PureStorage 昂塞特

Updated Aug 31, 2017

吐槽一句，这种万年不变面试题目的风格，真的好吗。。。

http://www.mitbbs.com/article_t/JobHunting/32702941.html
第一轮：定义buddy system为一棵complete binary tree。一个node可能为0也可能为1
. 它的
value为1，当且仅当它所有的child的value均为1.

1

|

1 2

| |

1 2 3 4

| | | |

1 2 3 4 5 6 7 8

实现下列的method。

1' clearBit(int offset, int len);

2' setBit(int offset, int len);

Q: offset 和 len都是指最下面一层的，对吗？

A: yes

Q: 用二维数组bits[level][number]存储是他们给出的吗？

A: yes

Clarify：http://www.meetqun.com/thread-10165-1-1.html

就是一个二维数组，但是每层长度不一样，A[0]长度为1，A[1]为2，A[2]为4，依次类推。每个元素可以为0或为1，clearBit或者setBit都是说的最后一层从offset开始到offset+len-1置为1或0，但是同时也会影响到上面的元素，所以要迭代处理。

- 找到start 和 end. 对于上一层的tree, start /=2, end /=2 PS:这个对clear bit肯定可以

对set bit, 你要想想

有幸遇到国人大哥的话可能会是投篮题和设计LRU题，投篮题具体另一篇论坛里的帖子有说哈～

第二轮：设计一个task dispatching system，里面有一个task queue和两个function。

1’ trigger。这个function运行并清空task queue中所有的tasks。

2‘ addTask。在trigger之前把task加入task queue，在trigger之后直接运行task。

一些分析:

http://www.1point3acres.com/bbs/thread-124049-1-1.html

我个人觉得queue是否被trigger的区别在于这个queue是否为空，因为queue被trigger后整个queue就空了没有必要去增加一个全局变量。而且addTask（）和trigger（）应该是不冲突的，trigger（）总是取queue前面的一个task来run，而新的task总被加到queue的末尾，上面这样加锁system性能会很差。以上是我的写法，这样加锁可以使queue不为空时，能够同时trigger和addTask。而且可以保证queue里的最后一个task被取出来后，总是能够在addTask（task）的task之前运行。但是这样写有个bug就是当queue为空，多个addTask(task)被call，task永远没有办法被add进queue，会处于顺序执行状态。持有锁的那个addTask会run task。这样queue就失去了存在的意义。不知道我的写法应该怎么改进，求指导。

```
trigger() {
  while(!queue.isEmpty()) {
    lock(&m);
    get & remove task from queue;
    run task;
    unlock(&m);
  }
}
```

```
addTask(task) {
  if (queue.isEmpty()) {
    lock(&m);
    run task;
    unlock(&m);
  } else {
    queue.add(task);
  }
}
```

第三轮：产生一个圆上的所有坐标。不用sqrt, sin, cos等内建函数。

提示：所有的点都是整点。首先我们可以利用对称性把圆分成8块，先画出0-45度角内的点，然后映射之。对于其中0-45度角中的点，当X＋1时，Y的值或者不变或者－1，然后放入圆方程中看哪一个是对的。

第四轮：设计一个Map<Integer, Integer>，满足下面的复杂度。

add: O(1) deletion: O(1) lookup: O(1) clear:O(1) iterate: O(number of elements)。

提示：

如果我们用randomly accessed array，复杂度如下：

add: O(1) deletion: O(1) lookup: O(1) clear: O(size of array) iterate: O(size of array)

如果我么用sequential array, 复杂度如下：

add: O(1) deletion: O(number of elements) lookup:O(number of elements)

clear: O(1) iterate:O(number of elements)

所以我们需要把这两个方法整合起来。

第四题：http://www.mitbbs.com/article_t/JobHunting/32706095.html

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=166205&highlight=pure%2B%2Bstorage

特别提醒:

1. Mutex的题只要注意order就行了, 加锁的order..

2. 画圆的题请先练手.

3. buddy system有优化,考虑如何level update,而不是heap-like update.

我面的是O(1) Set和task dispatching system，看来他家题目永远是那4

http://www.1point3acres.com/bbs/forum.php?

mod=viewthread&tid=141925&highlight=pure%2B%2Bstorage

onsite 四轮：

第一轮问题：给一个ｍａｔｒｉｘ，求每个点周围平均值结果，每个点的平均值就是周围最多８个临近点和本身点的平均值

第二轮：　ｆｉｂｏｎａｃｃｉ　递归和非递归，　表达式求值，只有，加，减，乘和括号

第三轮：　应用题：ｆｏｏｔｂａｌｌ比赛有多种得分方式，ｔｏｕｃｈｄｏｗｎ　６分，之后加踢罚球　１分，再ｔｏｕｃｈｄｏｗｎ　３分，

给一个比分，问有几种得分方式，　还有算法复杂度

第四轮：　ｍｉｎ　ｓｔａｃｋ，不断优化

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=147859&highlight=pure%2Bstorage

贡献一下10月底的pure storage面经，给下午的电面攒点人品。

我是10月25号左右在pure storage官网投的简历，一个小时之后，recruiter给我OA的链接，我晚上花一个小时做完之后，第二天早上收到onsite的通知。

所以感觉这家效率很高，如果内推没信的话，感觉可以试试直接投。另外我感觉投简历的时候一定要注意，选择的职位最好和自己的简历贴合程度高一些，因为我感觉recruiter好像不怎么筛简历，看你差不多就给你oa的机会了，你如果简历和职位不怎么匹配的话，很可能onsite之后才会因为experience的问题把你拒了，挺耽误时间的。

