MAXIMUM PRIZE

COURSE WORK #01

Author: prof. Yevhenii Borodavka

PROBLEM STATEMENT

You have received a money prize of **N** USD (10<=**N**<=999999). You can increase or decrease the prize by doing exactly **K** (0<=**K**<=10^9) digit permutations in this number (one permutation is swapping any of the two digits in the number). Find the maximum prize after **K** permutations.

Input. A single string with two numbers **N** and **K** divided by spaces.

Output. The maximum prize after **K** permutations.

Example. **N**=123, **K**=2

Input: 123 2

Output: 312

HEAP'S ALGORITHM

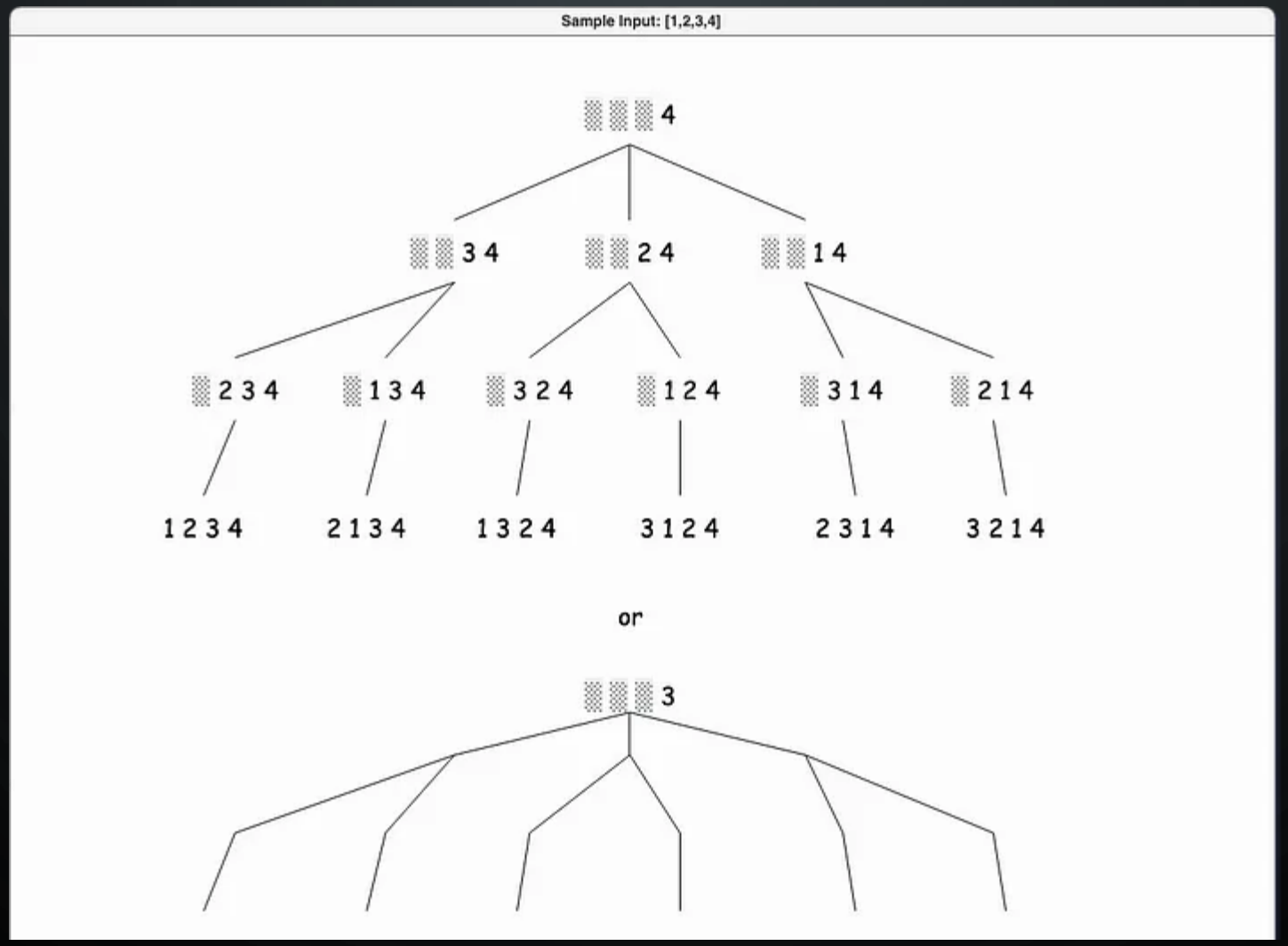
Heap's algorithm generates all possible permutations of n objects. It was first proposed by B. R. Heap in 1963. The algorithm minimizes movement: it generates each permutation from the previous one by interchanging a single pair of elements; the other n-2 elements are not disturbed.

