百度最新算法工程师笔试题

一、在图像处理中经常会用到卷积层。已知输入图片维度为 W*W, filter size 为 F*F， stride 为 S，padding 为 P，问经过一次卷积，输出特征图的宽度为

答案:
(W-F+2P)/S + 1

二、全高清视频的分辨率为 1920×1080P，如果一张 RGB 真彩色像素的 1920×1080 BMP 数字格式图像，所需存储空间是

答案:
5.93MB

三、请简要描述 GPU 的存储结构。CPU 与 GPU 之间怎样传输数据？试用 cuda 函数说明。

答案:
GPU 存储结构： Registers Shared memory Local memory Constant memory Texture memory Global memory 传输数据使用函数： cudaMemcpyHostToDevice cudaMemcpyDeviceToHost

四、简述流水线的三类冒险,并指出数据冒险有哪几类,以及解决数据冒险的方法。

答案:
流水线的三类冒险分别是: (1)结构冒险:当硬件在指令重叠执行中不能支持指令所有可能的组合时发生资源冒险。 (2)数据冒险:在同时执行的指令中,一条指令依赖于前一条指令的数据而得不到时发生的冒险。 (3)控制冒险:流水线中的转移指令或其他改写 PC 的指令造成的冒险。 其中有 3 类数据冒险:RAW(写后读):指令 j 试图在指令 i 写一个数据之前读取它, 这时 j 会读到错误的值,RAW 对应于数据的真相关;WAW(写后写):指令 j 试图在指令 i 写一个数据之前写该数据,留下的值将会是指令 i 的结果,WAW 对应于输出相关,只在特 定类型的流水线中才发生;WAR(读后写):指令 j 试图在指令 i 读一个数据之前写该数据, 这时指令 i 会错误的读出新值,WAR 对应于反相关,不会发生在静态流水线之中。 解决数据冒险的方法有:双跳(double bump);停顿(stall);转发(forwarding);指令 重排序(instruction reorder)。

五、

```
#include<iostream.h>
int c;
class A
{
  private:
int a;
static int b;
  public:
    A(){a=0; c=0;}
void seta(){a++}
void setb(){b++}
void setc(){c++}
void display(){cout<<a<<""<<b<<""<<c;}
 }
 int A::b = 0;
 void main()
 {
   A a1, a2;
   a1.seta();
   a1.setb();
   a1.setc();
   a2.seta();
   a2.setb();
   a2.setc();
   a2.display();
 }
```

答案:

122

六、完善以下代码，以给定 x 为基准，将链表分割为两部分，所有小于 x 的结点排在大于或等于 x 的结点之前。

```
public LinkedListNode partition(LinkedListNode node, int x)
{
  LinkedListNode beforeStart = null;
  LinkedListNode afterStart = null;

  while( node != null)
  {
LinkedListNode next =   1  ;
if (node.data < x)
{
  node.next =   2   ;
  beforeStart = node;
}
else{
     3  = afterStart;
  afterStart = node;
```

```
}
node = next
 }
 if(beforeStart == null){ return afterStart;}

 LinkedListNode head = beforeStart;
 while(beforeStart.next != null){
     4     = beforeStart.next;
 }
 beforeStart.next =   5   ;
 return head;
}
```

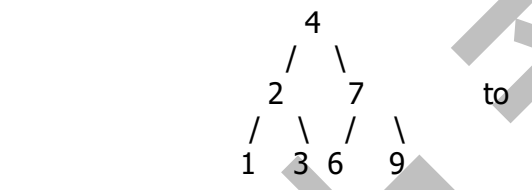
答案:

1.node.next 2. beforeStart 3.node.next 4. beforeStart 5.afterStart

七、翻转二叉树。树结点定义如下:

```
Struct TreeNode {
int val;
TreeNode * left;
TreeNode * right;
TreeNode(int x): val(x),left(NULL),right(NULL) {}
};
```

比如:



输入二叉树结构根节点指针 TreeNode* root,实现接口:

```
TreeNode* invertTree(TreeNode* root) ;
```

答案:

```
TreeNode* invertTree(TreeNode* root) {
        if(!root){return NULL;}
         TreeNode* temp = root->left;
         root->left = root->right;
         root->right = temp;
         invertTree(root->left);
         invertTree(root->right);
         return root;
     }
```

八、试用 cuda/opencl 等并行的方法实现快速排序算法。

答案:

```
__global__ void
```

```
cdp_simple_quicksort(unsigned int *data, int left, int right, int depth)
{
unsigned int *lptr = data+left;
unsigned int *rptr = data+right;
unsigned int  pivot = data[(left+right)/2];
// Do the partitioning.
while (lptr <= rptr)
{
// Find the next left- and right-hand values to swap
unsigned int lval = *lptr;
unsigned int rval = *rptr;
// Move the left pointer as long as the pointed element is smaller than the pivot.
while (lval < pivot)
{
lptr++;
lval = *lptr;
}
// Move the right pointer as long as the pointed element is larger than the pivot.
while (rval > pivot)
{
rptr--;
rval = *rptr;
}
// If the swap points are valid, do the swap!
if (lptr <= rptr)
{
*lptr++ = rval;
*rptr-- = lval;
}
}
// Now the recursive part
int nright = rptr - data;
int nleft  = lptr - data;
// Launch a new block to sort the left part.
if (left < (rptr-data))
{
cudaStream_t s;
cudaStreamCreateWithFlags(&s, cudaStreamNonBlocking);
cdp_simple_quicksort<<< 1, 1, 0, s >>>(data, left, nright, depth+1);
cudaStreamDestroy(s);
}
// Launch a new block to sort the right part.
if ((lptr-data) < right)
{
cudaStream_t s1;
cudaStreamCreateWithFlags(&s1, cudaStreamNonBlocking);
cdp_simple_quicksort<<< 1, 1, 0, s1 >>>(data, nleft, right, depth+1);
cudaStreamDestroy(s1);
}
}
```

九、是否了解 paddlepaddle, caffe2, tensorflow, pytorch，MXNet 等深度学习框架？请就框架结构、模型表述、执行机制或分布式训练等某一个方向对以上 2 个或以上框架进行对比。

怎样对 model 进行 fine-tuning 并将原有 model 的一些 node 从 graph 中剔除?

答案:

MXNet 是一款轻量级的深度学习框架，支持自动将计算任务并行化到多个 GPU 或分布式集群。

Tensorflow 的核心是 graph 和 session.它同样是一款支持分布式的深度学习框架。

其他区别由答题者自己发挥。

fine-tuning，以 tensorflow 为例，

1. 首先将 model 加载到 default graph 中

```
vgg_saver = tf.train.import_meta_graph(dir + '/vgg/results/vgg-16.meta')
```

2. 选择想要添加 connection 或改变的 node

```
output_conv = vgg_graph.get_tensor_by_name('conv1_2:0')
```

3. Stop gradient

```
output_conv_sg = tf.stop_gradient(output_conv)
```

4. 由该 node 添加新的操作

```
z1 = tf.nn.conv2d(output_conv_sg, W1, strides=[1, 1, 1, 1], padding='SAME')
```

将 node 从 graph 中移除可考虑 tf 提供的 remove_training_nodes 方法。