OA的题是老题了，我感觉似乎他们现在全面换新题，所以这里就不说了，大家注意版上的新题吧，感觉难度老题大。

可能因为是local的，所以oa后没有电面，直接给我onsite了，两个小时，两道白板题。题出自四大经典题里面，四大题的链接在这里：

http://www.mitbbs.com/article_t1/JobHunting/32702941_32703031_1.html

问我的是第二题和第一题，有点变化，不过思路差不多是一样的。

第二题要注意多线程加锁的问题。他们家对于virtual function和多线程要求挺高的，面试之前多看看比较好

面试我的第一个是个白人manager，人挺nice，不过我对于这种问题不是很熟，所以有些要求没有听的太明白，耽误了一些时间，最后做出来了，面试多用了10分钟。

后来是一个印度小哥和一个华裔女生一起面第二轮，感觉还是比较顺利的，和印度小哥沟通挺顺畅，提示也很快能明白，问题比较快的解决了。然后聊了一些。

onsite两天后recruiter来电说拒了。个人感觉发挥的还行，不过估计面这家的强手如云，所以最后被刷也是正常的。

另外他家基本上工作的时候使用c或者c++，所以最好面试的时候也用C/C++。

再附加一个链接：

http://softwarecareerup.blogspot.com/2015/03/pure-storage-interview.html

这个链接感觉资源非常好，面他们家之前看看会有不少收获。

--

Giving a multi-thread stack java program, ask you to mark when should use lock or unlock and also ask you throw a runtime exception when some one want to pop an element from the stack but the stack is empty.

How would you build a test plan for a particular UI?

What is virtual memory? How does it work? What conditions should be tested in a VM system?

Was asked about mutexes, spinlocks, systems level code. Was asked to implement a mutex with a spinlock

design a data structure that can retrieve specific key/value pair at const time complexity. The interviewer for this question is an indian guy. He is quite rule and gave some useless hints that made you feel more confused.

''' pure storage buddy system bitmap

Given a complete binary tree with nodes of values of either 1 or 0, the following rules always hold:

(1) a node's value is 1 if and only if all its subtree nodes' values are 1

(2) a leaf node can have value either 1 or 0

Implement the following 2 APIs:

set_bit(offset, length), set the bits at range from offset to offset+length-1

clear_bit(offset, length), clear the bits at range from offset to offset+length-1

clear_bit(offset, length), clear the bits at range from offset to offset+length-1

Buddy Bitmap:

```
def setbit_down(A, x, n):
  if x>=n:
    return
  if 2*x+1<=n and
A[2*x+1]==0:
    A[2*x+1]=1
    setbit_down(A,2*x+1,n)
  if 2*x+2<=n and
A[2*x+2]==0:
    A[2*x+2]=1
    setbit_down(A,2*x+2,n)


def set_bit(A, pos, length):
  if not A or pos<0 or
length<=0:
    return
  n = len(A)-1    #last index of
A
  for x in range(pos,
min(n+1,min(pos+length,
2*pos+1))):
    # set self
    if A[x] == 1:
      continue
    A[x]=1
    # set descendants
    setbit_down(A,x,n)
    # set ancestors
    while x>0:
      # make sure its sibling
is 1, if its sibling is 0, cannot
set ancestors
      if (x%2==0 and A[x-
1]==1) or (x%2==1 and x<n
and A[x+1]==1):
        A[(x-1)/2] = 1
      x = (x-1)/2

def clear_bit(A, pos, length):
  if not A or pos<0 or
length<=0:
```

Draw a Circle

```
def draw_circle(r2):
  result = Set([])
  x = 1
  y = 0
  while x*x <= r2:
    for y in range(x+1):
      if x*x+y*y == r2:
        result.update(Set([(x,y),(x,-y),(-x,-y),(-x,y),
(y,x),(y,-x),(-y,-x),(-y,x)]))
    x+=1
  return result

def draw_circle_bi_search(r2):
  result = Set([])
  x = 1
  y = 0
  while x*x <= r2:
    y_start = 0
    y_end = x
    while y_start <= y_end:
      y_mid = y_start+(y_end-y_start)/2
      if x*x + y_mid*y_mid == r2:
```

```
        return
    n = len(A)-1    #last index of
A
    for x in range(pos, min(n+1,
pos+length)):
        # clear self
        if A[x]==0:
            continue
        A[x]=0
        # clear descendants
        while 2*x+1<=n:
            A[2*x+1] = 0
            x=2*x+1
        # clear ancestors
        while x>0:
            if A[(x-1)/2]==0:
                break
            A[(x-1)/2] = 0
            x = (x-1)/2

if __name__=='__main__':
    A=[0,0,1,1,0,1,1,1,1,1,0,1]
    test_cases = [(x,y) for x in
range(len(A)) for y in
range(1,len(A)-x+1)]

    for each_test_case in
test_cases:
        pos, length =
each_test_case
        A=[0,0,1,1,0,1,1,1,1,1,0,1]
        set_bit(A,pos, length)
        print 'after setting bit from
', pos, 'for ', length,'A is: ', A
        A=[0,0,1,1,0,1,1,1,1,1,0,1]
        clear_bit(A,pos, length)
        print 'after clearing bit from
', pos, 'for ', length,'A is: ', A
```

```
                result.update(Set([(x,y_mid),(x,-y_mid),(-
x,-y_mid),(-x,y_mid),(y_mid,x),(y_mid,-x),(-y_mid,-
x),(-y_mid,x)]))
                break
            elif x*x + y_mid*y_mid < r2:
                y_start = y_mid+1
            else:
                y_end = y_mid-1



        x+=1
    return result

if __name__=='__main__':
    profile.run('print draw_circle(1000000)')
    profile.run('print
draw_circle_bi_search(1000000)')
```