数字创想社2024学年招新选拔笔试参考答案

1 A

2 (1) C

2 (2) BD

2 (3) 略。困难的可信度较高，解决方案可行即可。

3 略。书写控制装置和翻页装置结构构思合理，伪代码逻辑合理即可。

4 (1) C

4 (2) B

4 (3) C

5 (1) D

5 (2) C

5 (3) C

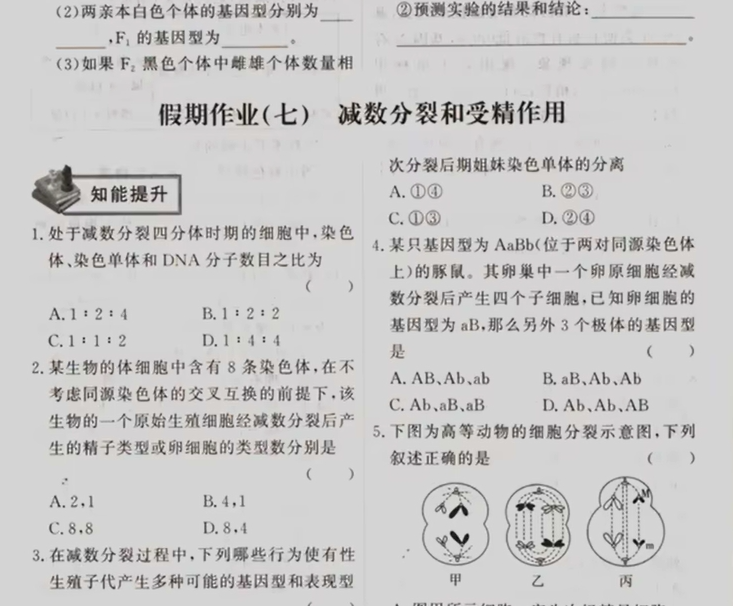
6 (1) 不能；停机问题是不可判定问题。

6 (2) 略。

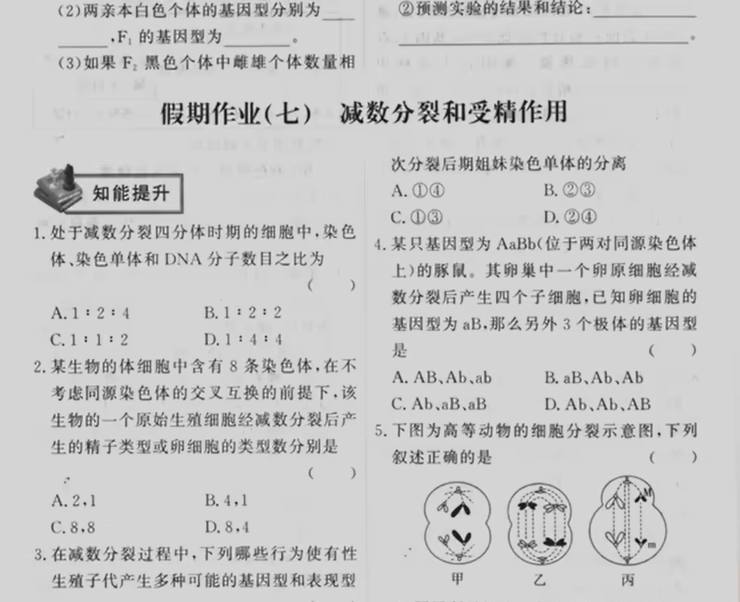
详细解析从第2页开始。

1

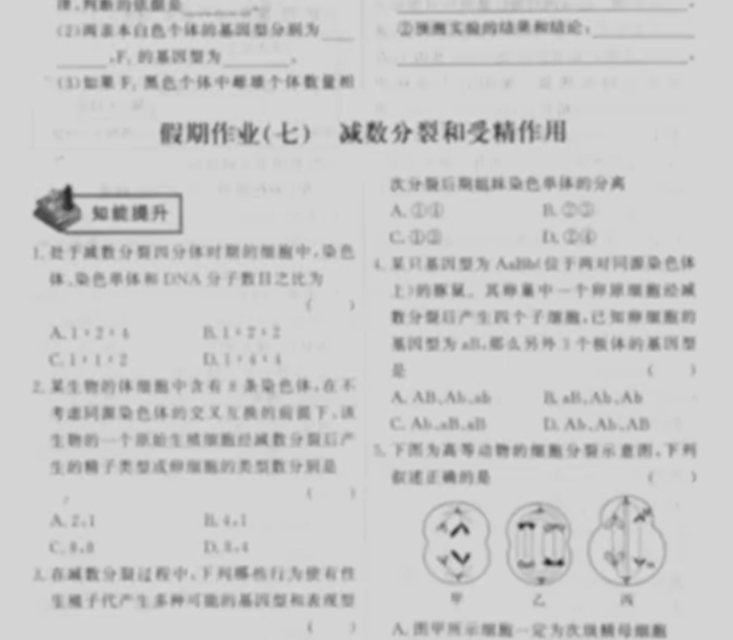
