Задание № 3 Вспомогательный метод

Исходный код:

```
public static class HashSetExtensions
{
  public static IReadOnlyList<T> ConvertToList<T>(this HashSet<T> hashSet)
  {
     var result = new List<T>();

     foreach (var value in hashSet)
     {
        result.Add(value);
     }

     return result;
  }
}
```

Решение:

В данном методе элементы добавляются по одному с помощью List.Add. Если текущий массив списка переполнится (что случится при таком количестве элементов), то будет происходить выделение новой памяти. Это плохо скажется на производительности.

1. Чтобы улучшить ситуацию, можно заранее проинициализировать result с нужной емкостью. Тогда память для списка выделится предварительно. Пример:

```
var result = new List<T>(hashSet.Count);
```

2. Если важно только чтение и не нужен доступ по индексу, то можно воспользоваться другим подходом без создания лишней копии элементов, например IEnumerable<T>. Пример:

```
public static class HashSetExtensions
{
   public static IEnumerable<T> ConvertToList<T>(this HashSet<T> hashSet)
   {
      return hashSet;
   }
}
```

3. Можно воспользоваться готовым решением из библиотеки LINQ. Метод ToList() преобразует коллекцию в список. В таком случае память выделяется предварительно (как и в пункте 1), что убирает проблему с перераспределением памяти. Пример:

```
public static class HashSetExtensions
{
   public static IReadOnlyList<T> ConvertToList<T>(this HashSet<T> hashSet)
   {
      return hashSet.ToList();
   }
}
```