Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Finding combinations of four-operations with Type-2 tree structure

Tip-2 arama yöntemiyle dört işlem kombinasyonlarının bulunması

Eda ÖZKUL[1], *Buğra Kaan TİRYAKİ*[2], *Özge TEZEL*[3*], *Elçin AĞAYEV*[4], *Orhan KESEMEN*[5]

[1,2,3,4,5]Department of Statistics and Computer Sciences, Faculty of Science, Karadeniz Technical University, Trabzon, Turkey.
eda.ozkul.gs@gmail.com, bugrakaantiryaki@gmail.com, ozge_tzl@hotmail.com, agayevelcinn@gmail.com, okesemen@gmail.com

Received/Geliş Tarihi: 08.01.2020 Revision/Düzeltme Tarihi: 06.06.2020 doi: 10.5505/pajes.2020.77906
Accepted/Kabul Tarihi: 11.06.2020 Research Article/Araştırma Makalesi

Abstract

Combination problems are one of the most important issues of probability theory. The four-operations combination problem underlies the basis of some competition programs broadcasted in many national channels. In these competition programs, the competitors are expected to reach the target number by using six numbers and four basic arithmetic operators. The numbers are used at most once, the operators can be used any desired number to reach the target number. In this problem, all four-operations combinations include the operation blocks consisting of two numbers and an operator. Therefore, the four-operations combination problem is solved by developing a "Type-2 Tree Structure" which is a new approach to accurately model the operation blocks. The performance of the proposed method for the four-operations combination problem is examined by a simulation study. Also, the statistics from experimental results are given in this study.

Keywords: Combination, Four-Operations, Type-2 tree, Operation block.

Öz

Kombinasyon problemleri olasılık teorisinin en önemli konularından biridir. Dört işlem kombinasyon problemi, birçok ulusal kanalda yayınlanan bazı yarışma programlarının temelini oluşturmaktadır. Bu yarışma programlarında, yarışmacıların 6 adet sayı ve dört işlem operatörlerini kullanarak, hedef sayıya ulaşması beklenmektedir. Hedeflenen sayıya ulaşmak için sayılar en fazla bir kez kullanılırken, dört işlem operatörleri istenilen sayıda kullanılabilir. Bu problemde, bulunacak olası tüm dört işlem kombinasyonları iki sayı ve bir operatörden oluşan işlem öbeklerini içermektedir. Dolayısıyla işlem öbeklerini tam anlamıyla modelleyebilmek için yeni bir yaklaşım olan "Tip-2 Ağaç" yapısı geliştirilerek dört işlem kombinasyon problemi çözülmüştür. Dört işlem kombinasyon problemi için önerilen yöntemin performansı bir simülasyon çalışması ile incelenmiştir. Ayrıca deneysel sonuçlardan elde edilen istatistikler de bu çalışmada verilmiştir.

Anahtar kelimeler: Kombinasyon, Dört işlem, Tip-2 ağaç, İşlem öbeği.

1 Introduction

Four-operations combination problem is first started broadcasting as a game in 1972 on French television as "des chiffres et des lettres". Later, it appeared on British television with the name of "countdown" in 1982. In addition, it is called as "bir kelime bir işlem" in Turkey. The main goal of the game is to achieve the target number by performing four-operations on the given number [1]. It is played with six numbers. Five of them are randomly selected from 1 to 9, and the other is randomly selected from the cluster {25, 50, 75, 100}. The competitions try to reach a randomly selected target number (from 101 to 999) by applying the basic arithmetic operations $\{+,-,\times,/\}$ to six numbers. According to the rules of the game, the selected six numbers can only be used at most once. However, all new results produced between two numbers can be used in the other operations. For example, let the selected numbers are {3, 6, 1, 4, 5, 25} and the target number is 403. In this case, the operation steps are given in Table 1.

Table 1. Example of solution of four-operations problem.

	Operations	Number Pool
Initial	...	{3,6,1,4,5,25}
Step 1	$6\times5=30$	{1,3,4,25,30}
Step 2	$30+1=31$	{3,4,25,31}
Step 3	$3\times4=12$	{25,31,12}
Step 4	$25-12=13$	{31,13}
Step 5	$13\times31=403$	{403}

In step 1, 30 obtained by multiplying 6 and 5, is added to the number pool and {6, 5} are excluded from the number pool. In each subsequent step, the used numbers are removed from the number pool and their results are added to the number pool. There may be multiple solutions to reach the target number. The solution that reaches the target number with minimum steps can be preferred. Despite the fact that games based on four-operations are very common, the limited number of scientific studies have been conducted in this regard. Defays [2],[3] used artificial intelligence search methods to solve these games. Hutton [4] developed a simple but ineffective functional program for solving four-operations problems. Despite different approaches and software for solving the game are exhibited on many websites, they cannot always give the desired results [5]. Alliot [5] has developed a new approach, which focuses solely on solving the problem. But it was not interested in other combinations.

