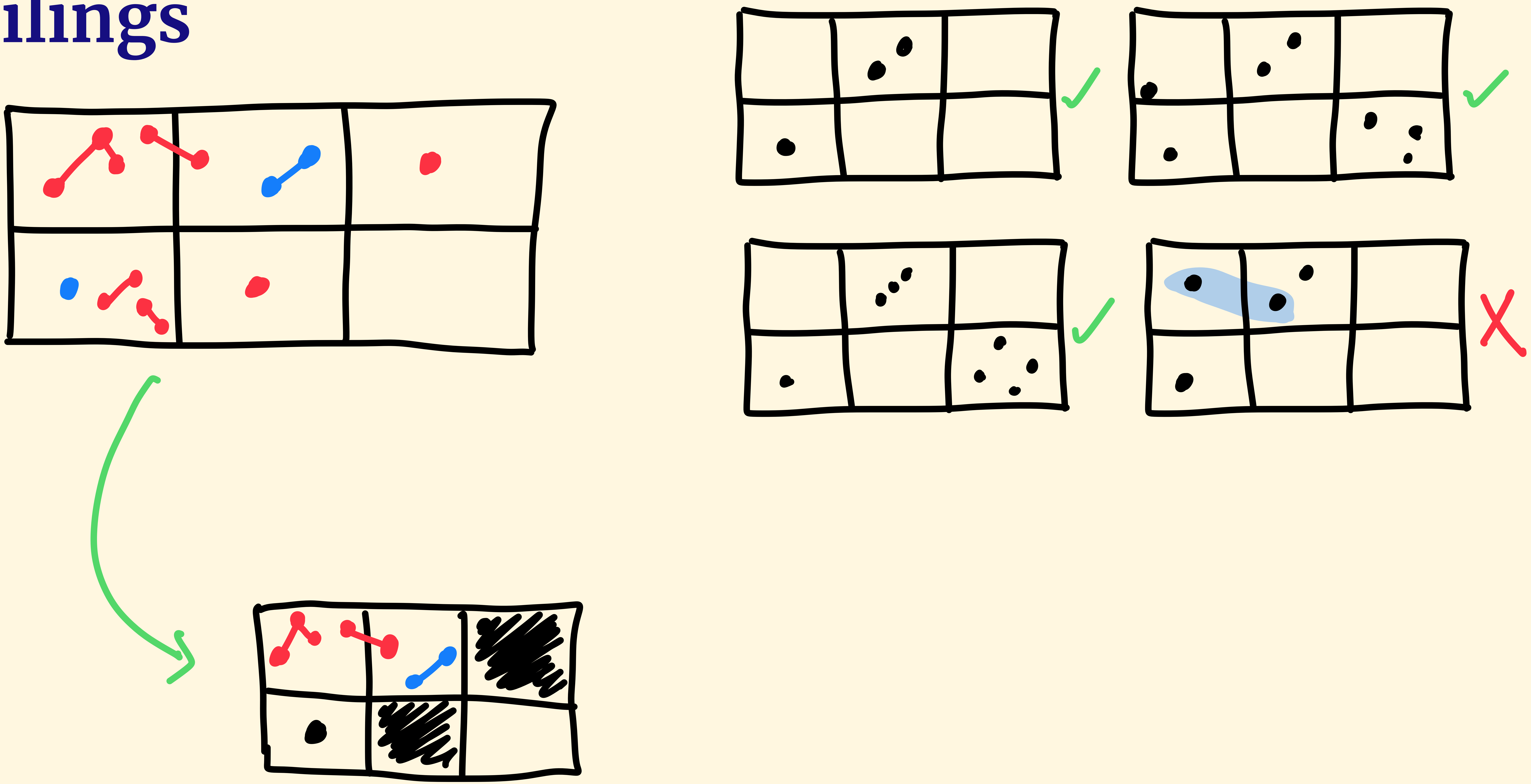




The tiling represents all gridded permutations on a 2x3 grid with:

- exactly one point in the bottom-left cell
- no points in the bottom-middle or top-right cells
- no 132 pattern in the top left cell
- no crossing 21 pattern between the top-left and top-middle cells
- contains a 12 pattern in the top-middle cell

Tilings



Tilings

We can generate the insertion encoding graph using tilings instead of slot configurations!

Tilings

We can generate the insertion encoding graph using tilings instead of slot configurations!

Each tiling represents a set of permutations just like each insertion encoding configurations represents the set of permutations that can be generated from that configuration.

Tilings

We can generate the insertion encoding graph using tilings instead of slot configurations!

Each tiling represents a set of permutations just like each insertion encoding configurations represents the set of permutations that can be generated from that configuration.

It is a fast operation to “place an entry into a slot” on a tiling and simplify the obstructions.

