Jack Church's overview of the Merge-Insert Sort Algorithm.
Alternativy called the Ford_Johnson algorithm.

Sources:

(i) The Art of Computer Programming, volume 3, Sorting and Searching, Second Edition.
Pages 183, 184 and 185.
Donald E. Knuth, 1998

(ii) Emuminov, 2025
https://dev.to/emuminov/human-explanation-and-step-by-step-visualisation-of-the-ford-johnson-algorithm-5g91

Lets sort 21 numbers:
52 66 15 86 51 38 18 5 61 11 40 23 2 45 39 42 14 48 36 7 85

Big step 1: Merging into pairs.

Break in into pairs half or set of 2:
The 85 at the end has no pair. Just keep it safe for now.

52 66	15 86	51 38	18 5	61 11	40 23	2 45	39 42	14 48	36 7	85

Sort the number within each pair, lowest to largest: highlighted the swaps:

52 66	15 86	38 51	5 18	11 61	23 40	2 45	39 42	14 48	7 36	85

Sort each pair of pairs based on the last number of each pair.

52 66	15 86	5 18	38 51	23 40	11 61	39 42	2 45	7 36	14 48	85

Make Pairs of pairs or set of 4:

52 66 15 86	5 18 38 51	23 40 11 61	39 42 2 45	7 36 14 48	85

Sort the pairs again based on the last number:
51 < 86 and 45 < 61 so swap both sets of pairs.
The 7 36 14 48 have no pair of 4. Just keep them safe for now.

5 18 38 51	52 66 15 86	39 42 2 45	23 40 11 61	7 36 14 48	85

Do it again, make another pair or pairs or set of 8:
7 36 14 48 can't form a pair at this level, just keep them safe for now.

5 18 38 51 52 66 15 86	39 42 2 45 23 40 11 61	7 36 14 48	85

Sort the pairs based on the last number in each pair: 61 < 86 so swap:

39 42 2 45 23 40 11 61	5 18 38 51 52 66 15 86	7 36 14 48	85

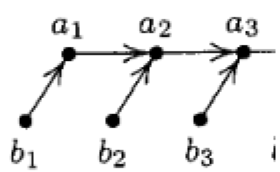
We don't have enough numbers for 2 sets of 16.
Notice how each individual number pair, the number on the left (bold) is smaller than the right? 39<42, 2<45, 23<40, etc

39 42 **2** 45 **23** 40 **11** 61	**5** 18 **38** 51 **52** 66 **15** 86	**7** 36 **14** 48	85

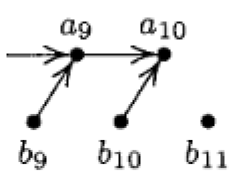
Big step 2: Undo the pairs and insert at the same time:
So where we have our set of 8 with some spare.
B2 and b3 are not sets of 8 and thus will sit recursion level out.

39 42 2 45 23 40 11 61	5 18 38 51 52 66 15 86	7 36 14 48	85
b1	a1	b2	b3

Let's try and make this representation of our number sets:



We don't have at the moment a2 and a3, so ignore them for the moment. Tha means there's no arrow from b2 and b3 to anything, like this example of b11:



Set of 8:
Our data in its "starting" sequence.

39 42 2 45 23 40 11 61	5 18 38 51 52 66 15 86	7 36 14 48	85
b1	a1	b2	b3

Imagine an arrow pointing from b1 to a1 like in the picture above.I will use colours instead:

At the set of 8, b2 and b3 don’t have any arrows.

Main	5 18 38 51 52 66 15 86
Main	a1

Pending merge	39 42 2 45 23 40 11 61	7 36 14 48	85
Pending merge	b1	b2	b3

B1’s last number is smaller than a1’s last number.
Because our data looks like this: b1->a1 -and b2 and b3 aren't sets of 8- we have nothing to do.

Let’s put the main sequence and the pending merge sequence into its new “starting” sequence:
(which is the same as we had nothing to do).