Example:

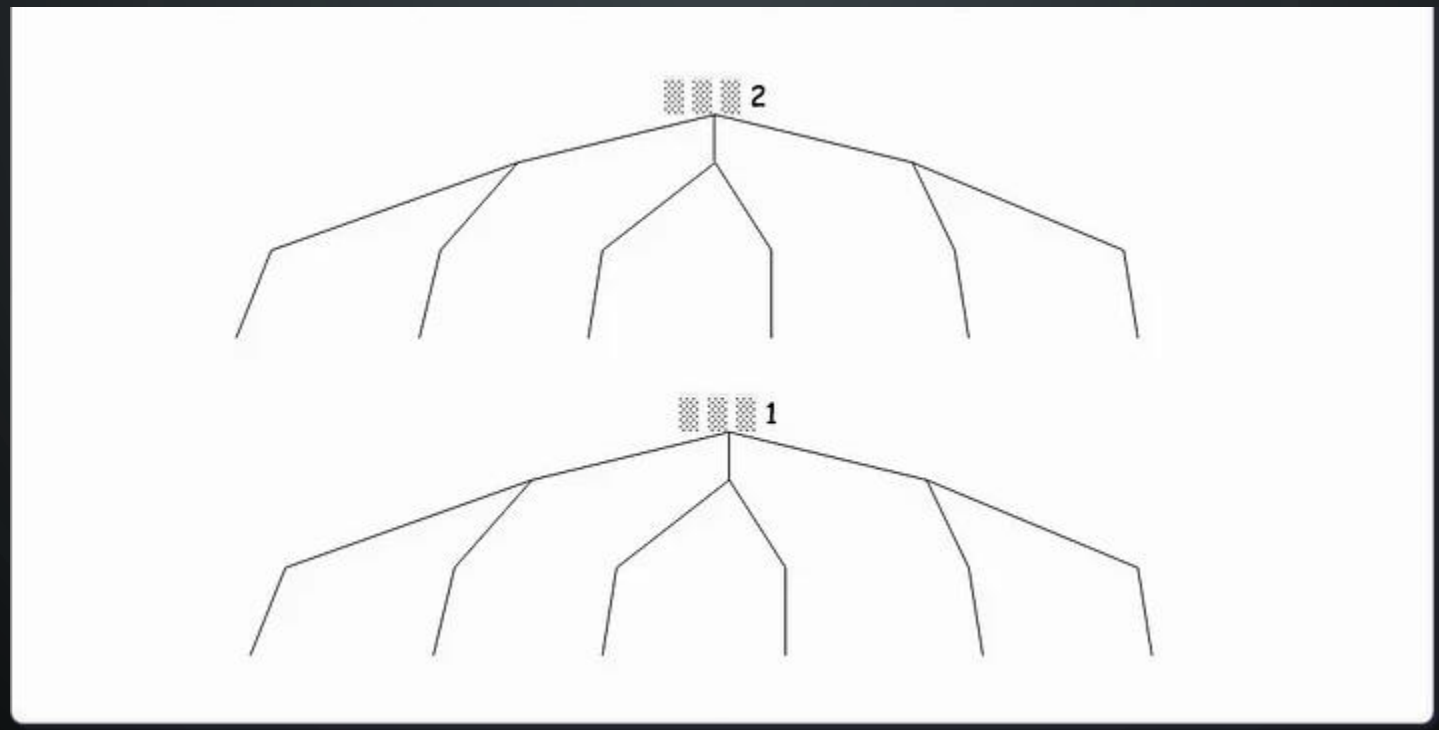
Given an integer array: [1,2,3,4]

