

Graph of Thoughts:

Solving Elaborate Problems with Large Language Models

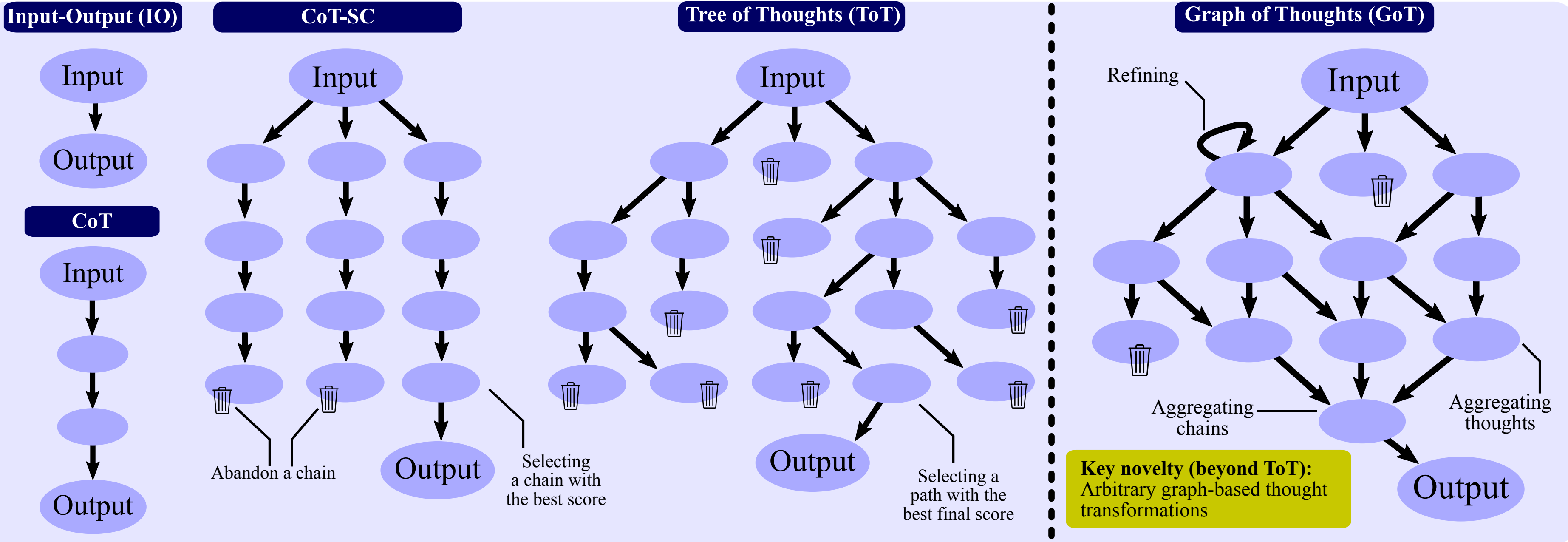
Maciej Besta, Nils **Blach**, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, Torsten Hoefler

