1.Puffy 是否仍能仅使用 .fvd 文件从其 FVD 映像中恢复数据？解释为什么或为什么不。

1. Can Puffy still recover the data from their FVD image using only the .fvd file? Explain why or why not

**Puffy 是什么系统，恢复需要用到什么：**

Puffy 是使用 FVD（Flexible Virtual Disk，灵活虚拟磁盘）驱动的系统用户。FVD 将磁盘的数据和管理信息分开存放：.fvd 保存实际数据，.fvd.ref 保存引用信息，用来标记哪些数据被分支或快照共享。当 .fvd.ref 丢失后，.fvd 中的原始数据虽然还在，但系统无法判断哪些部分仍被使用，哪些可以释放。要恢复数据，除了 .fvd 文件，还必须依靠 .fvd.ref 文件中的引用信息，让系统重新识别各个分支的内容。

1. **.fvd.ref 的作用：**

.fvd.ref 文件记录了各数据块的引用情况和分支对应关系，帮助系统在多个分支共享相同数据时追踪其使用状态，防止误删仍被引用的内容。它保证了快照和分支之间的正确关系，维持磁盘空间的合理利用。如果丢失 .fvd.ref，系统就无法判断哪些数据仍在使用，也无法保持快照间的独立性。

1. **恢复数据的流程：**

在 .fvd.ref 丢失时，Puffy 可以尝试先读取 .fvd 文件内容，再扫描结构重建分支对应关系，生成新的 .fvd.ref 文件，让系统重新挂载磁盘。但该过程复杂且结果不一定可靠，因为系统已失去原有的引用记录，只能部分恢复数据，难以保证完整一致。

**总结：**

在 FVD 系统中，.fvd 保存数据，.fvd.ref 记录引用关系，缺一不可。只有两者同时存在，系统才能判断哪些数据属于哪个分支并安全地操作。若 .fvd.ref 丢失，虽然可读出部分内容，但系统无法准确恢复结构，只能依靠重建方式恢复有限数据。

2.连续分配（在第 9 周讲座中介绍）是否适合在 FVD 文件中存储 refcount 元数据？解释为什么或为什么不。

2. Would contiguous allocation be suitable for the storage of the refcount metadata within the FVD file? Explain why or why not.

**1.访问模式：**

连续分配需要提前预留一段连续空间，而 .fvd.ref 的访问是随机的，每次只访问一个字节或一个块。连续分配在这种情况下效率会降低，因为频繁的随机访问会打破其顺序访问的优势。

**2.空间管理：**

连续分配需要提前知道文件长度，如果空间不足需要整体扩容或搬移，带来额外的开销。而 .fvd.ref 的数据量是动态变化的，很难预估长度，使用连续分配容易造成管理复杂或空间浪费。

**3.长期运行与碎片：**

随着使用时间增加，磁盘会出现碎片。链式访问的 .fvd.ref 可以灵活分配，不受碎片影响；而连续分配在空间不足或碎片化严重时可能找不到足够大的连续空间。

**4.性能：**

连续分配在顺序访问时性能更好，而 .fvd.ref 是随机访问，性能不会提升，反而会因为寻址增加而降低。

**总结：**

.fvd.ref 不适合使用连续分配，它更适合链式或按需分配的方式，能适应随机访问和动态增长的特点。

3. 使用链接分配（在第 9 周讲座中介绍）将 refcount 元数据存储在 FVD 文件中有哪些缺点？至少列出一个。

3. What are the disadvantages of using linked allocation (covered in week 9 lecture) for

storing the refcount metadata within the FVD file? List at least one

**1．随机访问效率低：**

链式分配需要沿指针逐个遍历块才能找到目标位置，而 refcount 的访问是随机的。原本一次磁盘 I/O 就能完成的读取，现在需要多次链式跳转，访问效率明显降低。

**2.空间开销大：**

每个数据块除了存放 refcount 数据，还需要额外存储一个指针信息。refcount 本身只占一个字节，但指针的大小远超过数据本身，导致额外的空间浪费。

**3.实现复杂度高：**

链式分配需要维护指针链表，处理插入和删除操作，代码实现更复杂，容易引入错误或维护困难。

4. 你会如何减小块地图的大小？列出一个与之相关的缺点。

4. How would you reduce the size of the block maps? List one disadvantage associated with it.

可以采用\*\*多级映射（multi-level mapping）\*\*的方式来减少 block map 的体积。原本每个 block map 都直接记录所有数据块的位置，而多级映射会把映射表分层，例如第一层只记录第二层表的位置，第二层才指向实际数据块。这样就能用更少的空间管理更多的数据。

**缺点：**

多级映射虽然节省了空间，但每次访问都需要经过多层查表才能找到目标块，**查找过程更复杂，访问速度会变慢**。