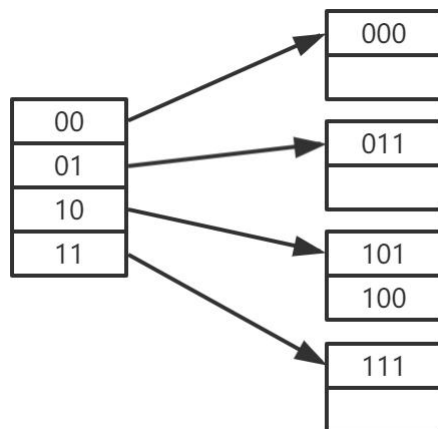


1、利用可扩展 hash 方法对以下记录进行 hash 存储：

3, 5, 7, 12, 16

设 hash 函数 $h(x) = x \bmod 8$ ，其中散列函数 $h(k)$ 是一个 b (足够大) 位二进制序列，序列的前 d 位用作索引，来区分每个元素属于哪个桶。

现要求每个桶至多包含 2 个元素，以上元素按从左往右的顺序依次添加。开始时只使用序列的前 1 位作索引（即 $d=1$ ），当桶满时进行分裂， d 相应增大。请画出添加完以上所有元素后，最终的索引结构。

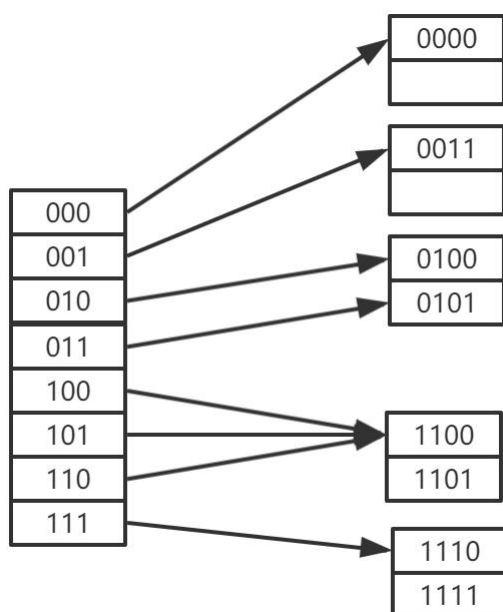


2、利用可扩展 hash 方法对以下记录进行 hash 存储：

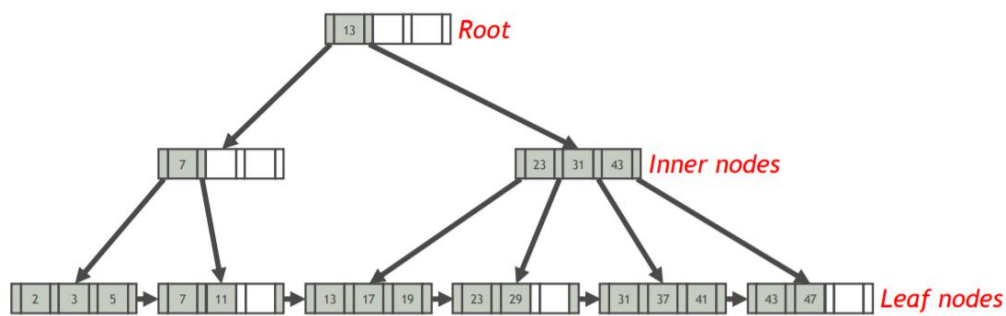
3, 16, 20, 21, 30, 44, 47, 61

设 hash 函数 $h(x) = x \bmod 16$ ，其中散列函数 $h(k)$ 是一个 b (足够大) 位二进制序列，序列的前 d 位用作索引，来区分每个元素属于哪个桶。

现要求每个桶至多包含 2 个元素，以上元素按从左往右的顺序依次添加。开始时只使用序列的前 1 位作索引（即 $d=1$ ），当桶满时进行分裂， d 相应增大。请画出添加完以上所有元素后，最终的索引结构。