In this paper, the four-operations problem is considered as a combination problem and it is aimed to solve the problem by developing the Type-2 tree structure.

2 Obtaining permutation and combination lists

The finite sample spaces represent a set that consists of the finite number elements. Any sample that is chosen randomly from this set creates a subset concurrently. Regular subsets are formed by sampling according to a certain rule or ordering the sample space according to a rule [6],[7]. Regular subset

[*]Corresponding author/Yazışılan Yazar

definitions change based on the problems. The most common of them are permutations and combinations.

2.1 Permutation

The permutation is called an arrangement or ordering of elements in a set [8],[9]. Each element can only be used once in the permutation. This can be given as an example of sampling without replacement. It is also important that which element come in which order [10]. Figure 1 shows the demonstration of the tree structure of {USED}-{REMAINED} approach for better understanding of the permutation process. It shows that there are 6 possible permutations of three elements such as {A, B, C}.

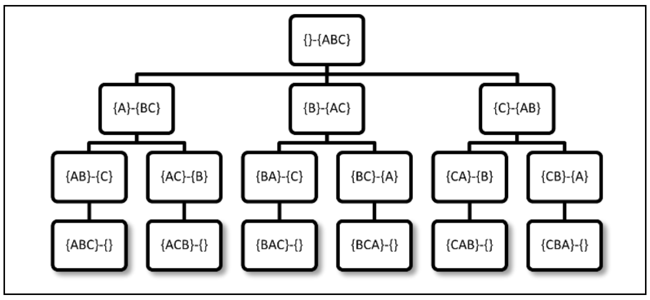


Figure 1. Tree representation of permutations with {USED}-{REMAINED} approach.

If n elements were given instead of 3 elements, then the number of all possible solutions can be found using the following equation.

$$n \cdot (n-1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1 = n! \tag{1}$$

If all elements in the given set are used, the number of permutations is calculated as follows.

$$P(n) = n! \tag{2}$$

The number of all possible permutations obtained by using k elements instead of using all elements is calculated as follows.

$$P(n,k) = \frac{n!}{(n-k)!} \tag{3}$$

Generally, the permutation problems are interested in the number of all possible samples [11]. This study deals with a set of all possible samples. Algorithm 1 can be used to obtain all possible ordered samples from a set.

Algorithm 1. Finding all permutations with {USED}-{REMAINED} approach.

```
function [OUTPUT] = Permutation(S, k)
// Input_
// S : The number set
// k : Number of selected elements

// Output_
// OUTPUT : The all results

// Set USED, OUTPUT and LIST to empty set

REMAINED.Push(S)
LIST.Push(pair(USED,REMAINED)
while length(LIST) > 0
  pair(USED,REMAINED) <- LIST.pop()
  N = length (REMAINED)
  if N == 0 then
    OUTPUT.append(USED)
    Continue
  end if
  for i in (N-1 to -1) do
    USED.Push(REMAINED.pop())
    LIST.Push(USED)
  end for
end while
return OUTPUT
```

In Algorithm 1, an integer array (S) consisting of n elements is added to the list. When the loop starts, each element in the {REMAINED} is added to {USED}. The added each element is removed from the {REMAINED} concurrently. Each element in the {REMAINED} is added one by one to the {USED} in the next loop. This process continues until all elements have been added to the {USED} (Figure 1).

2.2 Combination

The combination is the selection of k elements from a set with n elements, regardless of the order of elements [8],[12]. Since the order is insignificant, the selection cannot be made as in the permutation. This is because different orders of the same elements can represent a subset. The index numbers in the given array must be selected sequentially when obtaining the subset from the arranged in order elements in the set. In this case, the first element is the first $(n-k+1)$ elements. If the elements next $(n-k+1)^{\text{th}}$ element are selected, there are not enough elements to perform the other subsets according to the rule. The second element must be one of the elements which come after the first element. k elements can be selected with the same approach. The combinations of three elements without repetition taken from the set {A, B, C, D, E} consisting of five elements are shown in the tree structure in Figure 2.

Algorithm 2. Finding all combinations with {USED}-{REMAINED} approach

```
function [OUTPUT] = Combination(S, k)
// Input_
// S : The number set
// k : Number of selected elements

// Output_
// OUTPUT : The all results

// Set USED, OUTPUT and LIST to empty set

N=length(REMAINED)
REMAINED.Push(S)
LIST.Push(USED)
while len(LIST) > 0
  USED <- LIST.pop()
  if length(USED) == k then
    OUTPUT.Append(REMAINED[index] for i in USED
    continue
  end if
  if length(USED) > 0 then
    for i in (N-1 to USED[-1]) do
      USED.Push()
      LIST.Push(USED)
    end for
  else
    for i in (N-1 to -1) do
      USED.Push()
      LIST.Push(USED)
    end for
  end if
end while
return OUTPUT
```

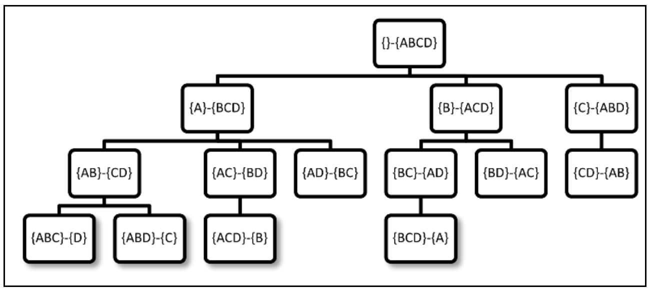


Figure 2. The tree structure of the combination with {USED}-{REMAINED} approach.