Tilings

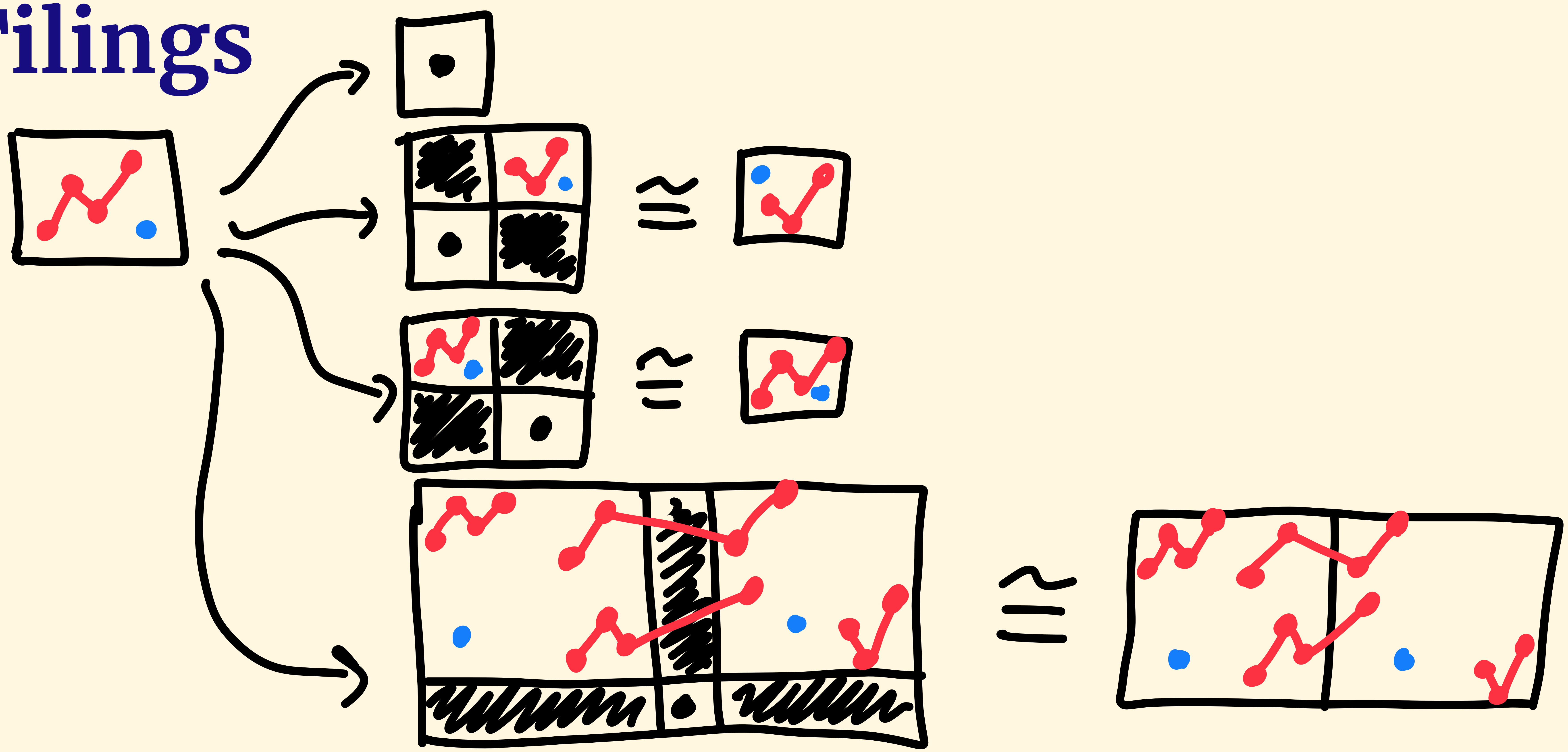
We can generate the insertion encoding graph using tilings instead of slot configurations!

Each tiling represents a set of permutations just like each insertion encoding configurations represents the set of permutations that can be generated from that configuration.

It is a fast operation to “place an entry into a slot” on a tiling and simplify the obstructions.

No expensive checks, just like the link patterns in 1324, but we didn’t need to first describe and prove any structure by hand.

Tilings



We can remove the points and only use the top row because the obstructions already keep track of where the bad patterns can show up.

Two states are isomorphic when they are simply the same tiling. For the original insertion encoding this was a very expensive check.

