

Решение тестового задания на стажировку в СКБ Контур (разработчики)

Схема отбора на стажировку

- Кандидатам были предложены две задачи. Их условия можно найти [здесь](#). До 15 мая можно было отправлять решения этих задач.
- Среди кандидатов, набравших по итогам проверки наибольшее общее количество баллов, отбор будет проводиться по итогам собеседований.

Пожалуйста, отправляйте любые вопросы и предложения по схеме стажировки или тестовым заданиям на kontur-student@skbkontur.ru

Задача 1. Pagination

Для решения этой задачи требовалось реализовать хранилище целых чисел, которое позволяет быстро выполнять операции:

- а) добавления целого числа
- б) нахождения элементов с L -го по R -й в порядке неубывания.

Решение участников запускалось на наборе тестов. При этом измерялось время работы программы на каждом тесте. Тест считался пройденным, если программа выдавала на нём правильный ответ и работала не больше 10 секунд.

Набор тестов состоял из трёх частей:

- (а) тесты 1-10: пример из условия, небольшие тесты на крайние случаи
- (б) тесты 11-20: количество запросов на добавление числа не превосходит 100
- (в) тесты 21-25: общее количество запросов не больше 10^5 .

Для получения полного набора тестов можете использовать этот [архив](#).

Решения оценивались по 5-балльной шкале следующим образом:

- 0 баллов** – решение работает неверно почти на всех тестах группы (а)
- 1 балл** – в решении есть мелкая ошибка
- 2 балла** – решение проходит тесты группы (а), но работает медленно на группе (б)
- 3 балла** – решение проходит тесты групп (а) и (б), но работает медленно на группе (в)
- 4 балла** – решение проходит почти все тесты
- 5 баллов** – решение проходит все тесты.

Чтобы получить 2 балла, достаточно было написать практически любую реализацию хранилища (например, просто хранить все числа в массиве, а для выдачи чисел с L -го по R -е сортировать этот массив). В тестах группы (б) ограничено количество запросов на добавление – их всего 100. Поэтому можно было сортировать массив длины N за время $O(N \log N)$ каждый раз при *добавлении* числа – этого хватало для получения 3 баллов.

Чтобы решить задачу на полный балл, нужно было воспользоваться более сложными структурами данных. Например, [сбалансированными двоичными деревьями поиска](#), которые позволяют добавлять элемент в хранилище и находить i -й по порядку элемент в хранилище размера N за время $O(\log N)$. Также можно было заранее считать все числа, которые встречаются в запросах, и построить на множестве этих чисел [дерево отрезков](#), которое также позволяет отвечать на запросы за время $O(\log N)$. Многие кандидаты пользовались обычным связным списком (`List<int>`), хорошая реализация которого в языке C# помогала им набрать 4 балла. Всего задачу на полный балл сдали **7 человек**, а ещё **9 человек** набрали 4 балла.

Задача 2. Generics

Жюри выделило 5 категорий тестов, которые должны были присутствовать в тестовом наборе:

1. Тесты на некорректное использование метода (передача null-параметров, не-generics типов и пр.)

Примеры:

```
CloseGeneric(null,null),
CloseGeneric(typeof(int),typeof(string)),
CloseGeneric(typeof(IEnumerable<>), typeof(IEnumerable<>)).
```

2. Тесты на простейшие случаи, 1-2 generic-параметра.

Примеры:

```
public interface I1<T>{}
public class C1<T1> : I1<T1>{}
public interface I2<T1,T2>{}
public class C2<T1> : I2<T1,T1>{}
CloseGeneric(typeof(C1<>),typeof(I1<int>)),
CloseGeneric(typeof(C2<>),typeof(I2<int,int>)),
CloseGeneric(typeof(C2<>),typeof(I2<int,string>))
```

3. Тесты на ограничения (constraints) в описании классов и интерфейсов.

Примеры:

```
public interface I1{}
public interface I2<T>{}
public class C2<T> : I2<T> where T : I1{}
CloseGeneric(typeof(C2<>),typeof(I2<int>))
```

4. Тесты на случаи нескольких generic-параметров, когда порядок описания generic-параметров меняется.

Примеры:

```
public interface I1<T1,T2,T3>{}
public interface I2<T1,T2,T3> : I1<T3,T2,T1>{}
public class C2<T1,T2,T3> : I2<T2,T1,T3>{}
CloseGeneric(typeof(C2<int,,long>),typeof(I1<int,string,long>))
CloseGeneric(typeof(C2<,string,>),typeof(I1<long,int,string>))
```

5. Тесты на сложные запутанные иерархии

Примеры:

```
public interface I1<T1, T2>{}
public class C1<T1, T2> : I1<T2, T1>{}
public class C2<T> : I1<T, long>{}
public class C3<T1, T2> : I1<T1, int>{}
public interface I2Base<T1, T2, T3>{}
public interface I2Impl<T1, T2> : I2Base<string, T2, T1>{}
public interface I3 : I2Impl<Guid, int>{}
public class C5 : I3{}
public class C4<T1, T2> : I2Impl<T2, T1>{}
public class C4A<T1, T2> : I2Impl<T1, T2>{}
public class C4Derived<T1, T2> : C4<T1, T2>{}
```

```
CloseGeneric(typeof(C4<,>),typeof(I2Base<string,int,long>)),  
CloseGeneric(typeof(C4A<,>),typeof(I2Base<string,int,long>)),  
CloseGeneric(typeof(C4<,>),typeof(I2Base<int,int,int>))
```

Наличие тестов каждой из описанных групп оценивалось в 1 балл. Соответственно, можно было набрать от 0 до 5 баллов. Стоит также отметить, что автоматически проверялось «оформление работы»: если в вашей работе присутствовали тесты какой-то из групп, но жюри их не заметило, то это значит, что описания были оформлены недостаточно хорошо. В программировании важно просто и доходчиво объяснять свои мысли ☺

Некоторые участники обнаружили в формулировке задачи неточность, позволяющую реализовать функцию `CloseGeneric` с помощью создания наследника от `Type` с переопределёнными методами `IsAssignableFrom` и `GetGenericTypeDefinition`. С одной стороны, это означает, что участники не вполне прониклись сутью задачи и не поняли, где может применяться метод `CloseGeneric`. С другой стороны, это лишь догадки жюри, поэтому за внимательность обнаружение данной неточности оценивалось также в 1 балл.

Полный балл по второй задаче не получил никто, а **4 кандидата** получили по ней 4 балла.