39 42 2 45 23 40 11 61	5 18 38 51 52 66 15 86	7 36 14 48	85
b1	a1	b2	b3

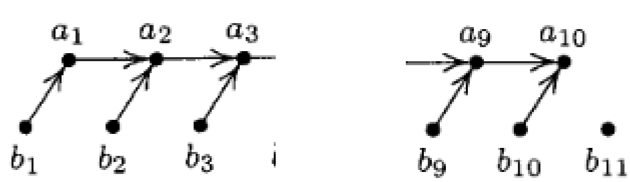
Set of 4:
Turn the 8 set starting sequence into a 4 set starting sequence:
The set numbers (b1, a1, b2, a2) will change/update for this set of 4.

39 42 2 45 23 40 11 61	5 18 38 51 52 66 15 86	7 36 14 48	85

Becomes…

39 42 2 45	23 40 11 61	5 18 38 51	52 66 15 86	7 36 14 48	85
b1	a1	b2	a2	b3	b3

Now recreate the main and pending merge sequences with sets of 4:



Note the last number of each main sequence set: b1 < a1 < a2 and b2 < a2.
B3 and b4 are just floating like b11 in the picture just above.
B4 isn’t a set of 4, thus it will sit this level of recursion out.

Arrows are b1 -> a1 -> a2; and b2->a2.

Main	39 42 2 45	23 40 11 61	52 66 15 86

Main	b1	a1	a2

Pending	5 18 38 51	7 36 14 48	85
Pending	b2	b3	b4

Inserting:
The main are sets of 4 and pend has items that are sets of 4. We can now insert pend into main using Jacobsthal numbers (J). It will make sense why soon.

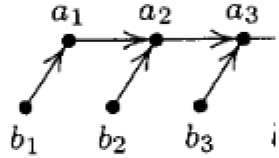
J->Previous + 2(J->Previous->Previous) = J
Given 0
Given 1
 1 + (2 * 0) = 1 + 0   = 1
 1 + (2 * 1) = 1 + 2   = 3
 3 + (2 * 1) = 3 + 2   = 5
 5 + (2 * 3) = 5 + 6   = 11
11 + (2 * 5) = 11 + 10 = 21
21 + (2 * 11) = 21 + 22 = 43
43 + (2* 21) = 43 + 42 = 85

Our set of 4 sequences:

Main	39 42 2 45	23 40 11 61	52 66 15 86
Main	b1	a1	a2

Pending	5 18 38 51	7 36 14 48	85
Pending	b2	b3	b4

Begin with the lowest Jacobsthal number in the pend sequence: it's 1. We will take b1 and merge it into the main sequence. Oh it's already there. What's the next Jacobsthal number? It's 3:
So we merge b3 into the main sequence using the last number in sets as the number to use for checking.



Because we have not a3 (the picture does, but we don't). We must search the entire main sequence.
I.e. we must compare 48(b3) to 45(b1), 61(a1), and 86(a2).

Use binary search:
A1 is half way. Is 48(b3) < 61(a1) ?
Yes.
Is 48(b3) < 45(b1) ?
No.
Insert b3 in-between b1 and a1:

Main	39 42 2 45	7 36 14 48	23 40 11 61	52 66 15 86
Main	b1	b3	a1	a2

Pending	5 18 38 51	7 36 14 48	85
Pending	b2	b3	b4

Then decrement from b3 to b2 and merge this set into the main sequence:
B2 is bound to element A2 and we know that B2 will be smaller than A2; so our search area is everything up to but not including A2.
The search area will be B1, B2 and A1:
Using binary search for the main sequence:
51(b2) is greater than 48(b3).
51(b2) is less than 61(a1).
Insert between b3 and a1.

Main	39 42 2 45	7 36 14 48	5 18 38 51	23 40 11 61	52 66 15 86
Main	b1	b3	b2	a1	a2

Pending	5 18 38 51	85
Pending	b2	b4

We have now sorted at sets of 4 b3 and b2. The previous Jacobsthal number is 1. We stopped at b2. One number above this Jacobsthal number.