As seen from the tree structure, in the all obtained combinations, there is no contradiction to the order in the main set. Based on this representation, there are 3 elements starting with A in the combinations. There is 1 element continue with B. Combination is calculated using the Equation 4.

$$C(n,k)=\frac{n!}{k!\,(n-k)!} \tag{4}$$

For example, three students are required from each class for a competition among the classes. In how many different ways can three students from a class of twelve students be selected? Note that the order of students is insignificant. The approach to be used in the combination is to calculate the permutations of the set by giving a rank index to the elements in the set. In the permutation, if the element with the bigger index is not be placed before the element with the small index, combinations can be selected from within the permutations. The point the note here is that the number of subset elements must not exceed the required k elements. Algorithm 2 can be used to obtain all possible samples from a given population.

In the combinations, each element in each subset is used once. An element in a set can be selected more than one is called combinations with repetition.

3 Four-operations combination problem

The four-operations combination problem aims to find a set of all possible results by constructing an arithmetic expression using n selected numbers. The nature of the problem, arithmetic operators can be used as often as requested, but the selected numbers can be used at most once.

In the literature, the brute force method is a very simple and basic problem-solving technique. It addresses all possible situations one by one in the solution space where the problem is defined systematically. It checks whether these situations provide a solution to the problem. In addition, this technique is used for various problems and can be considered as a general approach to problem solving. On the other hand, the solution of the problem depends on the number of possible solutions. Therefore, as the number of possible solutions increases, the calculation time also increases [13],[14].

In this study, the Type-2 tree that calculates all possible solutions of the problem is proposed similar to the brute force method. Furthermore, the entire solution space must be searched for the complete solution of this problem.

Although mathematical expressions consist of numerical values, textual (string) notations must be used for their storage and representation. A mathematical expression in the tree structure is converted to a node by using the string storage technique. Since this node corresponds to a node in the general tree structure, it provides convenience the use of the tree structure into the tree [15],[16].

As an example, operations given in Table 1 can be represented in the form of a single-line as follows.

$$\big(1+(6\times 5)\big)\times(25-(3\times 4)) \tag{5}$$

This mathematical expression can be stored in memory in a string structure. In this case, the extraction of the string expressions and the determination of their values will also require additional operations which are given below.

- Converting the string expression to the tree structure (STRING2TREE),
- Converting the tree structure to the value (TREE2VALUE),
- Converting the tree structure to the string expression (TREE2STRING).

3.1 Converting the string expression to the tree structure (STRING2TREE)

Initially, each operator must be given an order of priority to convert a mathematical expression given as text to the tree structure. In this study, the identified orders of priority are given in Table 2. In mathematics, the '+' and '−' operators have the same priority. But in this study, different priorities are given to avoid complexity. Similarly, the '×' and '/' operators have been given different priorities. However, if two or more operators having the same priority are in the same expression, the operations are performed from left to right, respectively. Furthermore, in mathematical expression, each operator is added to the priority value 10 after the left brace, but after the right brace, each operator is subtracted by 10 from the priority value.

Table 2. Priority assignments.

Operation	Operator	Priority
Addition	+	1
Subtraction	−	2
Multiplication	×	3
Division	/	4
Left Brace	(	+10
Right Brace	)	-10

For example, let a mathematical expression is "$5\times(3+2)-8/4$". The priority array of this expression is given in Table 3. Here, the priority values of the characters except the operators are given as 0. Although the priority value of the '+' operator is 1, after the left brace it has been increased by 10 to 11.

Table 3. Priority array of the example expression.

Characters	5	×	(	3	+	2	)	−	8	/	4
Priorities	0	3	0	0	11	0	0	2	0	4	0

The operator with the lowest non-zero priority in the text string is the root node of the tree. The characters which are the left or right side of this operator create the sub-nodes of it (Figure A1). In the following steps, each child node is created own child nodes. This process continues until there is no operator remained in the sub-node. As a result, only numbers may remain in all end-nodes. The nodes containing the operator in the text string are extracted and if there are nodes that do not contain an operator, the brackets are deleted, and only numbers remain in the end-nodes.

3.2 Converting the tree structure to the value (TREE2VALUE)

A mathematical expression which is given as a text string cannot be directly computed on a computer. The string expressions must be converted to the tree structure. The created tree can be kept in data structures such as queue. In the tree structure, the intermediate nodes contain operators and the end-nodes contain numbers. The result is achieved starting from the lowest depth to upwards in the mathematical expression converted to the tree structure (Figure A2). Sub-nodes containing numbers are operationalized with the operator in their connected parent node. The obtained value is written in the parent node and the used sub-nodes are deleted. This process, which is done with the graph, is performed by keeping the tree structure on the computer. The last two numbers in the queue are taken and processed with the first operator that is met from head to tail. Then, the obtained result is written in the position where the operator is located and, it is continued by deleting the last two numbers. The last remaining value is equivalent to the result (Figure A2).

Every step of the process is stored in a list as in Table 4 when a tree structure is transformed into a value as in Figure A2.

Additional information is always needed to manage operations on the tree structure. The tree is added to a list in the queue to eliminate this. After each node added to the list is processed and new child nodes are added to the list without being deleted from the list. The extracted operator is also added to the parent node.

Table 4. Solution of the example four-operations problem.

	Operations	Number Pool
Initial	…	{5,8,4,3,2}
Step 1	$3 + 2 = 5$	{5,5,8,4}
Step 2	$8 /4 = 2$	{2,5,5}
Step 3	$5 \times 5 = 25$	{25,2}
Step 4	$25 - 2 = 23$	{23}

3.3 Converting the tree structure to the string expression (TREE2STRING)

Combination problems can be solved with graphs via tree structure. The tree structure is usually kept on dynamic lists. However, saving and displaying them is a troublesome process. For this reason, the tree structure on dynamic lists is used by converting to the string expression (Figure A3). Starting from the sub-node of the lowest deep, operation block which consists of an operator connected to these nodes, is recorded as a string to the node in which location of the operator, and the sub-nodes are removed. In the new case, the operator node where the location of the operation block, becomes an external node. This process is repeated until only one node is left. In this case, it is the string expression of whole operations (Figure A3).

4 Solving four-operations combination problem

The four-operations combination problem is quite similar to the permutation-combination problems. However, while subsets are constructed according to a certain rule in the case of the permutation-combination problems, they are constructed in triple intersection with two sets which are numbers {A, B, C} and operators $\{+, -, \times, /\}$ in four-operations combination problems (Figure 3).

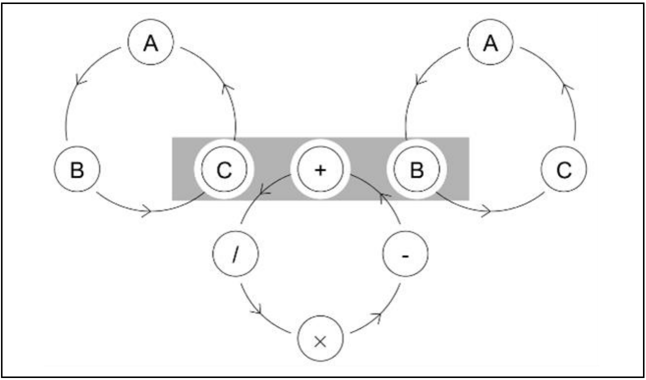


Figure 3. The operation block constructed with triple intersection of two sets.

The operation blocks can be consisted of using the binary permutations of the number set and the single permutations of the operator set. But, when the binary permutations of the number set are combined with the single permutation of the operator set, the combined operation blocks results can be same as $(B + C)$ and $(C + B)$. It causes to consist of the repetitive subset. Therefore, permutation pairs for operators "$-$" and "$/$", and combination pairs for operators "$+$" and "$\times$" are used to solve it (Table 5). Also, this preference is made by courtesy of the commutative property of the addition and multiplication.

Table 5. All operation blocks that can be consisted of a three-element set.

$+$	A	B	C	$\times$	A	B	C
A	…	$A + B$	$A + C$	A	…	$A \times B$	$A \times C$
B	…	…	$B + C$	B	…	…	$B \times C$
C	…	…	…	C	…	…	…
		(a)				(b)	

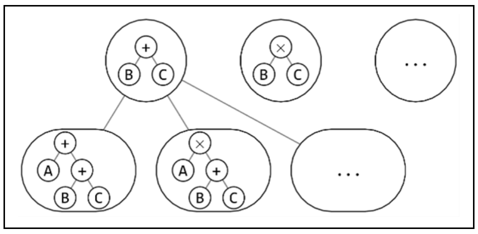
$-$	A	B	C	$/$	A	B	C
A	…	$A - B$	$A - C$	A	…	A/B	A/C
B	$B - A$	…	$B - C$	B	B/A	…	B/C
C	$C - A$	$C - B$	…	C	C/A	C/B	…
		(c)				(d)	

All the binary operations consisting of a set of elements {A, B, C} are given in Table 5. The binary operation blocks which are combinations with repetition represented by "..." can be added to the list according to the problem. In this study, the combinations with repetition are ruled out.

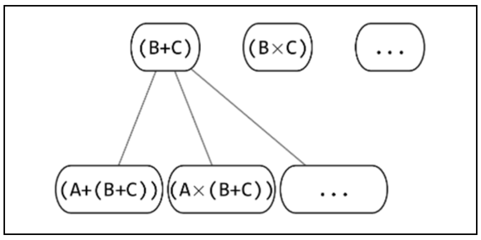
Each operation block can be a part of another operation block. Each single or multi operation block creates a tree. Different sequences of operation blocks cause different combinations. If each of them is represented by a tree, it is necessary that tree must create its own tree for the calculation of all combinations. In other words, tree of each operation block represents a node of the combination tree (Figure 4a). The combinations of the multi operation blocks can be kept by the Type-2 tree while a multi operation block is solved with the Type-1 tree. That is to say, the combined tree structure formed by keeping an operation block tree at each node of the combination tree can be defined as a Type-2 tree.

4.1 Solution of the problem with the type-2 tree structure

The subtree which aims to solve the four-operations combination problems in a simpler way using the Type-2 tree, is listed as a string, and the expansion of the tree (branching) can be resumed (Figure 4b). The numerical value of the string expression obtained from the operation result can be found.



(a)



(b)

Figure 4. The solution tree of the four-operations combination problem. (a): Schematic representation of the Type-2 tree. (b): String expression of the Type-2 tree.

If the operation result is used instead of the string expression, each generated operation block result is added to the {REMAINED}, and used numbers are removed from the {REMAINED}. In this way, a number set is created in each node, and the value of the current operation block is added to the list towards to the sub-depth. In addition, the numbers used in the operation blocks are removed from the {REMAINED} (Algorithm 3).

Algorithm 3. Type-2 tree structure to solve four-operations combination problem

```
function [OUTPUT] = FourOperator(REMAINED):
// Input_
// REMAINED : The number set

// Output_
// OUTPUT : The all results

// Set OUTPUT and LIST to empty set

LIST.Push(REMAINED)
while length(LIST) > 0
   b = LIST.pop()
   if length(b) is equal to 1 then
      continue while loop
   end if
   for i in (1 to length(b)) do
      for j in (1 to length(b)) do
         if i ≠ j then
            REMAINED <- set to zeros(n-1)
            index = 1
            for k in (1 to length(b)) do
               if(k ≠ i and k ≠ j) then
                  REMAINED[index] = b[k]
                  index += 1
               end if
            end for k
            if j > i then
               REMAINED[index] = b[i] + b[j]
               LIST.Push(REMAINED)
               OUTPUT.append(REMAINED[index])
               REMAINED[index] = b[i] * b[j]
               LIST.Push(REMAINED)
               OUTPUT.append(REMAINED[index])
            end if
            REMAINED[index] = b[i] - b[j]
            LIST.Push(REMAINED)
            OUTPUT.append(REMAINED[index])
            REMAINED[index] = b[i] / b[j]
            LIST.Push(REMAINED)
            OUTPUT.append(REMAINED[index])
         end if
      end for j
   end for i
end while
return OUTPUT
```

Consider a set of elements {A, B, C} for a simple implementation of the problem. In the first step, the binary permutations of the elements are shown as in Table 6.

Table 6. Binary operation blocks of the set with two elements.

Binary Permutation	$\{(A+C), B\}$	$\{B, (A+C)\}$
{REMAINED}	{}	{}
Addition	$\{(A+C)+B\}$	...
Subtraction	$\{(A+C)-B\}$	$\{B-(A+C)\}$
Multiplication	$\{(A+C)\times B\}$	...
Division	$\{(A+C)/B\}$	$\{B/(A+C)\}$

There are operation blocks having the same results in a row that contain addition and multiplication operators in Table 6. Thus, only one of them is considered. However, all results of subtraction and division in a row have been selected. All the elements in the rows that contain subtraction and division operations are taken. Thus, 18 new sets are obtained as a result of the binary four-operations combination of a set with three elements. Table 7 shows the four-operations combinations of the set $\{(A+C), B\}$ which is one of the new sets with two elements. Here, the expression $(A+C)$ is defined as a number.

The number of the binary operation blocks of three elements is 18. The number of triple operation blocks gives 6 results for each. As a result, the number of triple operations of three elements is $6 \times 18 = 108$. In addition, the number of all four-operations combination of a set with three elements is calculated as $18 + 108 = 126$, when the binary operations results are included.

4.2 Determination of the number of the four-operations combinations

The total number of operation blocks obtained according to the numbers and the operators. The number of new operation blocks obtained from them is calculated using permutation. In this case, the number of the binary operation blocks obtained from a set with n elements is calculated as in Equation (6).

$$D(n,2) = 3.P(n,2) \tag{6}$$

In Equation (6), while all permutations of the subtraction and division operators are considered, the half of the permutations of the addition and multiplication operators are considered.

Thus, the number of permutations is reduced 3 times. In the same way, the number of triple operation blocks of a set with n elements is determined as in Equation (7).

$$D(n,3) = 3^2.P(n,2).P(n-1,2) \tag{7}$$

Table 7. Binary operation blocks of a set with three-element.

Binary permutations	{A, B}	{A, C}	{B, A}	{B, C}	{C, A}	{C, B}
{REMAINED}	{C}	{B}	{C}	{A}	{B}	{A}
Addition	$\{(A+B), C\}$	$\{(A+C), B\}$	{}	$\{(B+C), A\}$	{}	{}
Subtraction	$\{(A-B), C\}$	$\{(A-C), B\}$	$\{(B-A), C\}$	$\{(B-C), A\}$	$\{(C-A), B\}$	$\{(C-B), A\}$
Multiplication	$\{(A\times B), C\}$	$\{(A\times C), B\}$	{}	$\{(B\times C), A\}$	{}	{}
Division	$\{(A/B), C\}$	$\{(A/C), B\}$	$\{(B/A), C\}$	$\{(B/C), A\}$	$\{(C/A), B\}$	$\{(C/B), A\}$

Hence, Equation (8) calculates the number of the n-operation blocks of a set with n elements.

$$D(n,n) = 3^{n-1}.P(n,2)\dots.P(4,2).P(3,2).P(2,2) \tag{8}$$

If it is generalized, Equation (9) is obtained and, thus, the number of the k-operation blocks is calculated.

$$D(n,k) = 3^{k-1}\prod_{t=n-k+2}^{n} P(t,2) \tag{9}$$

The number of all operation blocks of a set with n elements are figured out using Equation (10).

$$D(n) = \sum_{k=2}^{n}\left(3^{k-1}\prod_{t=n-k+2}^{n} P(t,2)\right) \tag{10}$$

Table 8 shows the results for $n = 2,3,4,5,6$.

Table 8. The number of the four-operations combinations.

$D(2,2)$	6
$D(2) = \sum_{k=2}^{2} D(2,k)$	6
$D(3,2)$	18
$D(3,3)$	108
$D(3) = \sum_{k=2}^{3} D(3,k)$	126
$D(4,2)$	36
$D(4,3)$	648
$D(4,4)$	3888
$D(4) = \sum_{k=2}^{4} D(4,k)$	4572
$D(5,2)$	60
$D(5,3)$	2160
$D(5,4)$	38880
$D(5,5)$	233280
$D(5) = \sum_{k=2}^{5} D(5,k)$	274380
$D(6,2)$	90
$D(6,3)$	5400
$D(6,4)$	194400
$D(6,5)$	3499200
$D(6,6)$	20995200
$D(6) = \sum_{k=2}^{6} D(6,k)$	24694290

5 Experimental results

This section investigates the performance of the proposed method for the four-operations combination problem. For the simulation study, a 64-bit computer with an Intel® Core ™ i7-3630QM CPU @ 2.40GHz processor and 8 Gb Ram was used, and *C#* was used for the applications. Random integers between 1 and 9 are selected as the simulation parameters and the results are recorded. This process is run 100 times with the same n value and the average computation time according to the count of the selected numbers is shown in Table 9.

Table 9. The average computation time of the proposed method in 100 trials.

The count of the used numbers	The number of the generated results	The average computation time (sec.)
2	6	0.00000209
3	126	0.00000704
4	4572	0.00023027
5	274380	0.01541066
6	24694290	1.94922640

The results obtained by the random integers between 1 and 9 were generated as many as the number used in Table 9, and the computation time was computed according to the results. According to Table 9, the number of results and the computation time increase, when the number of elements of the number set increases. Also, the complexity of the algorithm is calculated as $O(n!(n-1)!)$. On the other hand, the statistics such as the count of used number (n), the number of generated results for each experiment (N), the minimum result (Y_{min}), the maximum result (Y_{max}), the number of the infinite results (N_i), the number of the undefined results (N_n) and percentage of the number falling between 100-1000 (f%) obtained from experimental results are given in Table 10. According to Table 10, it is seen that as the count of used number (n) increases, the minimum result (Y_{min}) decreases and the maximum result (Y_{max}) increases. On the other hand, the expressions like a/∞, 0/0, 0×∞ which equal to infinity are expected. So, the number of the infinite results (N_i) and the number of the undefined results (N_n) increase, when the count of used number increases. The percentages of the number falling between 100-1000 are also given.

6 Conclusion

This study improved the Type-2 tree to solve the four-operations combination problems. A simulation study was performed to test the performance of the proposed method. The random integers used in the simulation study were selected in the range of [1,9], and the average computation time was obtained according to the count of the selected numbers. The computational complexity was found theoretically according to the parameter n and compared with the computation time. As the number of elements in the number set increases, the number of the generated results and the computation time increase. The four-operations combination problems are performed via integers given in a certain number. They can be expanded to both real and complex numbers. In addition, the heuristic approaches can be improved to find the shortest solutions that reach the target number.