From Probability, we know there would be **4!** permutations (results are order sensitive). If you expend the permutation in a tree form, it would look like as follow:

HEAP'S ALGORITHM

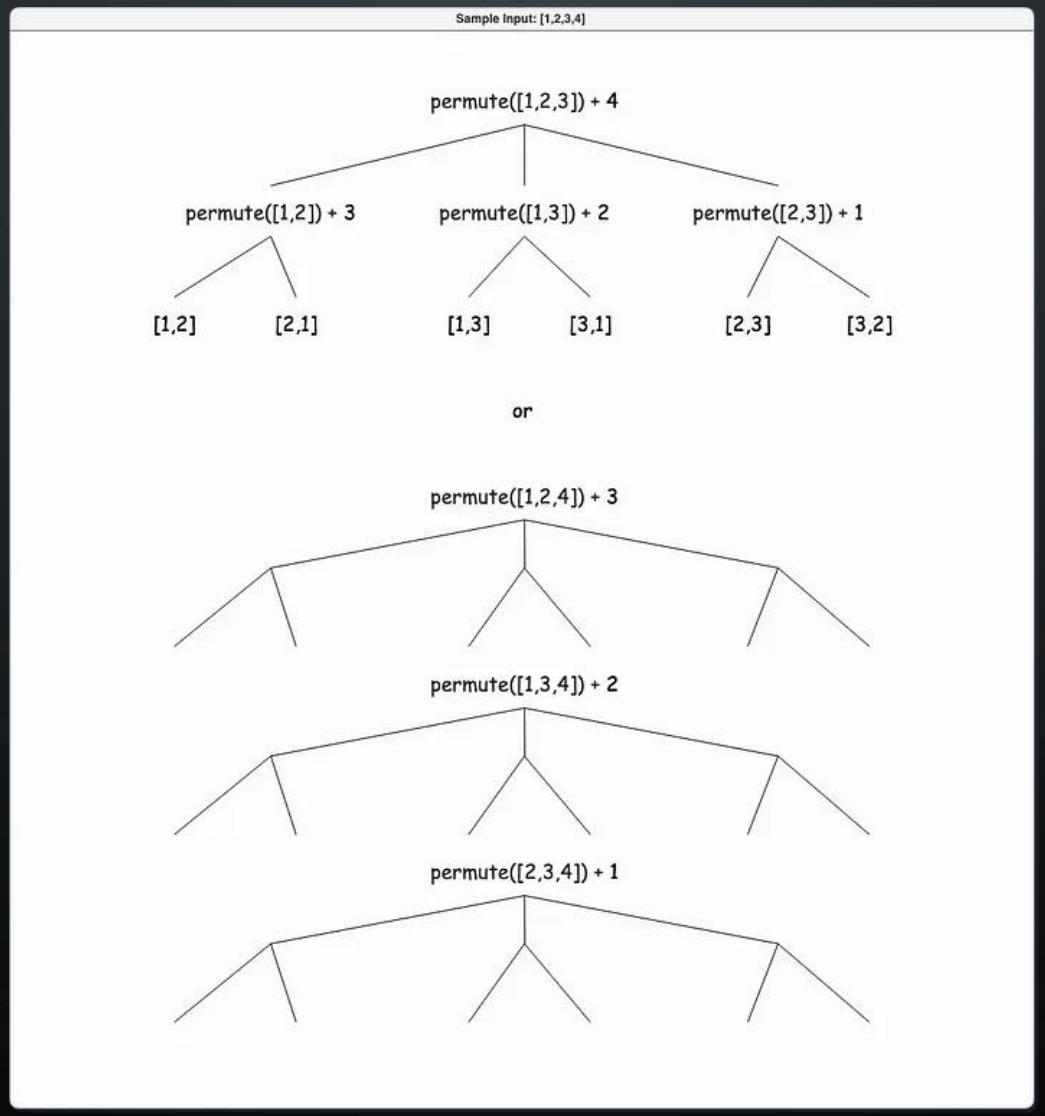


HEAP'S ALGORITHM



The above tree form denotes there is a recursion required to enumerate all the permutations.

HEAP'S ALGORITHM



HEAP'S ALGORITHM

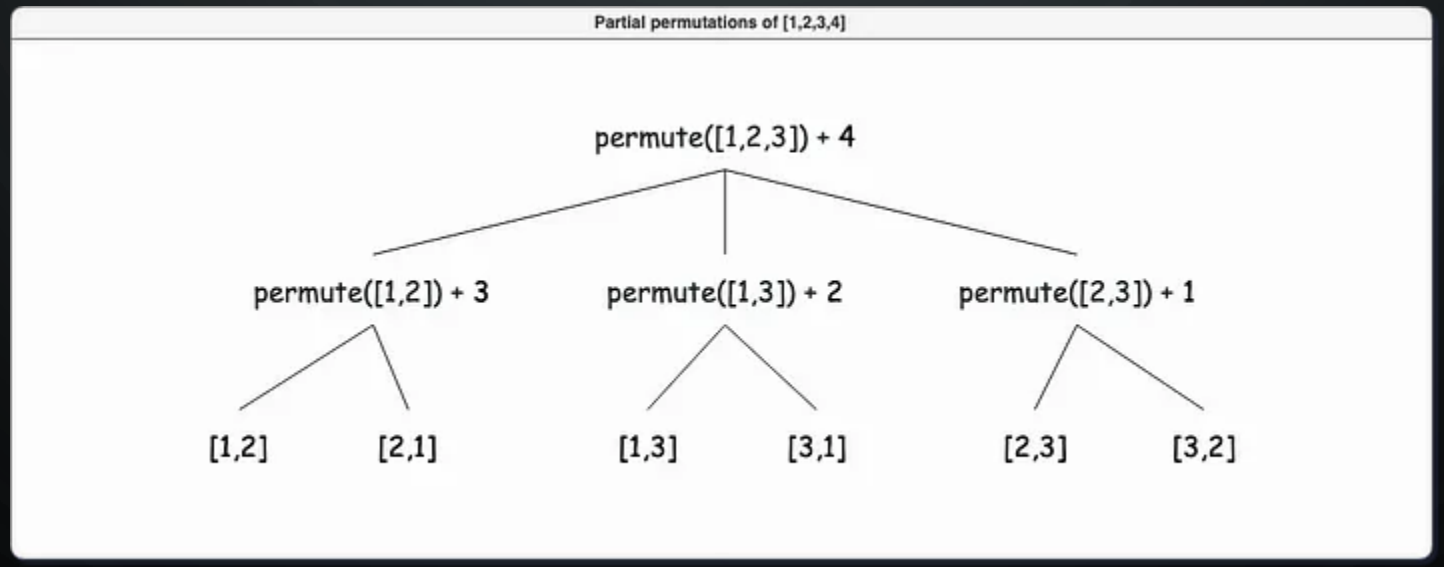
The algorithm follows the decrease-and-conquer strategy:

1. Isolate the end element and permute the K-1 elements.
2. Pick the next unique element and swap with the end element.
3. Repeat step 1 and 2 until all the elements became the end element once!

Alright, we notice the conditional swap might have something to do with picking the next unique element.