Department of Computer Science, ETH Zurich

Motivation

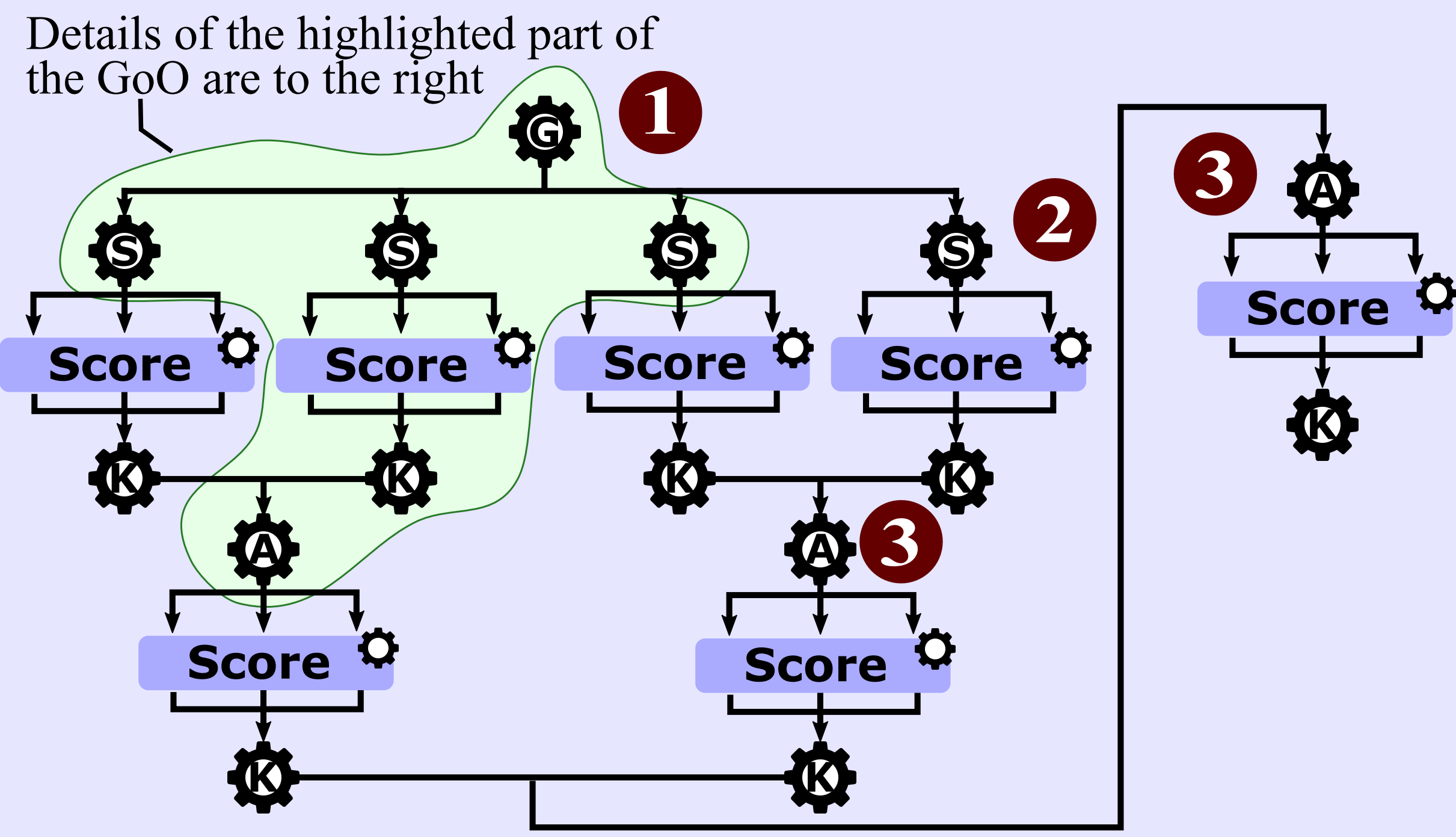
Large Language Models (LLMs) are taking over the world of AI, offering versatile solutions across a spectrum of complex tasks. Despite advancements, current prompting paradigms like Chain-of-Thought (CoT) and Tree of Thoughts (ToT) are restricted by their linear and tree-like structures, limiting the potential for modeling the multifaceted nature of reasoning. Graph of Thoughts (GoT) introduces a novel framework that models LLM reasoning as an arbitrary graph, enabling the integration of diverse thought patterns and transformations beyond the constraints of existing methods. This approach enables the combination of arbitrary LLM thoughts into synergistic outcomes, distilling the essence of whole networks of thoughts, or enhancing thoughts using feedback loops, thereby bringing LLM reasoning closer to human thinking and significantly improving LLM problem-solving capabilities.