Our new staring sequence with sets of 4:

39 42 2 45	7 36 14 48	5 18 38 51	23 40 11 61	52 66 15 86	85
b1	b3	b2	a1	a2	b4

Sets of 2:
Turn the 4 set starting sequence into a 2 set starting sequence:
The set numbers (b1, a1, b2, a2) will change/update for this set of 2.

39 42 2 45	7 36 14 48	5 18 38 51	23 40 11 61	52 66 15 86	85

Becomes…

39 42	2 45	7 36	14 48	5 18	38 51	23 40	11 61	52 66	15 86	85
b1	a1	b2	a2	b3	a3	b4	a4	b5	a5	b6

85(b6) isn't a set of 2. It will need to sit this recursion out.

Now recreate the main and pending merge sequences with sets of 2:

B3 and b4 are just floating like b11 in the picture just above.
B4 isn't a set of 4, thus it will sit this level of recursion out.
Note the last number of each main sequence set:

b1 < a1 < a2 < a3 < a4 < a5 < a6.
a2 < a2
b3 < a3
b4 < a4

Arrows are b1 -> a1 -> a2 -> a3 -> a4 -> a5.
B2 -> a2
B3 -> a3
B4 -> a4
B5 -> a5

The main are sets of 2 and pend has items that are sets of 2. We can now insert pend into main using Jacobsthal numbers (J).

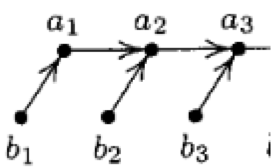
Our set of 4 sequences:

Main	39 42	2 45	14 48	38 51	11 61	15 86

Main	b1	a1	a2	a3	a4	a5

Pending	7 36	5 18	23 40	52 66	85
Pending	b2	b3	b4	b5	b6

Begin with the lowest Jacobsthal number in the pend sequence: it's 1. We will take b1 and merge it into the main sequence. Oh it's already there. What's the next Jacobsthal number? It's 3:
So we merge b3 into the main sequence using the last number in sets as the number to use for checking.



Because of our work before, we are guaranteed that 18(b3) is < 51(a3).
So we can limit our search to below 51(a3).

Use binary search:
A1 and A2 are half way. Use A2 for no apparent reason:
Is 18(b3) < 48(a2) ?
Yes.
Is 18(b3) < 45(a1) ?
Yes.
Is 18(b3) < 42(b1) ?
Yes.
Insert b3 before b1:

Main	5 18	39 42	2 45	14 48	38 51	11 61	15 86
Main	b3	b1	a1	a2	a3	a4	a5

Pending	7 36	5 18	23 40	52 66	85
Pending	b2	b3	b4	b5	b6

Then decrement from b3 to b2 and merge this set into the main sequence:
B2 is bound to element A2 and we know that B2 will be smaller than A2; so our search area is everything up to but not including A2.
The search area will be B3, B1 and A1.

Using binary search for the main sequence:
36(b2) is less than 42(b1).
36(b2) is greater than 18(b3).
Insert between B3 and B1.

Main	5 18	7 36	39 42	2 45	14 48	38 51	11 61	15 86
Main	b3	b2	b1	a1	a2	a3	a4	a5

Pending	7 36	23 40	52 66	85
Pending	b2	b4	b5	b6

We have now sorted at sets of 4 b3 and b2. The previous Jacobsthal number is 1. We stopped at 2 (b2). One number above this lower Jacobsthal number.

We have more items in Pend. Are any the next Jacobsthal number? Yes, 5 (b5). Let's sort b5 into main.

Main	5 18	7 36	39 42	2 45	14 48	38 51	11 61	15 86
Main	b3	b2	b1	a1	a2	a3	a4	a5

Pending	23 40	52 66	85
Pending	b4	b5	b6

B5 is guaranteed to be lower than A5, so our search is from B3 to A4 inclusive:
Binary search halfway is A1.
Is 66(b5) < 45(a1) ?
No.
Is 66(b5) < 51(a3) ?
No.
Is 66(b5) < 61(a4) ?
No.
The next set is A5, but B5 is guaranteed to be less than A5.
So insert B5 in between A4 and A5.

