**课后作业题**

1. **简答题：** 解释聚簇索引（Clustering Index）和非聚簇索引（Nonclustering Index）的区别，并说明它们在顺序扫描（Sequential Scan）时的性能差异。

聚簇索引改变表的物理存储顺序，数据按索引键排序，顺序扫描时性能更高；非聚簇索引不改变表的物理存储顺序，顺序扫描性能与非聚簇索引无关。聚簇索引适合固定排序查询，非聚簇索引适合复杂查询，但写操作时聚簇索引可能引发更多数据移动，维护成本更高。

1. **简答题：** 对比稠密索引（Dense Index）和稀疏索引（Sparse Index）的优缺点，主要从空间开销、维护成本和查询效率方面进行比较。

稠密索引具有高查询效率和精确定位能力，但空间占用大且维护成本高；稀疏索引空间占用小，维护成本低，但查询效率和精确查询能力较低。选择时需综合考虑空间开销、维护成本和查询性能需求。

1. **计算题：** 假设有一个静态哈希文件组织，有 5 个桶（bucket），编号为 0 到 4。哈希函数 h(x) 定义为字符串 x 的第一个字母的 ASCII 码对 5 取模。现有以下数据项（search-key values）："Apple", "Banana", "Cherry", "Date", "Fig", "Grape"。请计算每个数据项应该存入哪个桶中。如果发生桶溢出，请使用溢出链（overflow chaining）的方式处理，并画出最终的哈希文件组织结构示意图。

"Apple"：h("Apple") = ASCII('A') % 5 = 65 % 5 = 0 → 桶 0

"Banana"：h("Banana") = ASCII('B') % 5 = 66 % 5 = 1 → 桶 1

"Cherry"：h("Cherry") = ASCII('C') % 5 = 67 % 5 = 2 → 桶 2

"Date"：h("Date") = ASCII('D') % 5 = 68 % 5 = 3 → 桶 3

"Fig"：h("Fig") = ASCII('F') % 5 = 70 % 5 = 0 → 桶 0（冲突，与 "Apple" 同桶）

"Grape"：h("Grape") = ASCII('G') % 5 = 71 % 5 = 1 → 桶 1（冲突，与 "Banana" 同桶）

所以得到答案：

Bucket 0: "Apple" → "Fig"

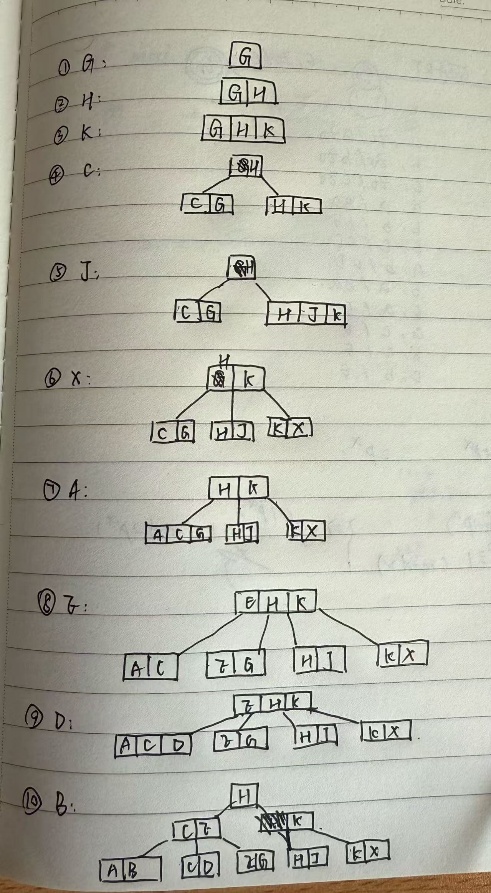
Bucket 1: "Banana" → "Grape"

Bucket 2: "Cherry"

Bucket 3: "Date"

Bucket 4: (空)

1. **计算题：** 构建一个阶为 n=4 的 B+-树。初始时树为空。请按顺序插入以下search-key values："G", "H", "K", "C", "J", "X", "A", "E", "D", "B"。请画出每一步插入操作（可能引起节点分裂）后的B+-树结构。要求非叶子节点的子节点数在 ⌈n/2⌉ 到 n 之间，叶子节点的关键字数在 ⌈(n−1)/2⌉ 到 n−1 之间，根节点至少有两个子节点（如果不是叶子节点）。



1. **计算题：**假设有一个关系（relation）有 6 条记录，编号从 0 到 5。对于某个属性，其可能取值为 'A' 和 'B'。现有该属性在每条记录上的取值如下：

* 记录 0: 'A'
* 记录 1: 'B'
* 记录 2: 'A'
* 记录 3: 'A'
* 记录 4: 'B'
* 记录 5: 'A'

请构建该属性的 Bitmap 索引。需要为每个可能的属性值（'A' 和 'B'）创建一个 Bitmap。请写出对应于值 'A' 和 'B' 的 Bitmap 串（使用 0 和 1 表示）。

Bitmap for 'A': 1 0 1 1 0 1

Bitmap for 'B': 0 1 0 0 1 0