Graph of Thoughts in the Prompting Landscape

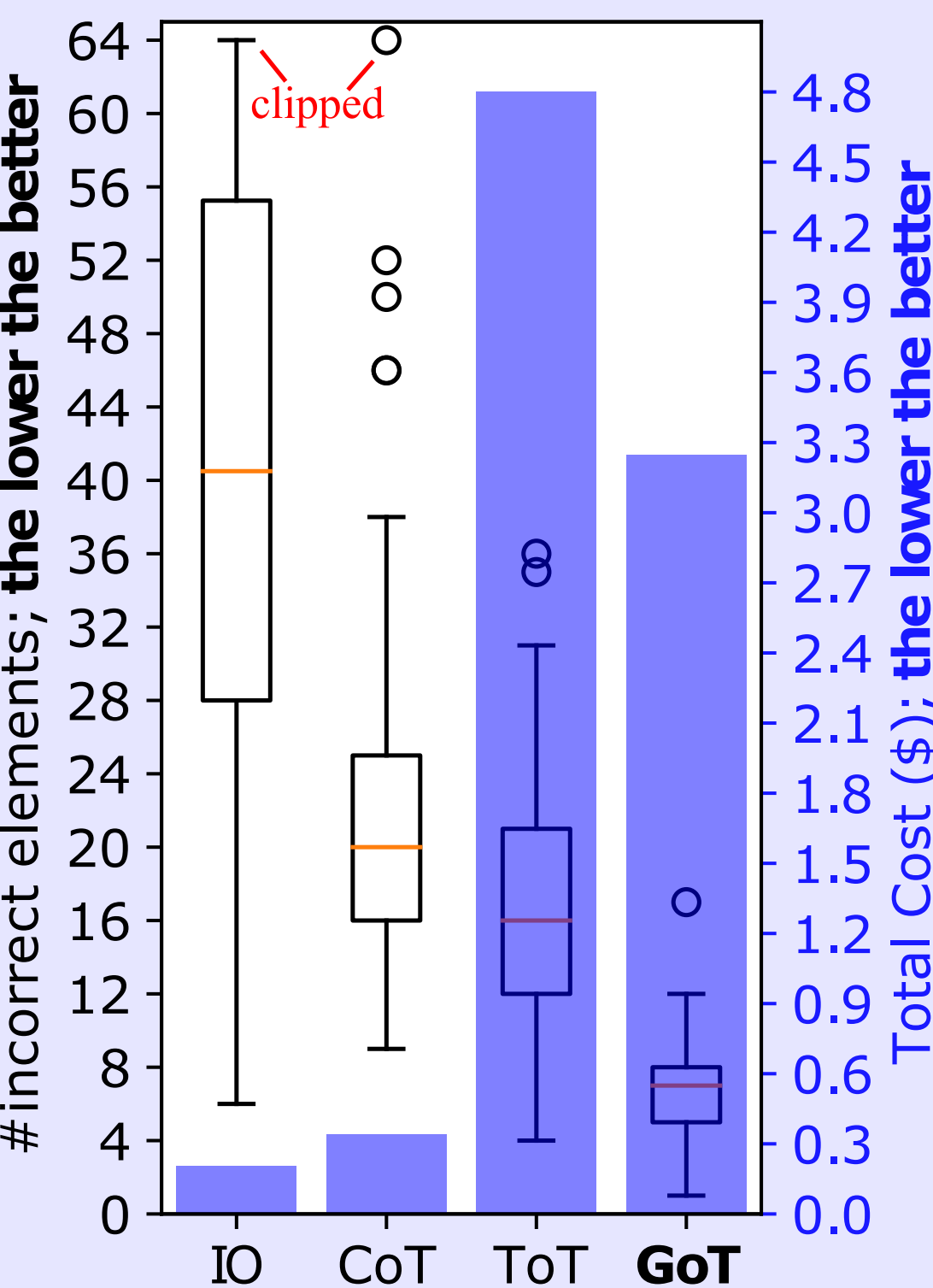


Evaluation

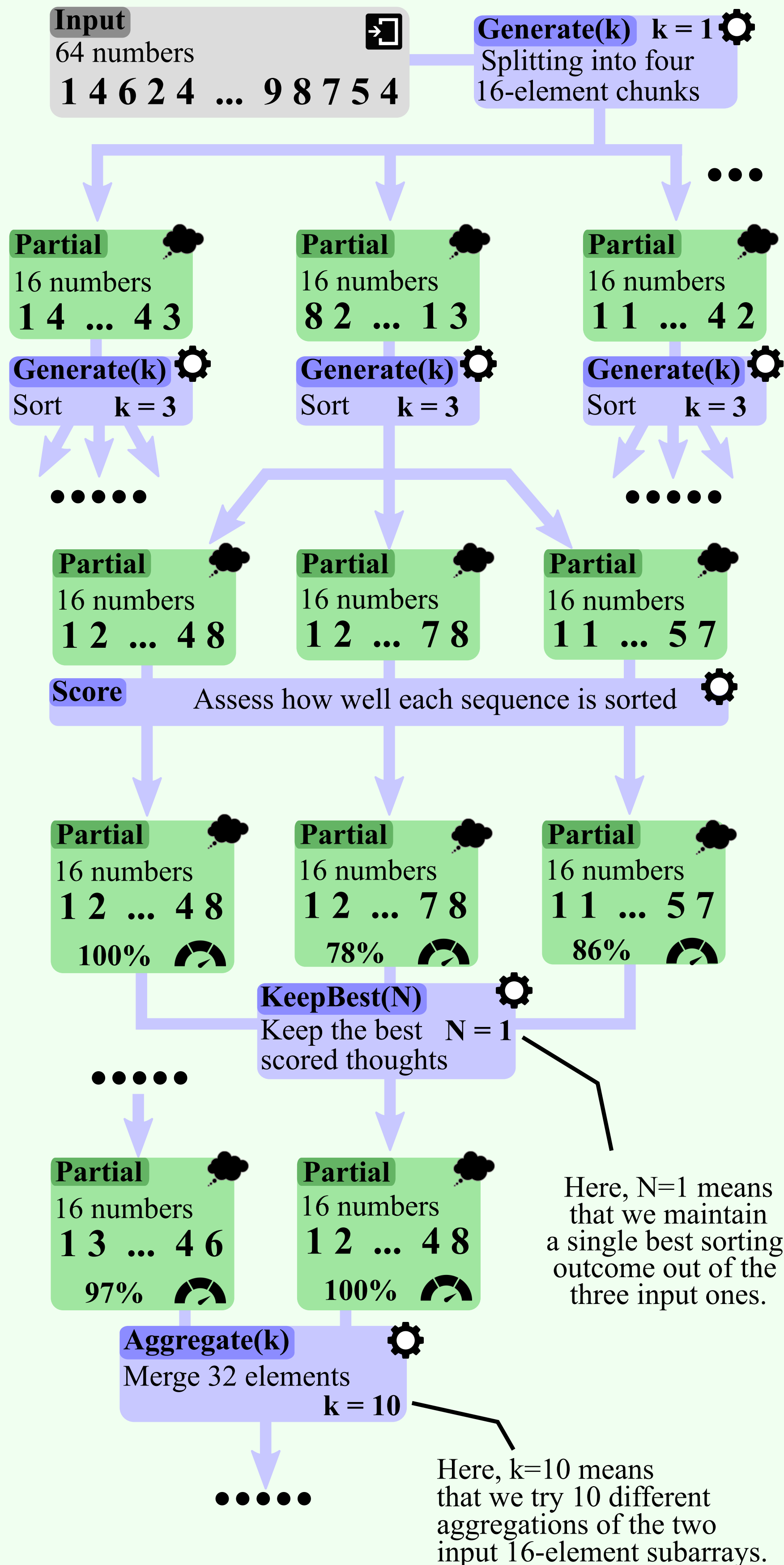
Graph of Operations (GoO) for Sorting 64 Numbers



Results for 64 Numbers



Details of the Highlighted Part of the GoO



Example Prompts for Sorting

A prompt used by `Generate(t, k=4)`

<Instruction> Split the following list of 64 numbers into 4 lists of 16 numbers each, the first list should contain the first 16 numbers, the second list the second 16 numbers, the third list the third 16 numbers and the fourth list the fourth 16 numbers. Only output the final 4 lists in the following format without any additional text or thoughts!

<Example>
Input: [3, 1, 9, 3, 7, 5, 5, 4, 8, 1, 5, 3, 3, 2, 3, 0, 9, 7, 2, 2, 4, 4, 8, 5, 0, 8, 7, 3, 3, 8, 7, 0, 9, 5, 1, 6, 7, 6, 8, 9, 0, 3, 0, 6, 3, 4, 8, 0, 6, 9, 8, 4, 1, 2, 9, 0, 4, 8, 8, 9, 9, 8, 5, 9]
Output: {{
"List 1": [3, 4, 3, 5, 7, 8, 1, ...],
"List 2": [2, 9, 2, 4, 7, 1, 5, ...],
"List 3": [6, 9, 8, 1, 9, 2, 4, ...],
"List 4": [9, 0, 7, 6, 5, 6, 6, ...]
}}
</Instruction>

A prompt used by `Generate(t, k=1)+Repeat(k=4)`

<Instruction> Sort the following list of numbers in ascending order. Output only the sorted list of numbers, no additional text.

<Example>
Input: [3, 7, 0, 2, 8, 1, 2, 2, 2, 4, 7, 8, 5, 5, 3, 9, 4, 3, 5, 6, 6, 4, 4, 5, 2, 0, 9, 3, 3, 9, 2, 1]
Output: [0, 0, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9]
</Example>

A prompt used by `Aggregate(t1, t2)+Repeat(k=3)+KeepBest(N=1)`

<Instruction> Merge the following 2 sorted lists of length {length1} each, into one sorted list of length {length2} using a merge sort style approach. Only output the final merged list without any additional text or thoughts!

<Approach>
To merge the two lists in a merge-sort style approach, follow these steps:
1. Compare the first element of both lists.
2. Append the smaller element to the merged list and move to the next element in the list from which the smaller element came.
3. Repeat steps 1 and 2 until one of the lists is empty.
4. Append the remaining elements of the non-empty list to the merged list.
</Approach>

Merge the following two lists into one sorted list:
1: {input1}
2: {input2}

Merged list: