DS-7. АиСД. 2.

16 янв 2022, 18:46:00
старт: 12 дек 2021, 12:14:01
финиш: 1 янв 2022, 14:14:01
длительность: 20д. 2ч.
начало: 12 дек 2021, 12:14:01
конец: 1 янв 2022, 14:14:01

A. Очередь с приоритетами (максимум)

Ограничение времени	4 секунды
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Напишите программу, которая будет обрабатывать последовательность запросов таких видов:
CLEAR — сделать очередь с приоритетами пустой (если в очереди уже были какие-то элементы, удалить все). Действие происходит только с данными в памяти, на экран ничего не выводится.

ADD n — добавить в очередь с приоритетами число *n* (вмещается в стандартный тип int). Действие происходит только с данными в памяти, на экран ничего не выводится.

EXTRACT — вынуть из очереди с приоритетами максимальное значение. Следует и изменить данные в памяти, и вывести на экран или найденное максимальное значение, или, если очередь была пустой, слово "CANNOT" (большими буквами).

Формат ввода

Во входных данных записано произвольную последовательность запросов CLEAR, ADD и EXTRACT — каждый в отдельной строке, согласно вышеописанному формату.Суммарное количество всех запросов не превышает 200000.

Формат вывода

Для каждого запроса типа EXTRACT выведите на стандартный выход (экран) его результат (в отдельной строке).

Пример

Ввод	Вывод

Ввод	Вывод
ADD 192168812	192168812
ADD 125	321
ADD 321	555
EXTRACT	7
EXTRACT	CANNOT
CLEAR	
ADD 7	
ADD 555	
EXTRACT	
EXTRACT	
EXTRACT	

Примечания

Следует использовать стандартную реализацию очереди с приоритетами в STL; она называется priority_queue, для её использования необходимо подключить заголовочный файл queue.

Язык Python 3.7.3

Набрать здесь | Отправить файл

```
class Heap:
    def __init__(self):
        self.tree_ = list('0')
        self.size_ = len(self.tree_)-1

    def add_number(self, inp):
        self.size_ += 1
        self.tree_.insert(self.size_, int(inp))
        p = self.size_
        while p>1 and self.tree_[p // 2] < self.tree_[p]:
            self.tree_[p // 2], self.tree_[p] = self.tree_[p], self.tree_[p // 2]
            p = p // 2

    def extract(self):
        if self.size_ == 0:
            return 'CANNOT'
        else :
            value = self.tree_[1]
            self.tree_[1] = self.tree_[self.size_]
            self.size_ -= 1
            p_ex = 1
            while p_ex * 2 <= self.size_:
                if p_ex * 2 == self.size_: #влево
                    max_idx = p_ex * 2
                elif self.tree_[p_ex*2] > self.tree_[p_ex*2+1]:
                    max_idx = p_ex*2
                else:
                    max_idx = p_ex * 2 + 1
                if self.tree_[p_ex] < self.tree_[max_idx]:
                    self.tree_[p_ex], self.tree_[max_idx] = self.tree_[max_idx], self.tree_[p_ex]
                p_ex = max_idx
            return value

    def clear(self):
        self.__init__()

a = Heap()
```

Отправить

Следующая