Main	5 18	7 36	39 42	2 45	14 48	38 51	11 61	52 66	15 86
Main	b3	b2	b1	a1	a2	a3	a4	b5	a5

Pending	23 40	52 66	85
Pending	b4	b5	b6

We now work backwards to sort B4 into main. It is guaranteed to be less than A4 so we use binary search from B3 to A3 inclusive:
B1 and A1 are halfway. Use B1 for no particular reason.
Is 40(b4) < 39(b1) ?
No.
Is 40(b4) < 45(a1) ?
Yes.
Insert b4 in between b1 and a1.

Main	5 18	7 36	39 42	23 40	2 45	14 48	38 51	11 61	52 66	15 86
Main	b3	b2	b1	b4	a1	a2	a3	a4	b5	a5

Pending	23 40	85
Pending	b4	b6

Main and Pend sequences now look like this:

Main	5 18	7 36	39 42	23 40	2 45	14 48	38 51	11 61	52 66	15 86
Main	b3	b2	b1	b4	a1	a2	a3	a4	b5	a5

Pending	85
Pending	b6

We have sorted all the sets of 4 and iterated to the previous Jacobsthal number of 3 (b3).
The new starting sequence with sets of 2 sorted:

5 18	7 36	39 42	23 40	2 45	14 48	38 51	11 61	52 66	15 86	85
b3	b2	b1	b4	a1	a2	a3	a4	b5	a5	b6

Sets of 1:
Let's turn it into sets of 1:
Turn the 2 set starting sequence into a 1 set starting sequence:
The set numbers (b1, a1, b2, a2) will change/update for this set of 1.

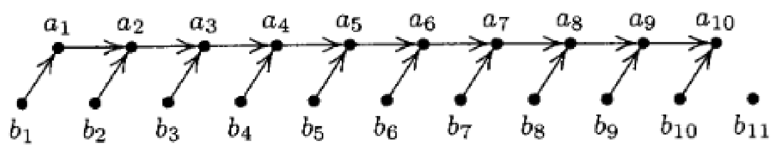
5 18	7 36	39 42	23 40	2 45	14 48	38 51	11 61	52 66	15 86	85

Becomes…

5	18	7	36	39	42	23	40	2	45	14	48	38	51	11	61	52	66	15	86	85
b1	a1	b2	a2	b3	a3	b4	a4	b5	a5	b6	a6	b7	a7	b8	a8	b9	a9	b10	a10	b11

85(b6) is a set of 1. It will be used in this set. Finally.

Now recreate the main and pending merge sequences with sets of 1:



B3 and b4 are just floating like b11 in the picture just above.
B4 isn't a set of 4, thus it will sit this level of recursion out.
Note the last number of each main sequence set:

b1 < a1 < a2 < a3 < a4 < a5 < a6 < a7 < a8 < a9 < a10. .
a2 < a2
b3 < a3
b4 < a4
Etc…

Arrows are b1 -> a1 -> a2 -> a3 -> a4 -> a5 -> a6 -> a7 -> a8 -> a9 -> a10.
B2 -> a2
B3 -> a3
B4 -> a4
Etc…
B11 has no arrows to point to anything.

The main are sets of 1 and pend has items that are sets of 1. We can now insert pend into main using Jacobsthal numbers (J).

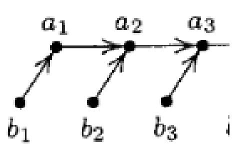
5	18	7	36	39	42	23	40	2	45	14	48	38	51	11	61	52	66	15	86	85
b1	a1	b2	a2	b3	a3	b4	a4	b5	a5	b6	a6	b7	a7	b8	a8	b9	a9	b10	a10	b11

Our set of 1 sequences:

Main	5	18	36	42	40	45	48	51	61	66	86
Main	b1	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10

Pend	7	39	23	2	14	38	11	52	15	85
Pend	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11

Jacobsthal Number 1.
Begin with the lowest Jacobsthal number in the pend sequence: it's 1. We will take b1 and merge it into the main sequence. Oh it's already there. What's the next **Jacobsthal number? It's 3:**
So we merge b3 into the main sequence using the last number in sets as the number to use for checking.