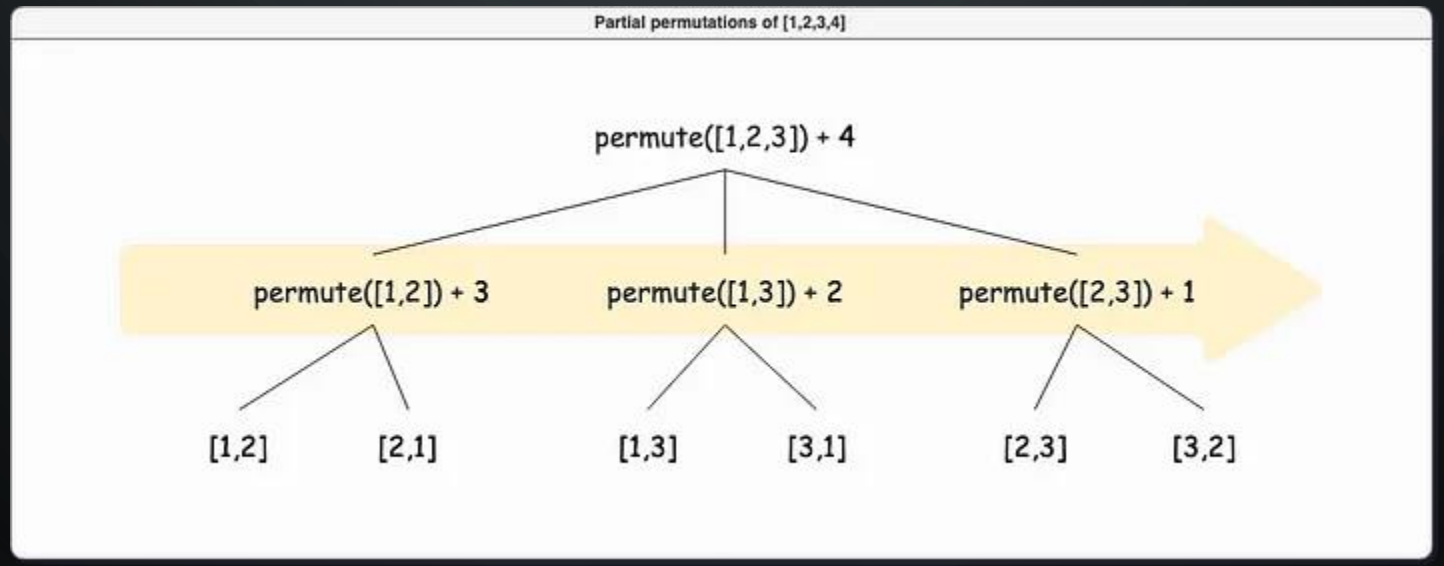
HEAP'S ALGORITHM

Let's see the partial answer of the given input [1,2,3,4] as the permutations ending with 4.



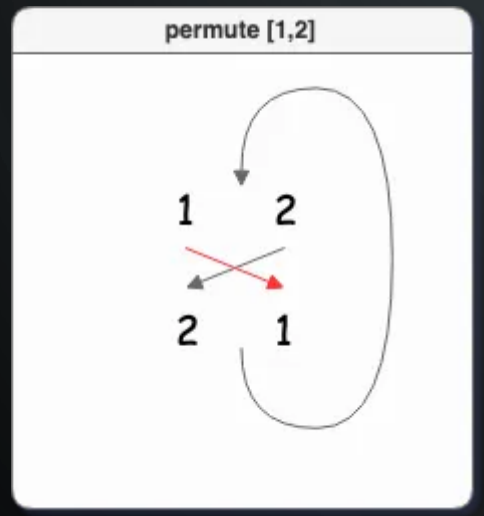
HEAP'S ALGORITHM

Heap's Algorithm systematically picks the next unique element in the descending order! e.g., permute([1,2,3]) will pick 3 > 2 > 1.