Tilings

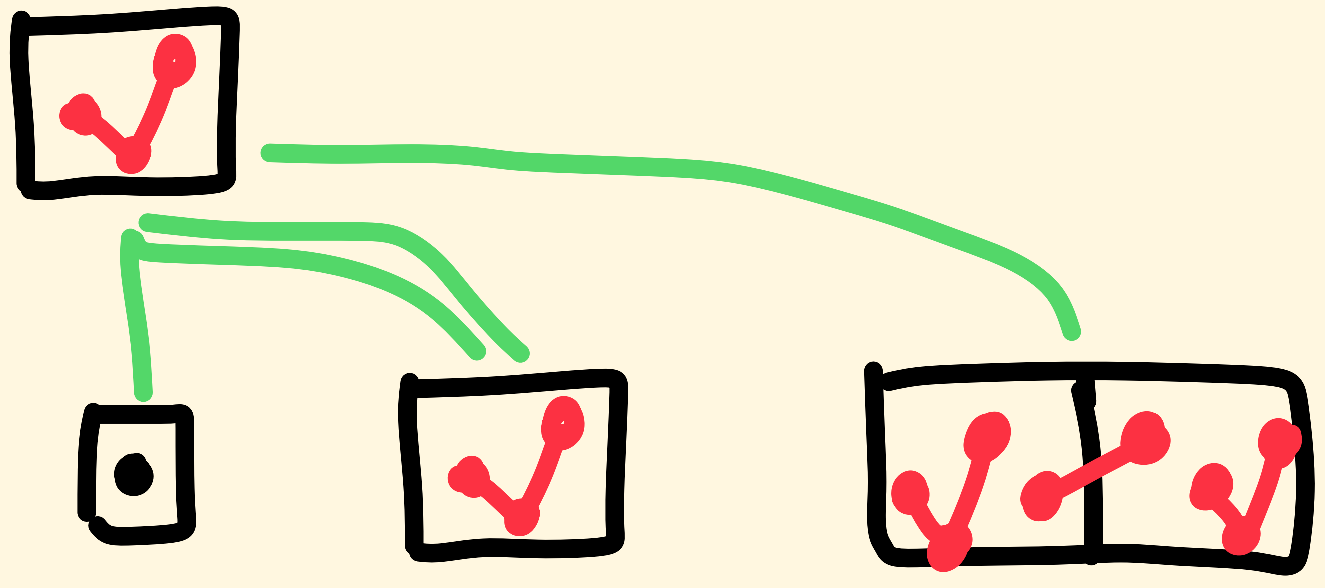
None of this is specific to 1324, and we can do it with any set of forbidden patterns.

Tilings

Av(213)

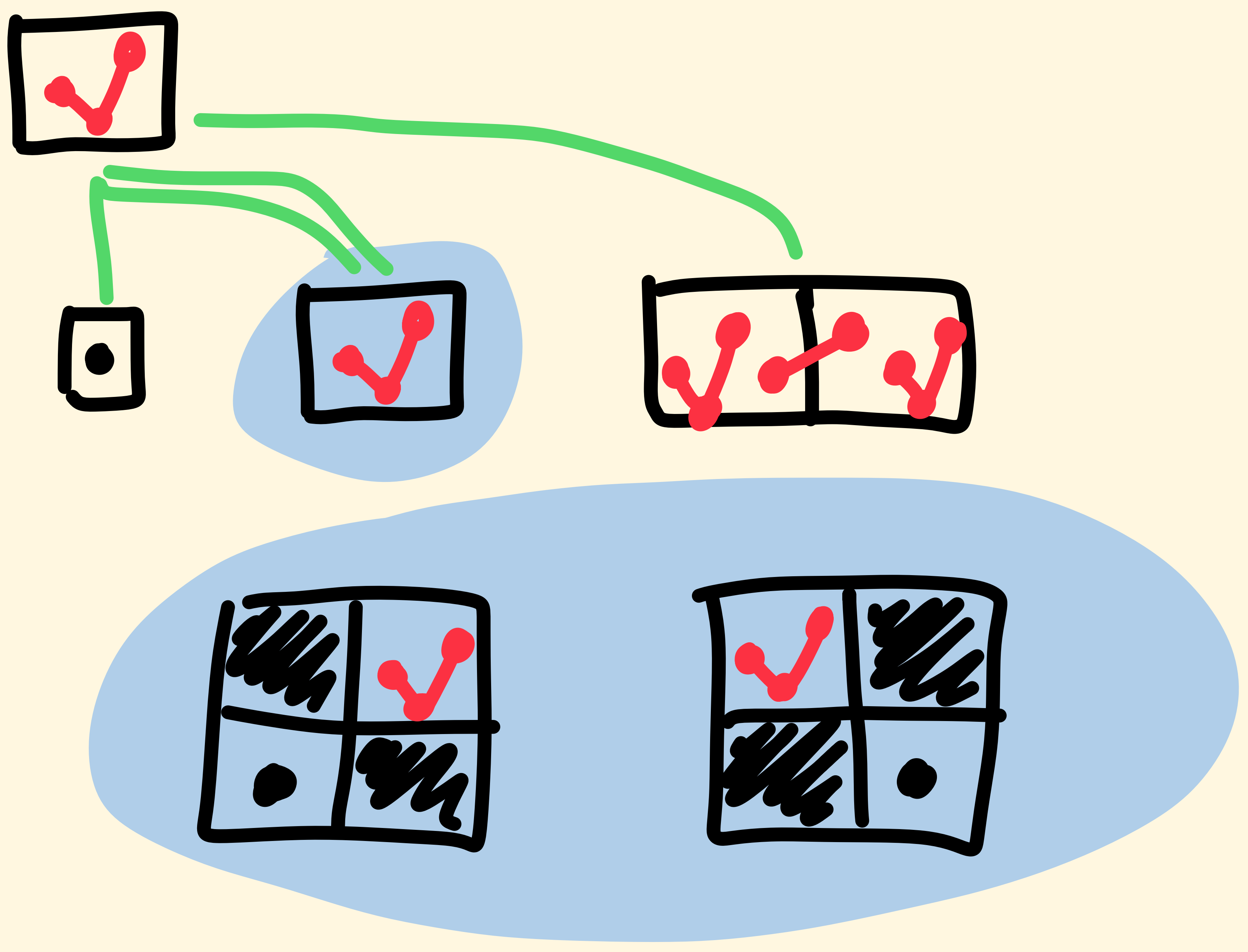
Tilings

Av(213)



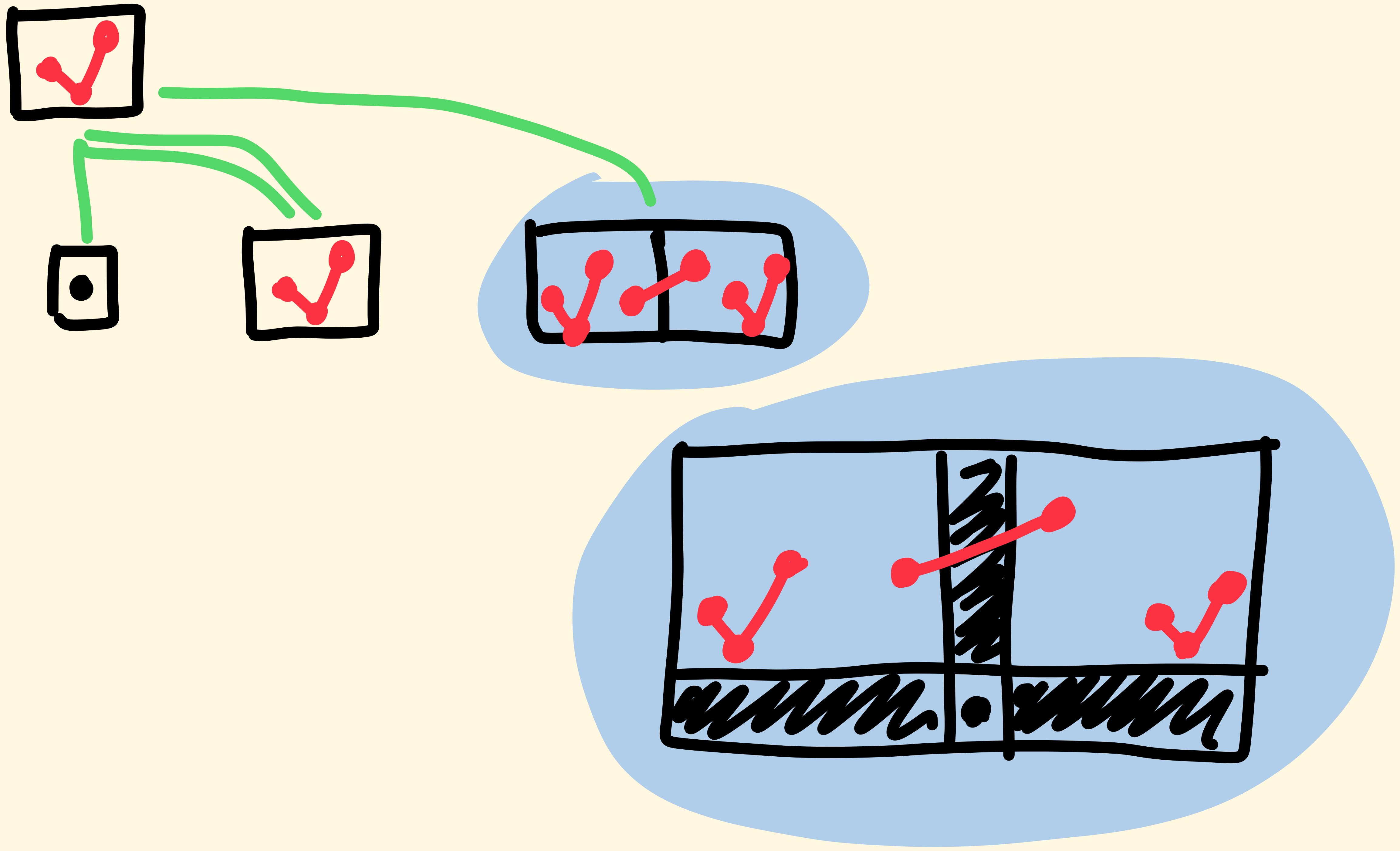
Tilings

Av(213)