Jacobsthal Number 3.
Because of our work before, we are guaranteed that 39(b3) is < 42(a3).
So we can limit our search to below 42(a3).
Bold text indicates search area.
Use binary search and insert 39(b3) into main in between A2 and A3.

Main	**5**	**18**	**36**	39	42	40	45	48	51	61	66	86
Main	**b1**	**a1**	**a2**	b3	a3	a4	a5	a6	a7	a8	a9	a10

Pend	7	39	23	2	14	38	11	52	15	85
Pend	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11

Then decrement from B3 to B2 and merge this set into the main sequence:
B2 is bound to element A2 and we know that B2 will be smaller than A2; so our search area is everything up to but not including A2.

The search area will be B1, B1 and A2.
Bold text indicates search area.

Use binary search and insert 7(b2) into main in between B12 and A1.

Main	**5**	7	**18**	36	39	42	40	45	48	51	61	66	86
Main	**b1**	b2	**a1**	a2	b3	a3	a4	a5	a6	a7	a8	a9	a10

Pend	7	23	2	14	38	11	52	15	85
Pend	b2	b4	b5	b6	b7	b8	b9	b10	b11

We have sorted from Jacobsthal number 3 down to 2. The previous Jacobsthal number is 1. We stopped at 2. This is 1 number above this lower Jacobsthal number of 1.

We have more items in Pend. Are any the next **Jacobsthal number? Yes, 5** (b5). Let's sort b5 into main.
Let's reset the colours.

Main	5	7	18	36	39	42	40	45	48	51	61	66	86
Main	b1	b2	a1	a2	b3	a3	a4	a5	a6	a7	a8	a9	a10

Pend	23	2	14	38	11	52	15	85
Pend	b4	b5	b6	b7	b8	b9	b10	b11

B5 is guaranteed to be lower than A5, so our search is from B3 to A4 inclusive:
Use binary search to insert 2(b5) into main before 5(b1).
Search bounds are bolded.

Main	2	**5**	**7**	**18**	**36**	**39**	**42**	**40**	45	48	51	61	66	86
Main	b5	**b1**	**b2**	**a1**	**a2**	**b3**	**a3**	**a4**	a5	a6	a7	a8	a9	a10

Pend	23	2	14	38	11	52	15	85
Pend	b4	b5	b6	b7	b8	b9	b10	b11

We now work backwards to sort B4 into main. It is guaranteed to be less than A4 so we use binary search
from B3 to A3 inclusive:

Search bounds are bolded.
Insert b4 in between A1 and A2:

Main	2	**5**	**7**	**18**	23	**36**	**39**	**42**	40	45	48	51	61	66	86
Main	b5	**b1**	**b2**	**a1**	b4	**a2**	**b3**	**a3**	a4	a5	a6	a7	a8	a9	a10

Pend	23	14	38	11	52	15	85
Pend	b4	b6	b7	b8	b9	b10	b11

We have sorted from Jacobsthal number 5 down to 4. The previous Jacobsthal number is 3. We stopped at 4. This is 1 number above this lower Jacobsthal number of 3.

We have more items in Pend. Are any the next **Jacobsthal number? Yes, 11** (b11). Let's sort b11 into main.

Let's clear the colours too.