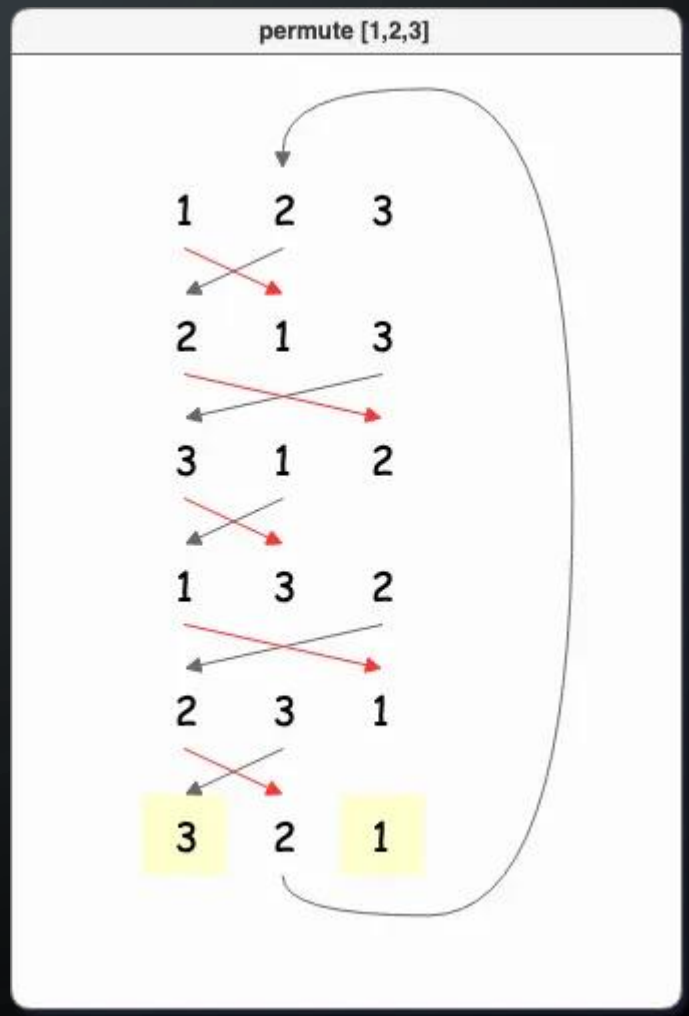
HEAP'S ALGORITHM

For permute([1,2]):



Note that the 3 and 1 are swapped at the end of iteration when the search range is odd.

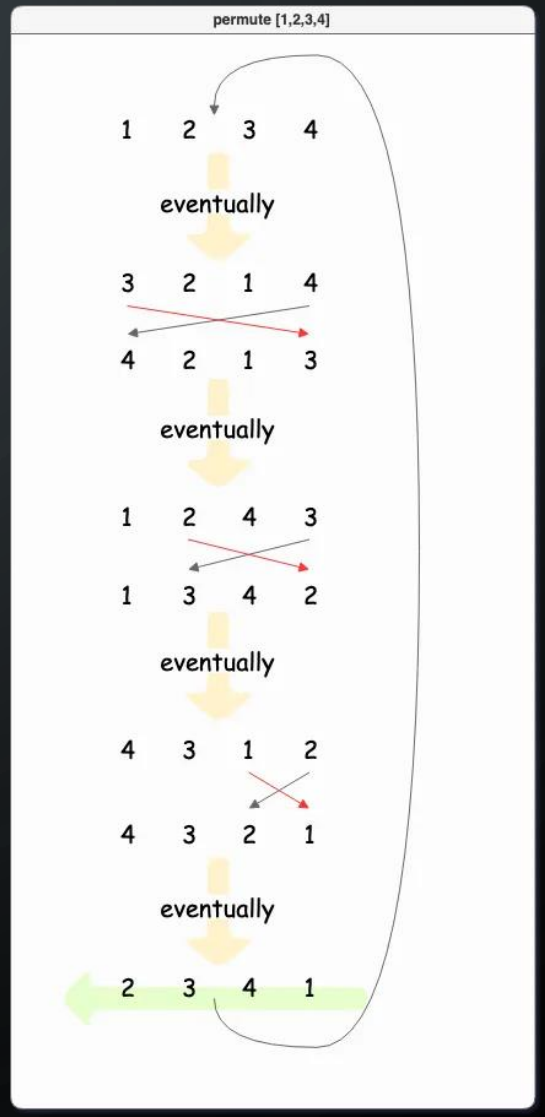
For permute([1,2,3]):