Table 10. The statistics obtained from experimental results.

n	N	Y_{min}	Y_{max}	N_i	N_n	$f\%$
2	6	-2.99	25.16	0	0	0
3	126	-34.37	128.24	0.72	0	1.23015873
4	4572	-176.58	514.52	42.46	1.12	1.610673666
5	274380	-1155.27	3208.38	3536.82	58.4	2.438435746
6	24694290	-2.37828E+16	2.37828E+16	396003.94	9475.52	2.853373148

$\boldsymbol{n}$: the count of used number; $\boldsymbol{N}$: the number of generated results for each experiment; $\boldsymbol{Y_{min}}$: the minimum result; $\boldsymbol{Y_{max}}$: the maximum result; $\boldsymbol{N_i}$: the number of the infinite results; $\boldsymbol{N_n}$: the number of the undefined results; $\boldsymbol{f}\%$: Percentage of the number falling between 100-1000.

7 References

[1] Colton S. "Countdown numbers game: solved, analysed, extended". *Proceedings of the AISB Symposium on AI and Games*, Canterbury, London, 1 April 2014.

[2] Defays D. *L'esprit en Friche: les Foisonnements de l'intelligence Artificielle*. Liege, Belgium, Pierre Mardaga, 1988.

[3] Defays D. *Numbo: A Study in Cognition and Recognition*. Editor: Hofstadter D. Fluid Concepts and Creative Analogies, 131-154, Basic Books Inc, 1995.

[4] Hutton G. "The countdown problem". *Journal of Functional Programming,* 12(06), 609-616, 2002.

[5] Alliot JM. "The (Final) Countdown". https://arxiv.org/abs/1502.05450 (03.06.2020).

[6] Fischler MA, Bolles RC. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, 24(6), 381-395, 1981.

[7] Montgomery DC, Runger GC. *Applied Statistics and Probability for Engineers*. 5th ed. Jefferson City, USA, John Wiley & Sons, 2011.

[8] Hines WW, Montgomery DC, Goldsman DM, Borror CM. *Probability and Statistics in Engineering*. Danvers, Massachusetts, USA, John Wiley & Sons, 2008.

[9] Code D. *Probability: Mastering Permutations and Combinations*. 2nd ed. California, USA, CreateSpace Independent Publishing Platform, 2017.

[10] Crawshaw J, Chambers J. *A Concise Course in Advanced Level Statistics: With Worked Examples*. 6th ed. London, UK, Oxford University Press, 2015.

[11] Tier R. *Probability with Permutations and Combinations: A Deeper and More Thorough Look at the Fundamental Equations*. California, USA, CreateSpace Independent Publishing Platform, 2017.

[12] Nicolaides A. *Pure Mathematics Series: 10. Combinations, Permutations, Probabilities*. London, UK, PASS Publications, 1994.

[13] Puntambekar A. *Analysis and design of algorithms.* 1st ed. Pune, India, Technical Publications, 2008.

[14] Mueller J, Massaron L. Algorithms for Dummies. Hoboken, USA, John Wiley & Sons, 2017.

[15] Goodrich MT, Tamassia R, Goldwasser, MH. *Data Structures and Algorithms in Java*. 6th ed. Hoboken, USA, John Wiley & Sons, 2014.

[16] Saha S, Shukla S. *Advanced Data Structures: Theory and Applications.* Boca Raton, USA, CRC Press: Taylor & Francis, 2019.

