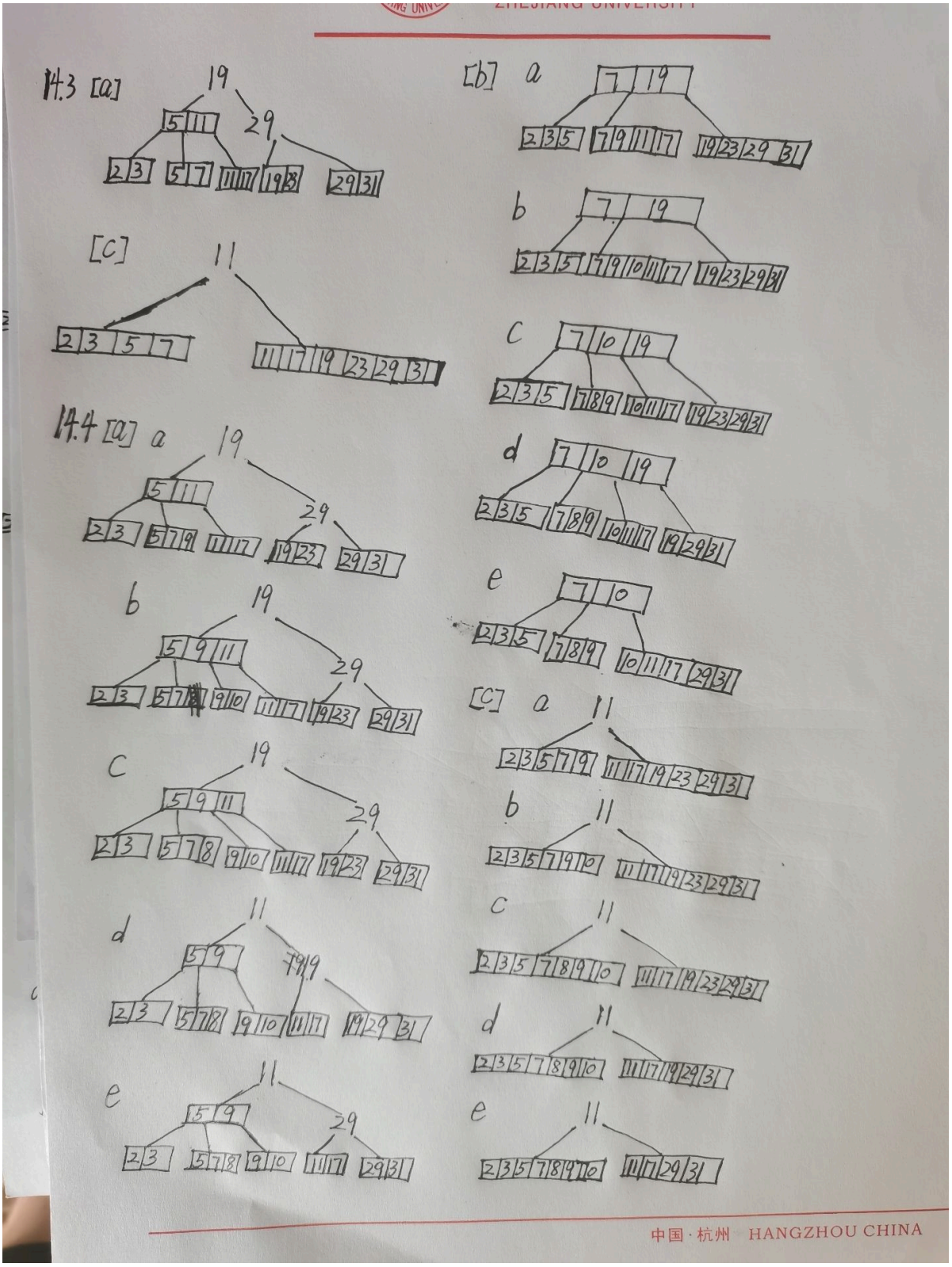
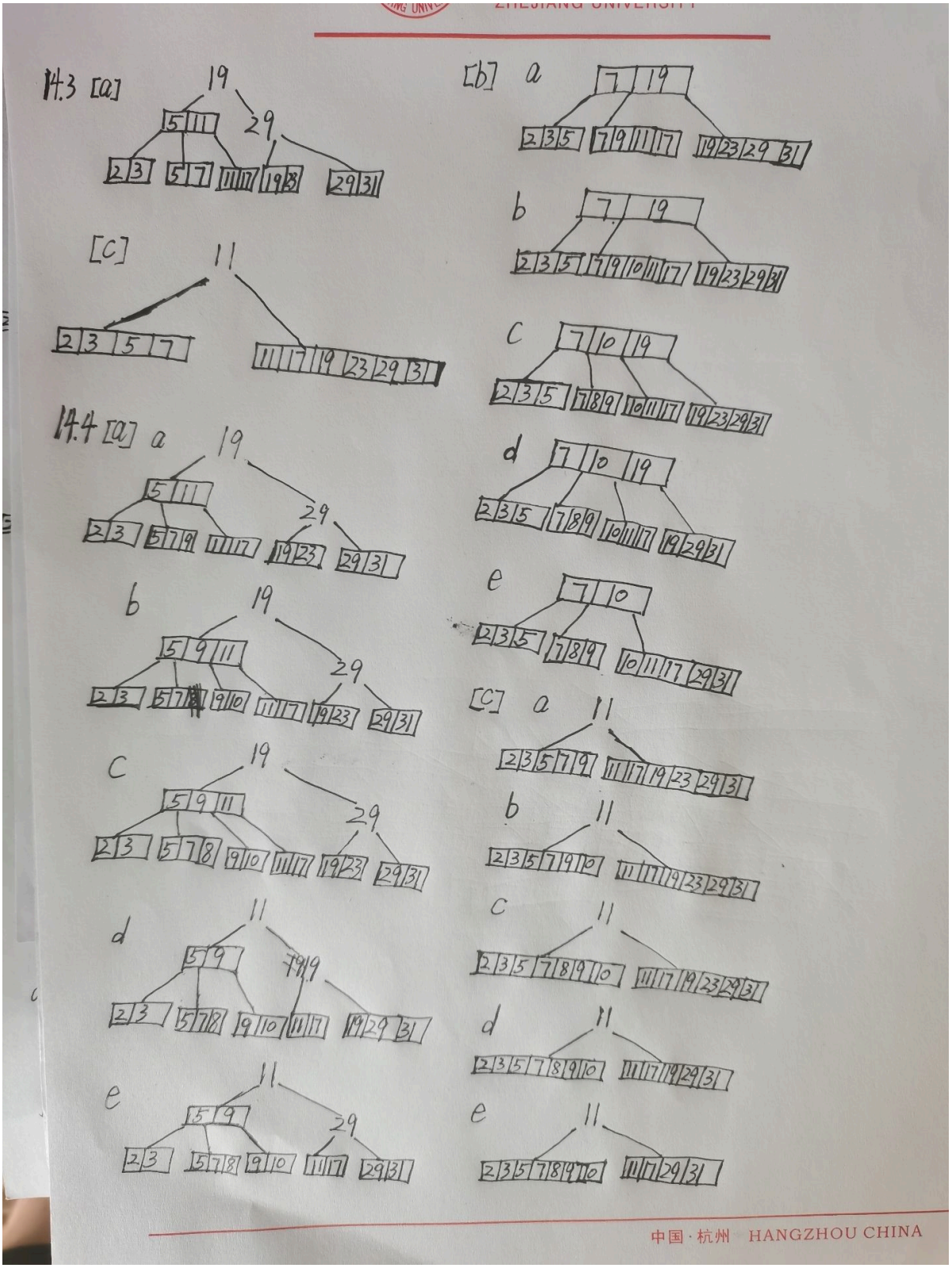
14.3

a.c.



14.4



14.11

如果在一段时间内没有更新操作，但对索引进行了大量查找，那么可以将某一层级（例如第*i*层）的条目提前合并到下一层级，即使该层级未满。这样做的好处是，读取操作无需再查找第*i*层的索引，从而降低读取成本。

24.10

优点

1. **提高写入吞吐量**
   - 每层多棵树允许并行处理写入操作（例如同时写入不同的树），减少锁竞争，提升并发写入能力。
   - 写入操作可以分散到多棵树中，避免单棵树过大导致的合并延迟。
2. **延迟合并操作的开销**
   - 合并仅在单棵树达到阈值时触发，而多棵树可以分摊合并压力。例如，Level 1 有 4 棵树时，每次只需合并其中一棵到 Level 2，而非全量合并，降低瞬时 I/O 负担。
3. **适应突发写入**
   - 短期的写入高峰可以被多棵树缓冲，避免因单棵树快速填满而频繁触发合并，从而平滑写入性能波动。

缺点

1. **读取成本增加**
   - 每层多棵树意味着读取时需要检查更多位置（例如 Level 1 的 4 棵树均需搜索），可能增加 I/O 和 CPU 开销。
2. **空间放大（Space Amplification）**
   - 多棵树可能导致相同键的多个版本存在于同一层的不同树中，暂未合并时会占用更多存储空间。
3. **合并策略复杂度提升**
   - 需设计更复杂的合并调度策略（例如优先合并哪棵树），可能引入额外的元数据管理开销。