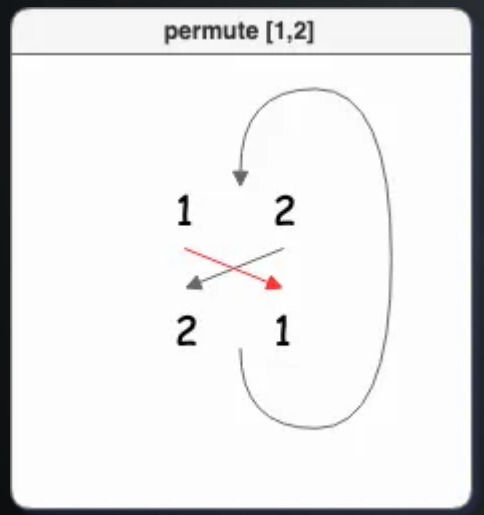
HEAP'S ALGORITHM

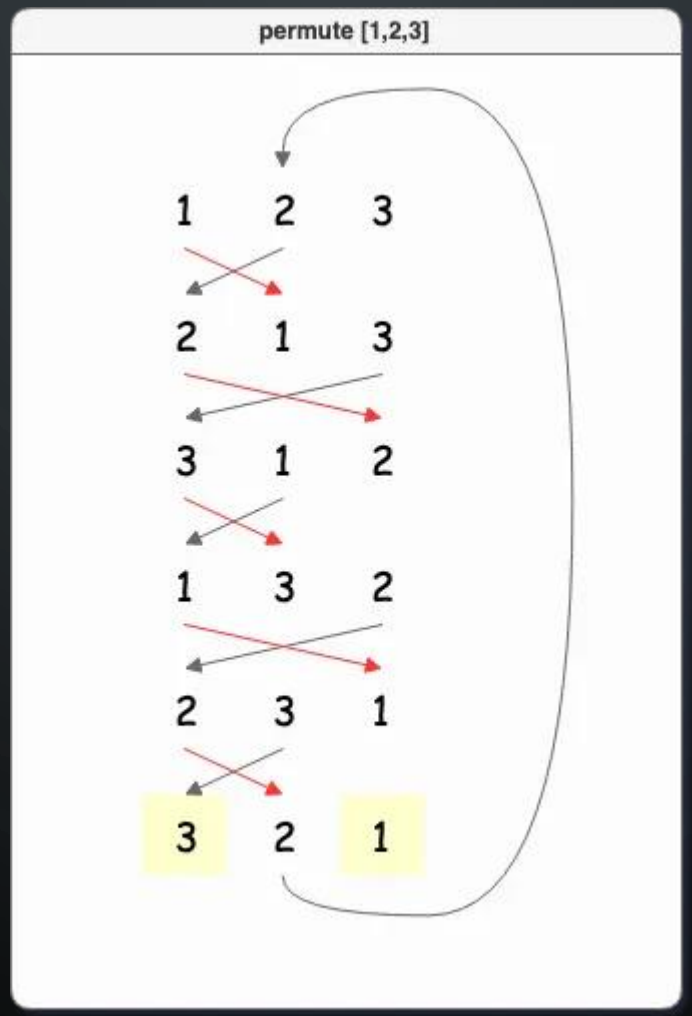
For permute([1,2,3,4]):