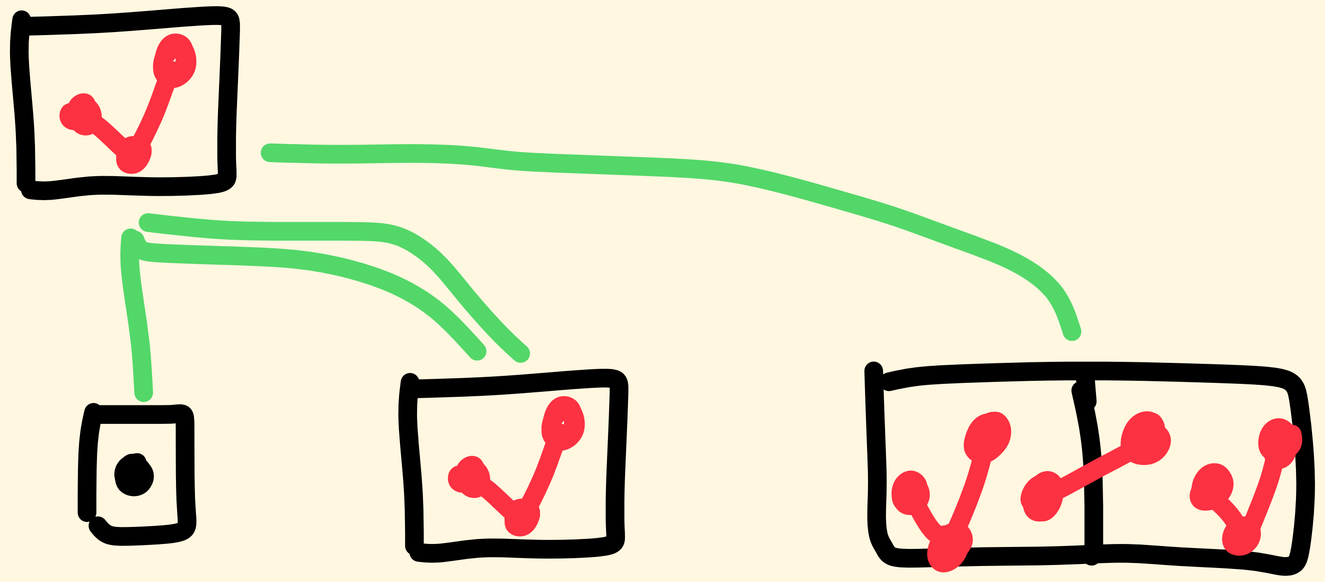
Tilings

Av(213)



