Spice IT Recruitment 59,46
ИТ специализированное кадровое агентство

22 мая в 16:43

Выпуск#2: ITренировка — актуальные вопросы и задачи от ведущих компаний

Программирование*, Занимательные задачки, Блог компании Spice IT Recruitment

На этой неделе мы публикуем подборку из задач и вопросов, которые даёт на собеседованиях **Uber**. Задачи подобрали различного уровня сложности от «Easy» до «Hard», чтобы всем было интересно. Условие дано на английском языке.

Ответы, как и прошлый раз, опубликуем в течение недели. Круто, если вы будете писать в комментариях свои варианты решений)

Вопросы:

1. Какие KPI вы бы использовали, если бы запустили новый сервис Uber в определенной части мира и хотели знать, насколько он успешен?

2. Какой проект, над которым вы работали, провалился? Могли бы вы сделать что-нибудь, чтобы предотвратить его провал?

Задачи:

1.

> Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
>
> push(x) — Push element x onto stack.
> pop() — Removes the element on top of the stack.
> top() — Get the top element.
> getMin() — Retrieve the minimum element in the stack.

Example:

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin();   --> Returns -3.
minStack.pop();
minStack.top();         --> Returns 0.
minStack.getMin();   --> Returns -2.
```

2.

> Design a data structure that supports all following operations in average O(1) time.
> insert(val): Inserts an item val to the set if not already present.
> remove(val): Removes an item val from the set if present.
> getRandom: Returns a random element from current set of elements. Each element must have the same probability of being returned.

Example:

```
// Init an empty set.
RandomizedSet randomSet = new RandomizedSet();

// Inserts 1 to the set. Returns true as 1 was inserted successfully.
randomSet.insert(1);

// Returns false as 2 does not exist in the set.
randomSet.remove(2);

// Inserts 2 to the set, returns true. Set now contains [1,2].
randomSet.insert(2);

// getRandom should return either 1 or 2 randomly.
randomSet.getRandom();
```

```
// Removes 1 from the set, returns true. Set now contains [2].
randomSet.remove(1);

// 2 was already in the set, so return false.
randomSet.insert(2);

// Since 2 is the only number in the set, getRandom always return 2.
randomSet.getRandom();
```

3.

> Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.
> Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should wo just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.
> For example, you may serialize the following tree
> 1
> / \
> 2 3
> / \
> 4 5
> as "[1,2,3,null,null,4,5]", serializes a binary tree. You do not necessarily need to follow this format, so please be creative and come up with differ approaches yourself.
> Note: Do not use class member/global/static variables to store states. Your serialize and deserialize algorithms should be stateless.

uber, SpiceIT, ITренировка, собеседование

+9 3,1k 39

Автор: @SpiceIT

Spice IT Recruitment рейтинг 59,46
ИТ специализированное кадровое агентство

Яндекс.Директ

Nissan Almera от 476 000 руб.
Надежный семейный седан с выгодой до 150 000 руб. Кредит 3% на 3 года!
nissan.ru Адрес и телефон
Содействие в подборе финансов...

Курсы английского любого уровня!
Английский язык за 3 месяца? Легко! -30% на курс "Интенсив" для взрослых.
ef.ru Адрес и телефон
Казань

Комментарии (8)

nickolaym 24 мая 2017 в 15:49

Стек с O(1) доступом к минимальному элементу:
Заведём два стека. Один — с актуальными данными, а другой — с минимумами.

```
data = [] # на питоне стек дёшево делается из списка
mins = [] # инвариант: mins[i] = min(data[0:i+1])

def empty() : return not data
def gettop() : return data[~0] # идиома: ~0 = -1, а -1 - это первый с конца
def getmin() : return mins[~0]

def push(v) :
  # минимум из расширенного набора равен
  # минимуму из
  #   минимума старого набора (мы его уже знаем)
  #   и нового элемента
  m = getmin() if not empty() and getmin() < v else v
  data.append(v)
  mins.append(m)

def pop():
  mins.pop()
  return data.pop() # на тот случай, если нам надо не выбросить, а ещё и вернуть выброшенное
```

Если данные тяжёлые, то в стеке минимумов можно хранить не сами данные, а, например, индексы. Заодно и ссылочную уникальность соблюдём.