B11 is guaranteed to be lower than A11, which doesn't exist, so our search is all of the main sequence.
Search area is bold text. Everything.
85(b11) into main using binary search:
Insert in between 66(a9) and 86(a10):

Main	**2**	**5**	**7**	**18**	**23**	**36**	**39**	**42**	**40**	**45**	**48**	**51**	**61**	**66**	85	**86**
Main	**b5**	**b1**	**b2**	**a1**	**b4**	**a2**	**b3**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	b11	**a10**

Pend	14	38	11	52	15	85
Pend	b6	b7	b8	b9	b10	b11

Sort B10 using binary search.
B10 is guaranteed to be lower than A10. Our search is limited to upper bounds of A10 which happens to be the whole of the main sequence.
Search area is bolded. (Everything again)
Insert in between 66(a9) and 86(a10):

Main	**2**	**5**	**7**	15	**18**	**23**	**36**	**39**	**42**	**40**	**45**	**48**	**51**	**61**	**66**	**85**	**86**
Main	**b5**	**b1**	**b2**	b10	**a1**	**b4**	**a2**	**b3**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	**b11**	**a10**

Pend	14	38	11	52	15
Pend	b6	b7	b8	b9	b10

Sort B9 using binary search.
B10 is guaranteed to be lower than A9. Our search is limited to upper bounds of A9 which excludes A9, B11 and A10.
Search area is bolded.
Insert in between 66(a9) and 86(a10):

Main	**2**	**5**	**7**	15	**18**	**23**	**36**	**39**	**42**	**40**	**45**	**48**	**51**	52	**61**	66	85	86
Main	**b5**	**b1**	**b2**	b10	**a1**	**b4**	**a2**	**b3**	**a3**	**a4**	**a5**	**a6**	**a7**	b9	**a8**	a9	b11	a10

Pend	14	38	11	52
Pend	b6	b7	b8	b9

Clear the colours again:

Sort B8 using binary search.
B8 is guaranteed to be lower than A8. Our search is limited to upper bounds of A8 which excludes A8, A9, B11 and A10.
Search area is bolded.
Insert in between 7(b2) and 15(b10):

Main	**2**	**5**	**7**	11	**15**	**18**	**23**	**36**	**39**	**42**	**40**	**45**	**48**	**51**	**52**	**61**	66	85	86
Main	**b5**	**b1**	**b2**	b8	**b10**	**a1**	**b4**	**a2**	**b3**	**a3**	**a4**	**a5**	**a6**	**a7**	**b9**	**a8**	a9	b11	a10

Pend	14	38	11
Pend	b6	b7	b8

Sort B7 using binary search.
B7 is guaranteed to be lower than A7. Our search is limited to upper bounds of A7 which excludes A7, B9, A8, A9, B11 and A10.
Insert in between 36(a2) and 39(b3):

Main	**2**	**5**	**7**	**11**	**15**	**18**	**23**	**36**	38	**39**	**42**	**40**	**45**	**48**	**51**	**52**	61	66	85	86
Main	**b5**	**b1**	**b2**	**b8**	**b10**	**a1**	**b4**	**a2**	b7	**b3**	**a3**	**a4**	**a5**	**a6**	**a7**	**b9**	a8	a9	b11	a10

Pend	14	38
Pend	b6	b7

Sort B6 using binary search.
B6 is guaranteed to be lower than A6. Our search is limited to upper bounds of A6 which excludes A6, A7, B9, A8, A9, B11 and A10.
Search area is bolded.
Insert in between 36(a2) and 39(b3):

Main	**2**	**5**	**7**	**11**	14	**15**	**18**	**23**	**36**	**38**	**39**	**42**	**40**	**45**	48	51	52	61	66	85	86
Main	**b5**	**b1**	**b2**	**b8**	b6	**b10**	**a1**	**b4**	**a2**	**b7**	**b3**	**a3**	**a4**	**a5**	a6	a7	b9	a8	a9	b11	a10

Pend	14
Pend	b6

The Pend sequence is now empty and we are at the end of the final iteration (singles).

Main	2	5	7	11	14	15	18	23	36	38	39	42	40	45	48	51	52	61	66	85	86
Main	b5	b1	b2	b8	b6	b10	a1	b4	a2	b7	b3	a3	a4	a5	a6	a7	b9	a8	a9	b11	a10

Pend	
Pend	

The final sorted result via the **Ford-Johnson** Merge-Insert Sort Algorithm with Jacobsthal numbers:

2	5	7	11	14	15	18	23	36	38	39	42	40	45	48	51	52	61	66	85	86