原图



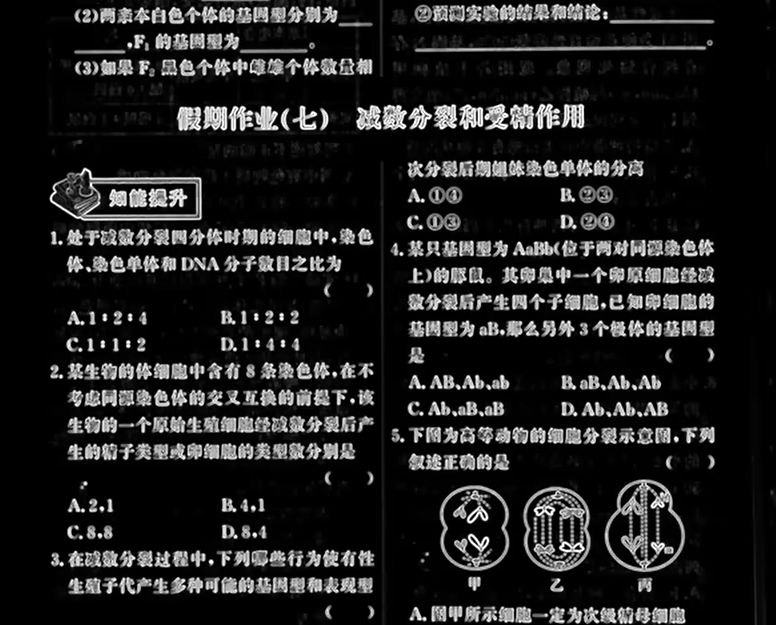
第一步：黑白



第二步：高斯模糊



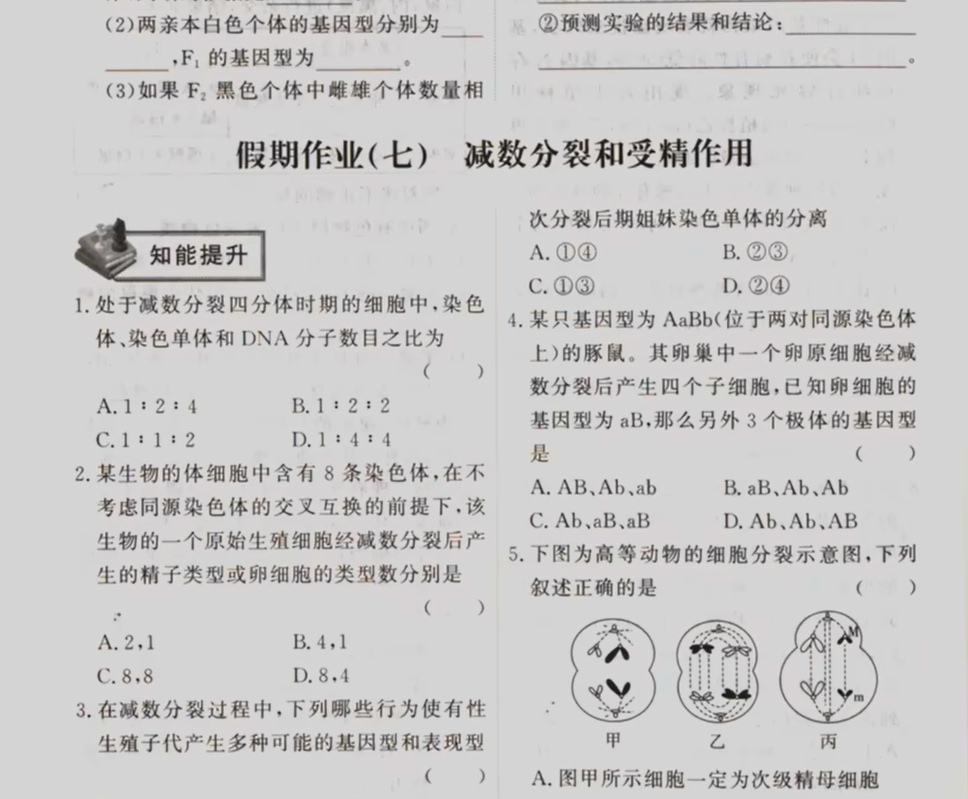
第三步：查找边缘



第四步：闭合图形



第五步：边角定位



是不是变清晰了？

2 (1) 目前的语言模型不擅长逻辑推理，故BD排除，GPT-4o-all由美国OpenAI公司开发，以英语为母语进行训练，故选C。

2 (2) Checkpoint是指在模型训练过程中保存的模型状态。这些检查点通常包括模型的参数（权重和偏置）、优化器状态和其他相关的训练信息；LoRA是一种高效、灵活且适用于多种场景的模型微调技术，它在保持原始模型性能的同时，允许用户根据需要进行定制化调整；Embbeding将能将离散数据映射为连续变量，捕捉潜在关系，提升模型性能，增强泛化能力，降低计算成本；ControlNet是一个控制预训练图像扩散模型（例如 Stable Diffusion）的神经网络。它允许输入调节图像，然后使用该调节图像来操控图像生成。故选BD。

2 (3) 略。困难的可信度较高，解决方案可行即可。

3 略。书写控制装置和翻页装置结构构思合理，伪代码逻辑合理即可。书写装置可以是在操作平台上安装一根横杆，可双向滑动，在横杆上有笔固定组件能够沿着横杆垂直方向双向滑动。翻页装置可以再安装一根横杆，旁边用一个固定风扇在书的开页侧吹风使书页不粘连，再用另一个装在横杆上的风扇反向吸起书页，带动它翻过一面。