```
data = []
mins = []

def empty() : return not data
def gettop() : return data[~0]
def getmin() : return data[mins[~0]]

def push(v) :
  # индекс нового минимума - тот же или индекс нового элемента
  m = mins[~0] if not empty() and getmin() < v else len(data)
  data.append(v)
  mins.append(m)
```

Правда, если у нас есть несколько одинаковых минимальных элементов, то вопрос, на кого из них отдавать ссылку, остаётся открытым. (Регулируется, в частн выбором оператора < или <=)

 **nickolaym** 24 мая 2017 в 16:44

Задача про множество уникальных элементов со случайным доступом.
Понятное дело, что это хеш-таблица.
А для того, чтобы эффективно реализовать случайный доступ, будем хранить элементы в сплошном массиве.

Можно сделать такую хеш-таблицу с самого начала (на трёх массивах), но это довольно муторное занятие. (Я делал в боевом коде, когда боролся за производительность).
Или внести небольшую избыточность, используя готовые хеш-таблицы из стандартных библиотек.

```
V = [] # вектор данных
D = {} # хеш-таблица - значение : индекс
# инвариант: V[D[v]] = v, D[V[i]] = i

def add(v) :
  if v in D :
    return False
  D[v] = len(V)
  V.append(v)
  return True

def pop(v) :
  if v not in D :
    return False
  i = D.pop(v)
  if i == len(V)-1 : # это последний элемент, берём и выкидываем
    V.pop()
  else : # переместим последний элемент на это место
    V[i] = V.pop()
    # и обновим хеш-таблицу
    D[V[i]] = i
  return True

R = random.Random()
def randvalue() : # случайный доступ
  return V[R.randint(0, len(V)-1)]
```

 **nickolaym** 24 мая 2017 в 17:54

Сериализация дерева.

Вот чисто для того, чтобы разнообразить подходы.

Каким способом мы можем строить дерево? Например, последовательностью команд стековой машины
— NUL — положить на стек нулевую заглушку
— NODE(«sss») — взять со стека два узла, создать корневой узел со значением «sss» и поддеревьями, положить обратно на стек

Каждая команда занимает один бит оператора плюс, если надо, операнд. Но вряд ли мы будем ужимать до битов, — то есть, каждый оператор — это це байт. Поэтому введём больше операторов:
— B(str) — узел-лист, без поддеревьев
— L(str) — узел с левым поддеревом
— R(str) — узел с правым поддеревом
— D(str) — узел с двумя поддеревьями

Сериализация — это восходящий обход дерева (если у нас дерево двусвязное, то он делается за линейное время и с константной памятью) и выдача соответствующих команд.
Десериализация — исполнение этих команд.

```
#-*- encoding:utf-8

# узел дерева - это тройка (строка, левое, правое)
# на вход подаётся корневой узел
```

```
def serialize(node) :
  v, l, r = node
  # наше дерево односвязное, поэтому спускаемся рекурсивно, фиг с ним.
  if l :
    for s in serialize(l) : yield s # это питонья идиома - монада списка, если кто не узнал ;)
  if r :
    for s in serialize(r) : yield s
  # пишем команду
  c = "DRLB"[(not l)*1 + (not r)*2] # немножко колдунства ;)
  yield c + repr(v)

def serialize_tree(node) :
  for s in serialize(node) : yield s
  yield "." # команда останова

def deserialize_tree(ss) :
  #ss = iter(ss) # гарантированно превратим ss в итератор
  nodes = []
  while True :
    cs = next(ss) # команда
    c = cs[0]
    if c == '.' :
      break
    v = eval(cs[1:]) # строка
    # интерпретируем:
    l, r = None,None
    if c == 'B' :
      pass
    elif c == 'L' :
      l = nodes.pop()
    elif c == 'R' :
      r = nodes.pop()
    elif c == 'D' :
      r = nodes.pop()
      l = nodes.pop()
    else :
      assert False, "bad command %s %s" % (repr(c), repr(v))
    nodes.append((v,l,r))
  assert len(nodes) == 1
  return nodes[0]

##############################################

# тестовое дерево
T = ('1\na a', # строки могут содержать всё, что угодно
      ('"2"',None,None),
      ("'3'",
        ('4',None,None),
        ('5',None,None)))