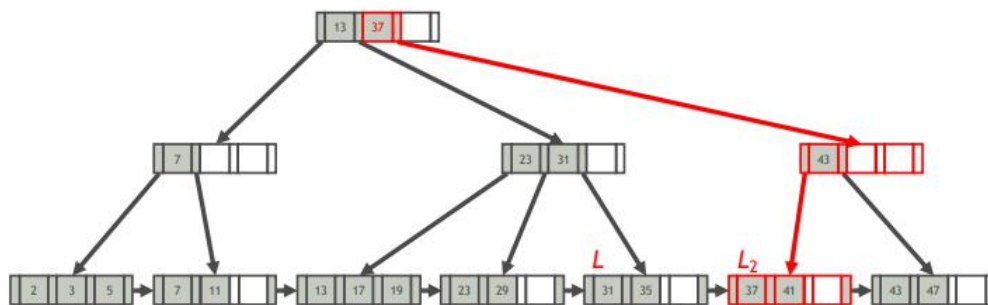
3、已知有如下 b+树：



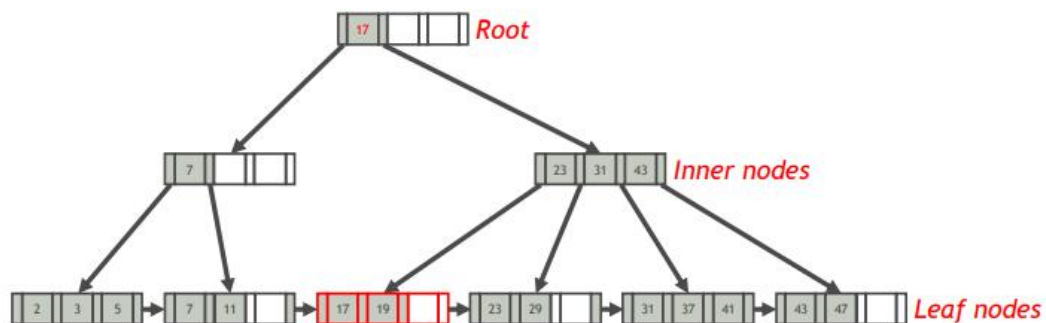
回答下列问题：

- (1) 插入键值为 35 的索引项(index entry)后，该 B+树变成什么样？请绘制出来；
- (2) 删除键值为 13 的索引项(index entry)后，该 B+树变成什么样？请绘制出来；

(1)



(2)



4.设教学管理数据库有如下 3 个关系模式：

S(S#, SNAME, AGE, SEX)

C(C#, CNAME, TEACHER)

SC(S#, C#, GRADE)

其中 S 为学生信息表、SC 为选课表、C 为课程信息表；S#、C#分别为 S、C 表的主码，(S#, C#)是 SC 表的主码，也分别是参照 S、C 表的外码

用户有一查询语句：

Select SNAME

From S, SC, C

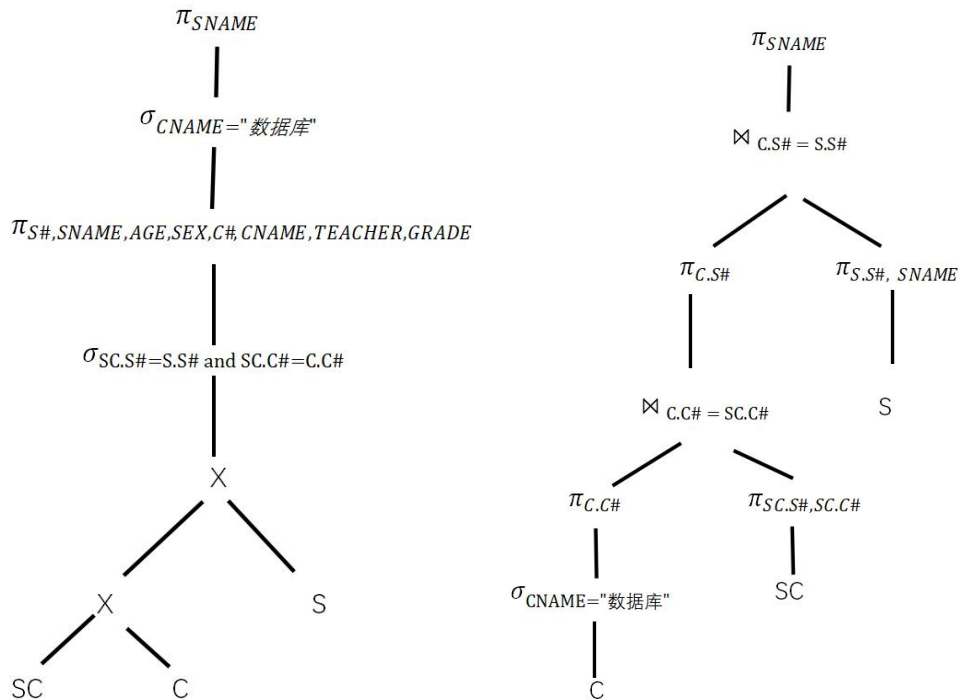
Where SC.S#=S.S# and SC.C#=C.C# and CNAME= “数据库”