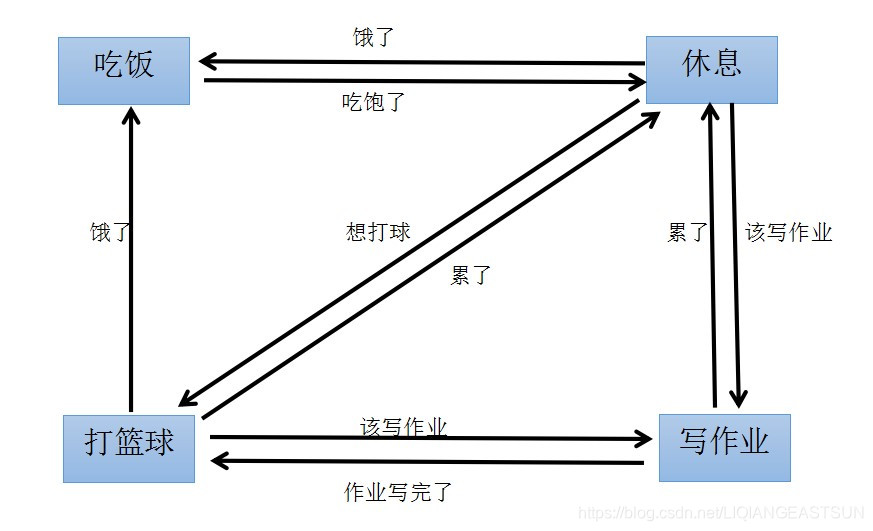
4 (1) B站的视频在4K分辨率下帧数为30 fps。故选C。

4 (2) 增加动态模糊虽使画面更“流畅”，但观感下降；添加更多元素治标不治本；降低画面帧率是烧杯行为，只有调整动画曲线才能让视觉元素平移更流畅。故选B。

4 (3) flac是一种无损压缩音频格式，无法在Adobe After Effects中导入。其它选项都可以做到。故选C。

5 (1) 题目答案数据量很大，冒泡排序的平均和最坏情况时间复杂度均为O ( n2 )，插入排序的最坏情况时间复杂度亦为O ( n2 )，对于大规模且随机分布的数据，它们的效率并不高。快速排序和归并排序的最好和最坏时间复杂度均为O ( nlog2n )，但快速排序不是稳定的排序算法，故选D。

5 (2) 有限状态机拥有有限数量的状态，每个状态代表不同的意义，每个状态可以切换到零-多个状态。任意时刻状态机有且只能处在一个状态。比如以下是以一个学生的四个状态（吃饭、休息、打篮球、写作业）构建的状态机：



Excel工作簿、Minecraft和康威生命游戏都是图灵完备（假设运行在无限内存的计算机中），可以拥有无限种不同的状态，不属于有限状态机，故选C。

5 (3) 题目序号构成偏序集，不能直接映射到数组下标。链表查询时间是O (n)，故选C。建议学习数据结构相关知识。

6 (1) 不能建造。停机问题是不可判定问题。

假如完全不考虑写作业机的内部结构（包含硬件结构和软件结构），预测机器的逻辑就是：

如果知道写作业机在有限时间内不会停机，就强制停止写作业机；

如果知道写作业机在有限时间内会停机，就耐心等待写作业机运行完毕；

所以上述预测写作业机是否会在有限时间内停机的机器不存在。

6 (2) 什么都不会发生。

根据题意，(1) 中的结论依然适用，预测停机的机器不可能存在。证明如下：

设一个程序为P，运行时需要传入一个参数，要运行它，记为P (参数)。

强调P的代码数据本身，记为 [P]。

定义停机自动判定程序H，它应该有两个参数。参数1：待判定的程序P，参数2：运行P程序时传入的参数。

（啰嗦一下：对于某个需要参数的程序P，你不能单纯说它会不会停机。有可能传入参数101的时候会停机，而传入42又死循环了，所以判定时必须把参数配齐。所以，停机自动判定程序H，它用起来应该是这种感觉：如果P(42)会死循环不停机，那么 H( [P], 42 ) 返回 false。如果P("apple")会正常结束，那么H( [P], "apple" ) 返回 true。）

再定义另一个程序U：

|  |
| --- |
| 定义程序 U( 参数 )  {  if ( H(参数,参数) == true )  {  while (true) { } // 死循环  }  else  {  return true; // 返回true，正常退出  }  } |

U是对停机判定程序H的简单包装。如果H返回了true，那么就陷入死循环；如果H返回了false，那就返回true。

U(参数) 会被转化成 H(参数, 参数)。比如把U的代码数据传进去，也就是参数 [U]，它就变成了：

|  |
| --- |
| 定义程序 U( [U] )  {  if ( H([U], [U]) == true )  {  while (true) { } // 死循环  }  else  {  return true; // 返回true，正常退出  }  } |

上面代码有两处出现了 UU，一开始是 U( [U] )，然后紧接着是 H( [U], [U] )。

H( [U], [U] ) 代表着，用自动判定程序H，判定程序 U( [U] ) 是否会停机。假如H我已经写好了，那么有两种结果：

H( [U], [U] ) 返回true，这代表判定程序认为 U( [U] )会正常停机。而如果if那里进入true的分支，则实际上 U( [U] ) 陷入了死循环。

H( [U], [U] ) == true 与 U( [U] ) 死循环 矛盾！

H( [U], [U] ) 返回false，这代表判定程序认为 U( [U] )会死循环。而如果进入else的分支，则实际上 U( [U] ) 会返回停机返回true。

H( [U], [U] ) == false 与 U( [U] ) 停机 矛盾！

矛盾是不可能矛盾的，唯一的答案就是“停机自动判定程序H”不可能存在，即便能写出来也一定有漏洞。