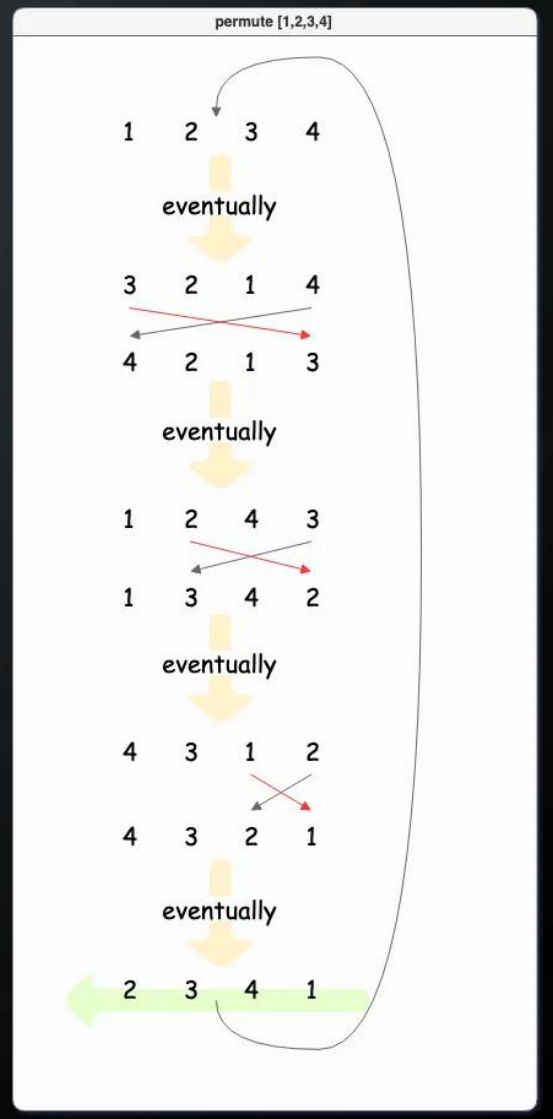
Note that the elements shift one position to the left at the end of iteration when the search range is even!



HEAP'S ALGORITHM







HEAP'S ALGORITHM

From there, we find the repeating pattern as:

1. The 1st element and last element are swapped at the end of iteration when the search range is odd. (The array is flipped in horizontal)
2. The elements shift one position to the left at the end of iteration when the search range is even.

Now we know the key of the conditional swap to the search range. Actually, the conditional swap is the systematic way to pick the next unique element to isolate.

THE PROBLEM SOLVING

Actually, the Heap's algorithm in pure implementation do not allow us to solve the maximum prize problem. We need to find not all permutations, but the case where the prize will be maximum. So, we need to check all permutation with swap count 1, then with swap count 2 and so on to swap count **K**. But, does it really needed to make **K** swapping?

Let's consider an example with 5 digit number.

THE PROBLEM SOLVING

	0				
0	9	8	7	2	1
1	9	8	1	2	7
2	9	2	1	8	7
3	2	1	7	8	9
4	2	1	8	7	9

1				
9	8	7	2	1
9	8	1	2	7
9	1	7	8	2
9	1	8	7	2

2				
9	8	7	2	1
9	8	7	1	2
9	8	1	7	2

3				
9	8	7	2	1
9	8	7	1	2

As you can see, the swap count for reaching the maximum number is **D-1**, where **D** is the digit count in number **N**. That means we do not need to do more than **D-1** swaps in case **K** > **D-1**. The possible count of swaps on the number is **D***(**D-1**)/2.

THE PROBLEM SOLVING

The algorithm to solve this problem.

1. Find the possible maximum of the prize. It needs D*(D-1)/2 swaps.
2. Set the maximum swap S count to D-1 if K > D-1 or to K otherwise.
3. Create permutation recursive function with depth of S or until maximum prize has been reached. Keep current maximum prize and current recursive depth (swap count).
4. If K < D-1 then the answer is the current maximum prize. Otherwise, we need to fix the current maximum prize according to the K value. Think wisely about how to do it.

THANK YOU!