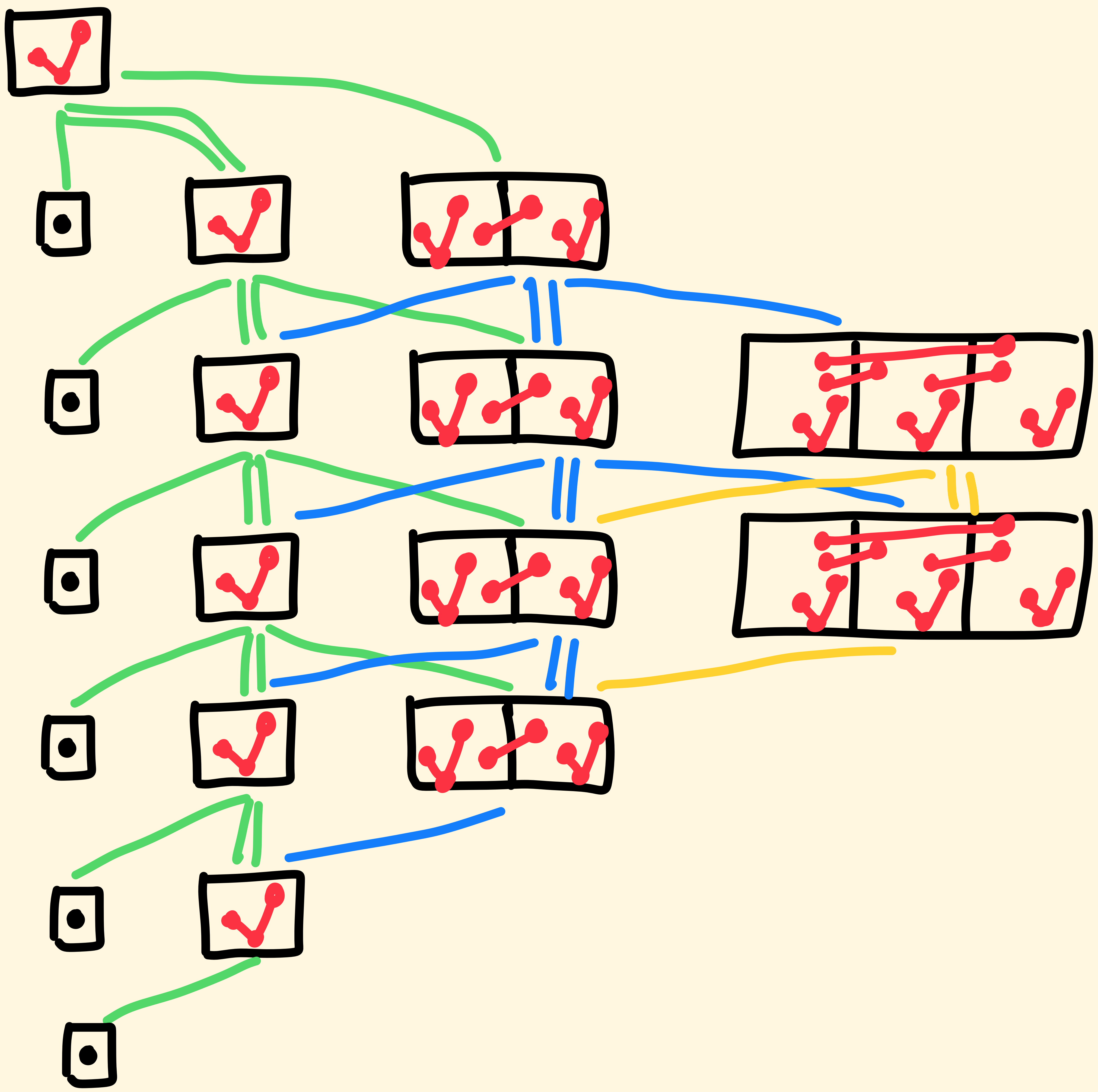
Tilings

Av(213)