Appendix A

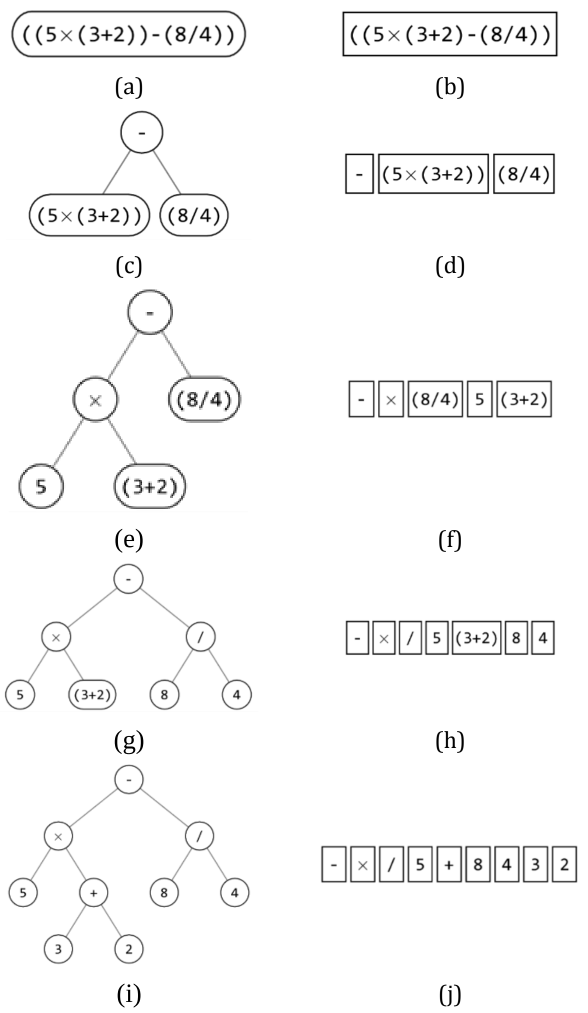


Figure A1. Steps for converting string expression to tree structure. The left column shows the tree structure, and the right column is the memory model adapted for the queue data structure.

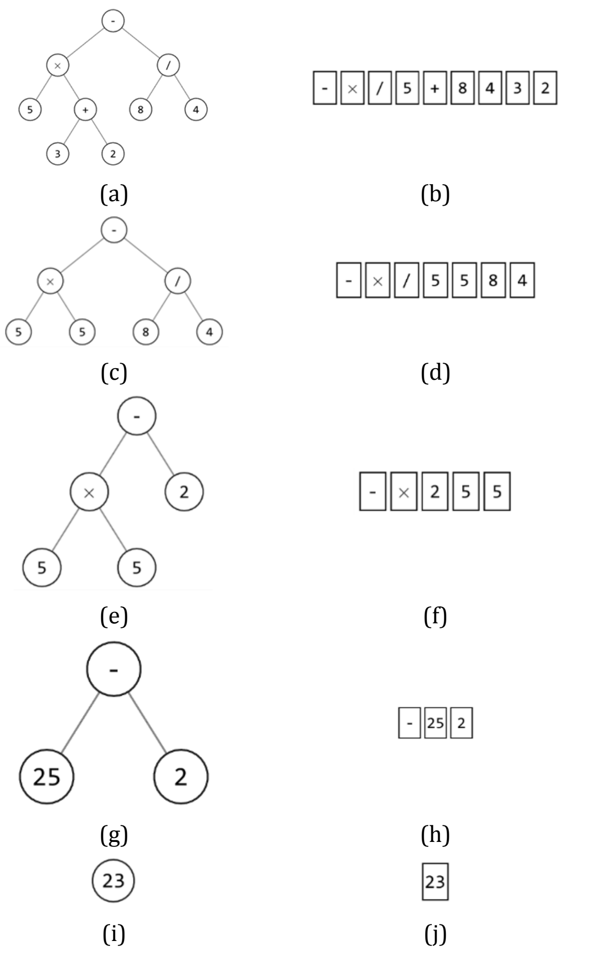


Figure A2. Steps for converting tree structure to value. The left column shows the tree structure, and the right column is the memory model adapted for the queue data structure.

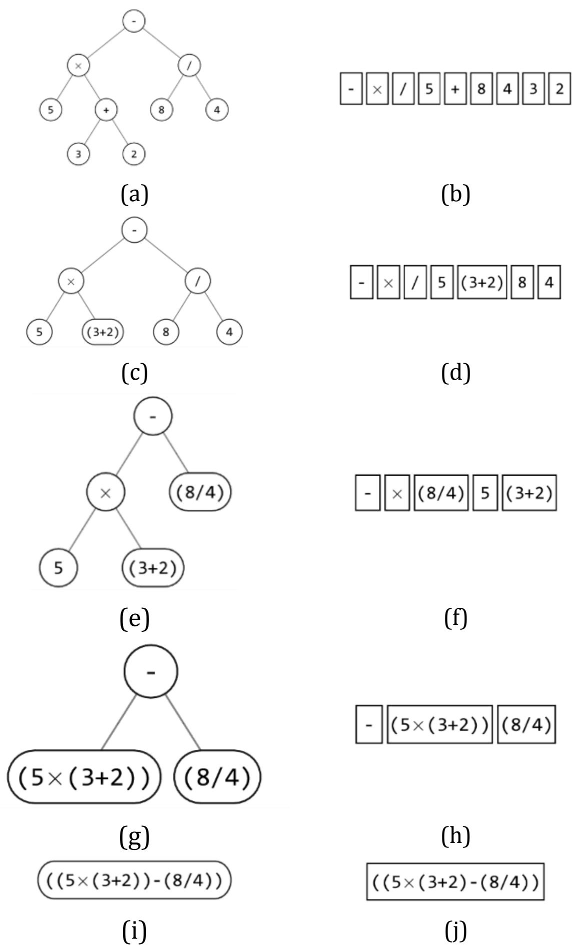


Figure A3. Steps for converting tree structure to string expression. The left column shows the tree structure, and the right column is the memory model adapted for the queue data structure.