print 'ORIGINAL:', T

# вот так это сериализуется в поток
with open('tree.txt', 'w') as f :
  for sss in serialize_tree(T) : print >>f, sss

# а вот так - в строку
S = '\n'.join(serialize_tree(T))
print S

# вот так десериализуется из строки
T1 = deserialize_tree(iter(S.splitlines()))
print 'FROM STR:', T1

# а вот так - из потока
with open('tree.txt') as f :
  T2 = deserialize_tree((s[:-1] for s in f))
print 'FROM TXT:', T2
```

sverhnovaia 29 мая 2017 в 12:49

@nickolaym Хэй, спасибо за активность и решения :)
Ниже будет вариант, взятый с исходного ресурса.

Nikolay_Ch 29 мая 2017 в 12:49

Там будет достаточно одного стека, только поле данных должно содержать два поля: основное — собственно хранящееся значение и вспомогательное поле значение минимума, которое было до текущего значения. Тогда минимум всегда хранится в ТОПовом элементе во вспомогательном поле, а все операции с будут выполняться за константное время.

nickolaym 1 июня 2017 в 14:33

А, ну да, стек пар изоморфен моей паре стеков :)

horsones 24 мая 2017 в 18:04

Было бы полезно для каждой задачи указывать ссылку на LeetCode и список компаний, которые спрашивают этот вопрос (Список основных вопросов крупных компаний):

- 155. Min Stack Google, Uber, Amazon и др.
- 380. Insert Delete GetRandom O(1) Согласно статистике по ссылке выше не очень частый вопросб либо появился недавно.
- 297. Serialize and Deserialize Binary Tree Amazon, Facebook и др.

Тем более что текст задач 1 в 1 совпадает.

sverhnovaia 29 мая 2017 в 13:14 *(комментарий был изменён)*

Радует, что вы решаете задачи :)
Ниже публикуем по одному решению с исходного ресурса. Текст оставили на английском, чтобы избежать недопонимания.

Atention! Spoiler!

1.Решение на Java с использованием одного стека:

```
class MinStack {
    int min = Integer.MAX_VALUE;
    Stack<Integer> stack = new Stack<Integer>();
    public void push(int x) {
        // only push the old minimum value when the current
        // minimum value changes after pushing the new value x
        if(x <= min){
            stack.push(min);
            min=x;
        }
        stack.push(x);
    }

    public void pop() {
        // if pop operation could result in the changing of the current minimum value,
        // pop twice and change the current minimum value to the last minimum value.
        if(stack.pop() == min) min=stack.pop();
    }

    public int top() {
        return stack.peek();
    }

    public int getMin() {
        return min;
    }
}
```

2. Простое решение на Python
We just keep track of the index of the added elements, so when we remove them, we copy the last one into it.

From Python docs (https://wiki.python.org/moin/TimeComplexity) we know that list.append() takes O(1), both average and amortized. Dictionary get and set functions tak average, so we are OK.

```
import random

class RandomizedSet(object):

    def __init__(self):
        self.nums, self.pos = [], {}

    def insert(self, val):
        if val not in self.pos:
            self.nums.append(val)
            self.pos[val] = len(self.nums) - 1
            return True
        return False


    def remove(self, val):
        if val in self.pos:
            idx, last = self.pos[val], self.nums[-1]
            self.nums[idx], self.pos[last] = last, idx
            self.nums.pop(); self.pos.pop(val, 0)
            return True
        return False
```

```
    def getRandom(self):
        return self.nums[random.randint(0, len(self.nums) - 1)]

# 15 / 15 test cases passed.
# Status: Accepted
# Runtime: 144 ms
```

3. Решение на C++

```
class Codec {
public:
    // Encodes a tree to a single string.
    string serialize(TreeNode* root) {
        if (root == nullptr) return "#";
        return to_string(root->val)+","+serialize(root->left)+","+serialize(root->right);
    }

    // Decodes your encoded data to tree.
    TreeNode* deserialize(string data) {
        return mydeserialize(data);
    }
    TreeNode* mydeserialize(string& data) {
        if (data[0]=='#') {
            if(data.size() > 1) data = data.substr(2);
            return nullptr;
        } else {
            TreeNode* node = new TreeNode(helper(data));
            node->left = mydeserialize(data);
            node->right = mydeserialize(data);
            return node;
        }
    }
private:
    int helper(string& data) {
        int pos = data.find(',');
        int val = stoi(data.substr(0,pos));
        data = data.substr(pos+1);
        return val;
    }
};`
```

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ Разра

Сейчас Сутки Неделя Месяц

Да, Python медленный, но меня это не волнует
+38 14k 84 107

Атака на АБ-тест: рецепт 'R'+t(101)+'es46'
+93 9,7k 50 45

StringBuffer, и как тяжело избавиться от наследия старого кода
+15 5,1k 21 19

Поиск по дереву методом Монте-Карло и крестики-нолики
+9 1,9k 28 0

Как поменьше беспокоиться о собственной бездарности
+69 31,2k 208 146

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Orange Pi 2G-IoT — идеальный одноплатник для IoT GT

2,4k 23 13

Pandemic: инструмент ЦРУ для скрытой подмены файлов GT

2,3k 5 0

Небесный тихоход: «Почта России» задумалась о собственной авиакомпании GT

2,7k 0 35

В преддверии Гейзенбага: Grid Dynamics о тестировании

801 7 0

Как сделано интро на 64k

4,7k 53 8