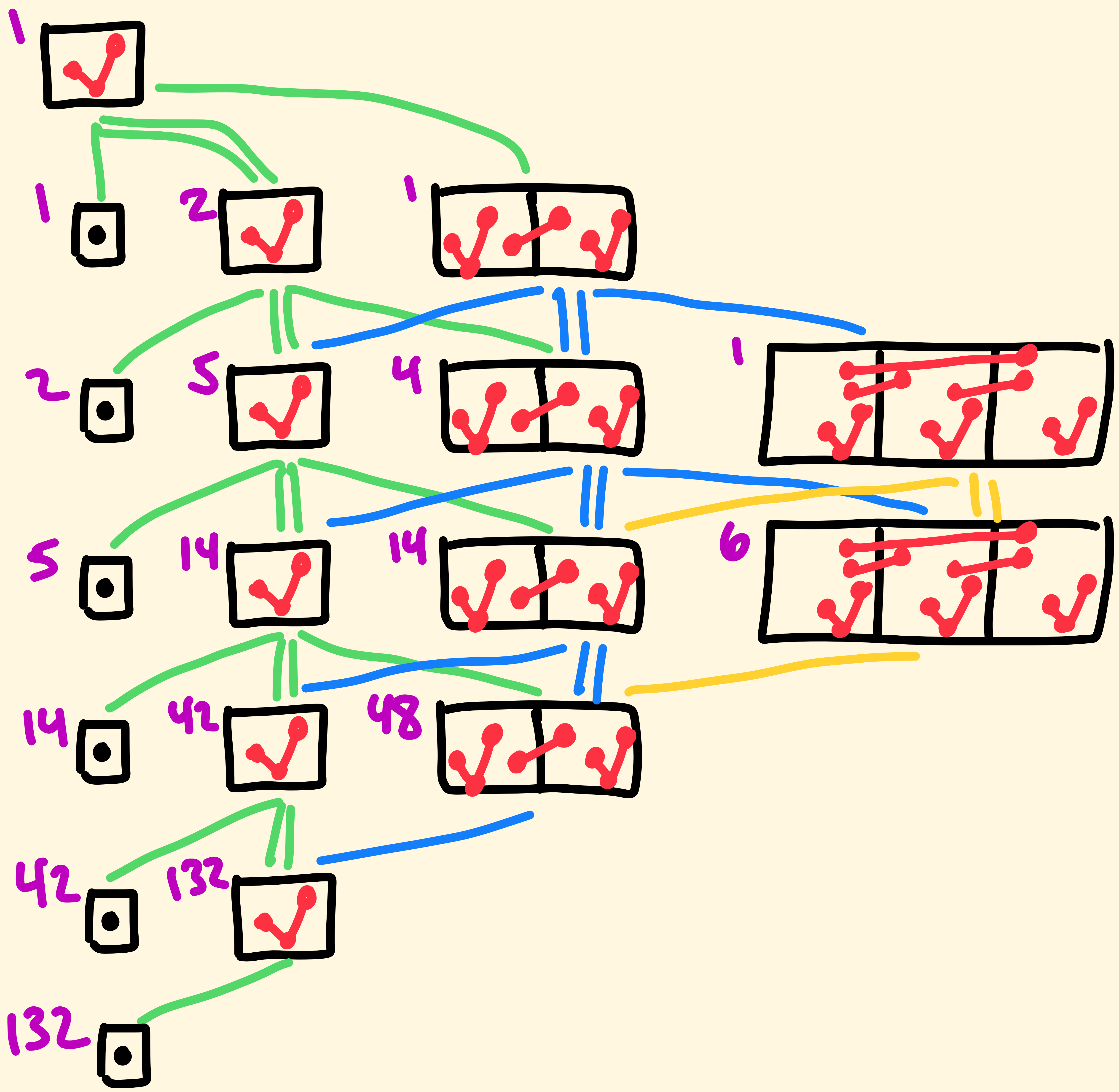
Tilings

Av(213)



Tilings

Av(213)



Results

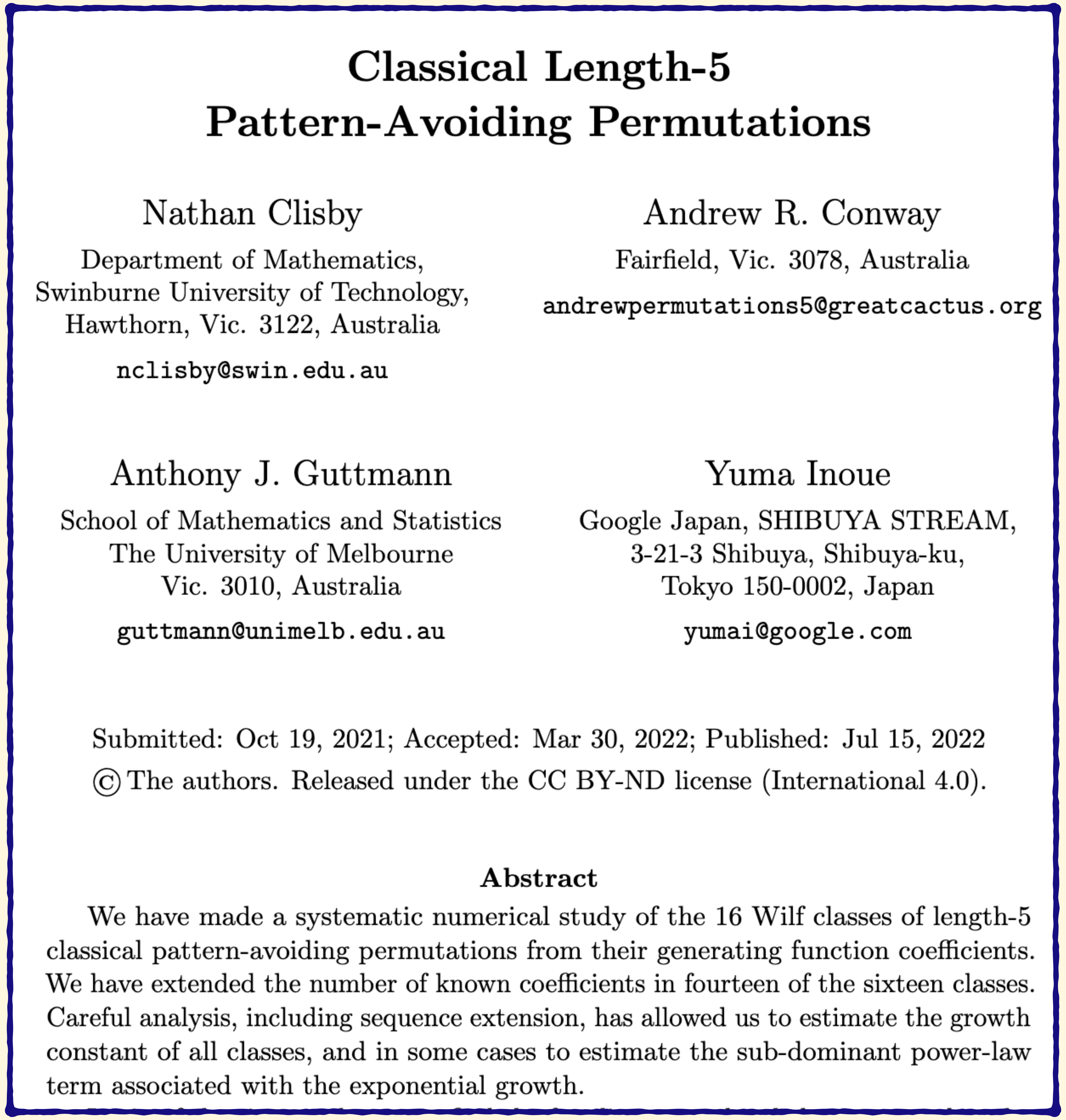
Very preliminary — still improving the parallel implementation

Definitely does not beat 50 terms of Av(1324)!

