Нахождение в массиве пары элементов с заданной суммой [1]

Описание задачи: для неотсортированного целочисленного массива найдите пару элементов с заданной суммой в нем.

**Например**:

Вход:  
   
arr = [8, 7, 2, 5, 3, 1]  
sum = 10  
   
Выход:  
 пара найдена (8, 2)  
или  
пара найдена (7, 3)  
  
Вход:  
arr = [5, 2, 6, 8, 1, 9]  
sum = 12  
  
Выход: пара не найдена

Есть несколько методов решения этой задачи с использованием грубой силы, сортировки и хеширования.

1. Использование метода полного перебора (brute force)

Самое простое решение - рассматривать каждую пару в данном массиве и возвращать, если желаемая сумма найдена. Продемонстрируем подход на языке C# [2]:

using System;

namespace ConsoleApp

{

public class Program

{

// Метод нахождения пары значений в массиве с заданной суммой

public static void FindPair(int[] A, int target)

{

// проходим все элементы кроме последнего

for (var i = 0; i < A.Length - 1; i++)

{

// проходим, начиная с i-того элемента до последнего включительно

for (var j = i + 1; j < A.Length; j++)

{

// если требуемая сумма найдена, выводим результат на консоль

if (A[i] + A[j] == target)

{

Console.WriteLine("Pair found (" + A[i].ToString() + ", " + A[j].ToString() + ")");

return;

}

}

}

// если доходим до этого участка кода, то пара не найдена

Console.WriteLine("Пара не найдена");

}

public static void Main(String[] args)

{

int[] arr = {8 , 7 , 2 , 5 , 3 , 1};

var target = 10;

FindPair(arr, target);

Console.ReadKey();

}

}

}

Временная сложность приведенного выше решения составляет O (n2) и не требует дополнительного места, где n - размер ввода.

2. Использование сортировки

Идея состоит в том, чтобы отсортировать данный массив в порядке возрастания и сохранить пространство поиска, сохраняя значения двух индексов (low и high), которые изначально указывают на две крайние точки массива. Затем необходимо уменьшать пространство поиска arr[low…high] на каждой итерации цикла, сравнивая сумму элементов, присутствующих в индексах low и high с искомой суммой. Увеличиваем значение индекса low на единицу, если сумма меньше ожидаемой; в противном случае уменьшаем значение индекса high на единицу, если сумма больше желаемой.

Ниже приводится реализация на C#, основанная на этой идее [3]:

using System;

namespace ConsoleApp

{

public class Program

{

// Метод нахождения пары значений в массиве с заданной суммой

public static void FindPair(int[] A, int target)

{

// сортируем массив в порядке возрастания значений

Array.Sort(A);

// Сохраняем два индекса, указывающих на крайние точки массива

var low = 0;

var high = A.Length - 1;

// уменьшаем пространство поиска `A[low ... high]` на каждой итерации цикла

// цикл идет до тех пор, пока пространство поиска не будет исчерпано

while (low < high)

{

// сумма найдена

if (A[low] + A[high] == target)

{

Console.WriteLine("Pair found (" + A[low].ToString() + ", " + A[high].ToString() + ")");

return;

}

// инкрементируем значение индекса `low`, если сумма проиндексированных элементов меньше, чем требуемая сумма;

// декрементируем значение индекса `high`, если сумма проиндексированных элементов больше, чем требуемая сумма;

if (A[low] + A[high] < target)

{

low++;

}

else

{

high--;

}

}

// Мы достигли этого места в коде, если пара не найдена

Console.WriteLine("Пара не найдена");

}

public static void Main(String[] args)

{

int[] A = { 8 , 7 , 2 , 5 , 3 , 1 };

var target = 10;

FindPair(A, target);

Console.ReadKey();

}

}

}

Временная сложность вышеуказанного решения составляет O (n.log (n)) и не требует дополнительного объема памяти.

3. Использование хеширования

 Мы можем использовать хеш-таблицу для решения этой проблемы за линейное время. Идея состоит в том, чтобы вставить каждый элемент массива arr[i] в словарь типа *Map* (ключ-значение). Мы также проверяем, существует ли разница (arr[i], sum - arr[i]) в словаре или нет. Если разница встречалась раньше, выводим результат на консоль и возвращаемся. Алгоритм может быть реализован следующим образом на C# [4]:

using System;

using System.Collections.Generic;

namespace ConsoleApp

{

public class Program

{

// Метод нахождения пары значений в массиве с заданной суммой с использованием хэширования

public static void FindPair(int[] A, int target)

{

// создаем пустой словарь

var map = new Dictionary<int, int>();

// проходим по всему полученному массиву

for (var i = 0; i < A.Length; i++)

{

// проверяем, существует ли пара `(A[i], target-A[i])`

// если разница уже встречалась раннее, выводим значение на консоль и выходим

if (map.ContainsKey(target - A[i]))

{

Console.Write("Пара найдена ({0}, {1})", A[map[target - A[i]]], A[i]);

return;

}

// сохраняем значение индекса текущего элемента в словаре

map[A[i]] = i;

}

// мы достигли этого места в коде, если пара не найдена

Console.WriteLine("Пара не найдена");

}

public static void Main(String[] args)

{

int[] A = { 8 , 7 , 2 , 5 , 3 , 1 };

var target = 10;

FindPair(A, target);

}

}

}

Временная сложность приведенного выше решения составляет O (n) и требует дополнительного объема памяти O(n), где n - размер входных данных.

**Упражнение**: расширите решение, чтобы вывести все пары в массиве с заданной суммой.

**Источники**:

1. Find a pair with the given sum in an array, https://www.techiedelight.com/find-pair-with-given-sum-array/
2. Использование метода полного перебора (brute force), https://github.com/AlexeiBuravcev/Algorithms/tree/main/Sections/Array/Find%20pair%20with%20given%20sum%20in%20the%20array/Brute-Force
3. Использование сортировки, <https://github.com/AlexeiBuravcev/Algorithms/tree/main/Sections/Array/Find%20pair%20with%20given%20sum%20in%20the%20array/Sorting>
4. Использование хэширования,