Since this is general purpose, it doesn't "know" a structural theorem like the link patterns ahead of time.

But, I can get to the mid-30s on my laptop and into the 40s on a larger machine.

Results



Classical Length-5 Pattern-Avoiding Permutations

Nathan Clisby
Department of Mathematics,
Swinburne University of Technology,
Hawthorn, Vic. 3122, Australia
nclisby@swin.edu.au

Andrew R. Conway
Fairfield, Vic. 3078, Australia
andrewpermutations5@greatcactus.org

Anthony J. Guttmann
School of Mathematics and Statistics
The University of Melbourne
Vic. 3010, Australia
guttmann@unimelb.edu.au

Yuma Inoue
Google Japan, SHIBUYA STREAM,
3-21-3 Shibuya, Shibuya-ku,
Tokyo 150-0002, Japan
yumai@google.com

Submitted: Oct 19, 2021; Accepted: Mar 30, 2022; Published: Jul 15, 2022

Abstract

We have made a systematic numerical study of the 16 Wilf classes of length-5 classical pattern-avoiding permutations from their generating function coefficients. We have extended the number of known coefficients in fourteen of the sixteen classes. Careful analysis, including sequence extension, has allowed us to estimate the growth constant of all classes, and in some cases to estimate the sub-dominant power-law term associated with the exponential growth.

There are 120 classes of the form Av(β) where $|\beta| = 5$. They split into 16 different groups based on their counting sequence.

One is already solved, one independently counted up to length 38, and this paper computed the other 14 up to lengths between 23 and 27.

Our method looks like to get most of the 14 up to length 30, some up to 35 or 40.

Efficiency varies a lot between classes. The number of different tilings computed could be exponential, polynomial, even linear.

Results

Contents lists available at ScienceDirect

Journal of Combinatorial Theory, Series A

www.elsevier.com/locate/jcta

ELSEVIER

Generating permutations with restricted containers

Michael H. Albert [a], Cheyne Homberger [b], Jay Pantone [c,1], Nathaniel Shar [d], Vincent Vatter [e,1]

[a] *Department of Computer Science, University of Otago, Dunedin, New Zealand*
[b] *Department of Mathematics, University of Maryland Baltimore County, Baltimore, MD, USA*
[c] *Department of Mathematics, Dartmouth College, Hanover, NH, USA*
[d] *Department of Mathematics, Rutgers University, New Brunswick, NJ, USA*
[e] *Department of Mathematics, University of Florida, Gainesville, FL, USA*

A R T I C L E   I N F O

Article history:
Received 12 July 2016
Available online 7 March 2018

Keywords:
Permutation patterns
Enumeration
Stack
Sorting machine

A B S T R A C T

We investigate a generalization of stacks that we call $\mathcal{C}$-machines. We show how this viewpoint rapidly leads to functional equations for the classes of permutations that $\mathcal{C}$-machines generate, and how these systems of functional equations can be iterated and sometimes solved. General results about the rationality, algebraicity, and the existence of Wilfian formulas for some classes generated by $\mathcal{C}$-machines are given. We also draw attention to some relatively small permutation classes which, although we can generate thousands of terms of their counting sequences, seem to not have D-finite generating functions.

Av(1432, 1324)

- up to length 100 on my laptop in under 10 minutes
- quadratic number of states per layer

Av(1432, 1243)

- up to length 100 on my laptop in under 10 minutes
- linear number of states per layer

Av(1324, 1234)

- up to length 100 on my laptop in under 2 minutes
- constant number of states per layer

Av(1432, 1324, 1243)

- up to length 100 on my laptop in under 1.5 minutes
- linear number of states per layer

Bounds on the Growth Rate

In addition to the counting sequences, you can also turn these truncated insertion encoding trees into rigorous lower bounds for the growth rate of the class. (maybe upper bounds too?)

Av(12453):

growth rate is known to be $9 + 4\sqrt{2} \approx 14.6568$

we get a lower bound of 13.3748 by counting up to length 30

Av(41235):

Tony estimates the growth rate is ≈ 13.703 using 27 terms

we get a lower bound of 12.1619 by counting up to length 27

Other Avenues

We have adapted this to count pattern-avoiding involutions, and applied it to the patterns 1324 and 4231. Forthcoming paper with Christian Bean and Tony.

Christian and I have also adapted it to count pattern-avoiding inversion sequences. You can really do this for any combinatorial object that you can make a tiling-like object for.

Happy Birthday Tony!