检索选学“数据库”课程的学生姓名。

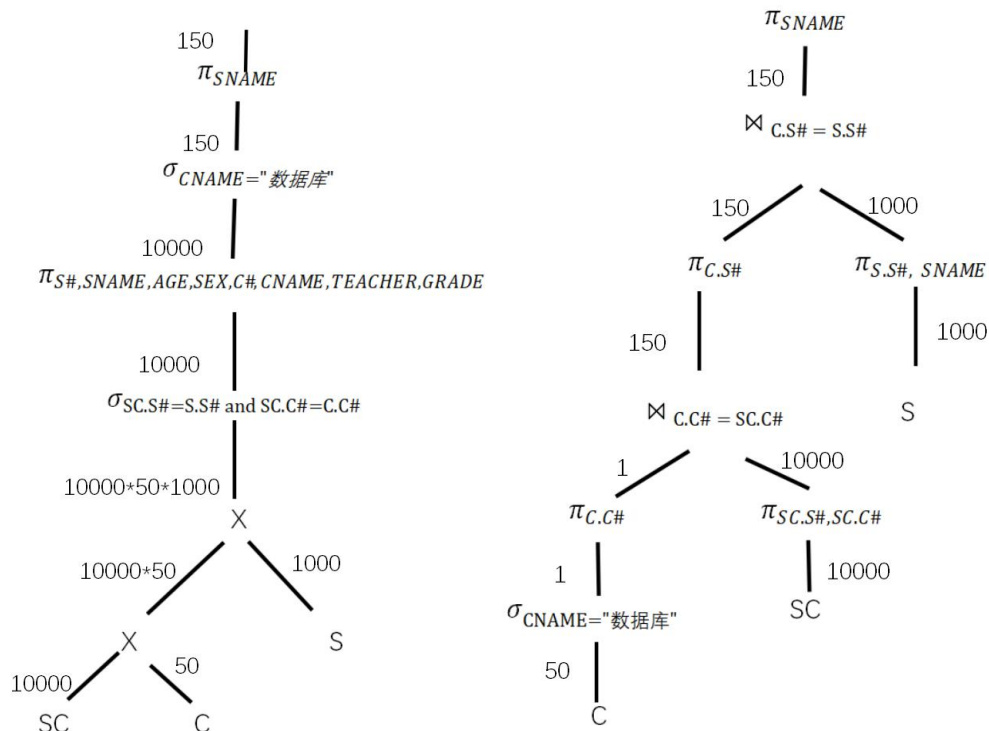
(1) 写出以上 SQL 语句所对应的关系代数表达式。

$$\pi_{SNAME}(\sigma_{CNAME=\text{数据库} \wedge SC.S\# = S.S\# \text{ and } SC.C\# = C.C\#}(SC \times C \times S))$$

(2) 画出上述关系代数表达式所对应的查询计划树。使用启发式查询优化算法，对以上查询计划树进行优化，并画出优化后的查询计划树。



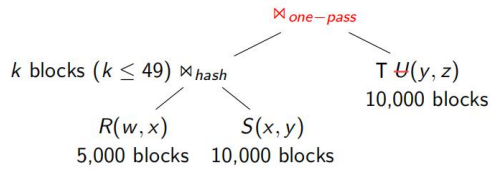
(3) 设 SC 表有 10000 条元组，C 表有 50 条元组，S 表中有 1000 条元组，SC 中满足选修数据库课程的元组数为 150，计算优化前与优化后的查询计划中每一步所产生的中间结果大小



5. 已知关系 $R(w,x), S(x,y), T(y,z)$ 的块数分别为 5000, 10000, 10000。我们准备执行关系代数查询 $(R \bowtie S) \bowtie T$ 。假设缓冲池中有 $M = 101$ 个页可用, R, S, T 上均无索引且未按连接属性排序。请回答下列问题。

- 使用什么算法执行 $R \bowtie S$ 最适合? 说明理由。
- 使用(a)中选择的算法执行 $R \bowtie S$ 的 I/O 代价是多少?
- 如果 $R \bowtie S$ 的结果不超过 49 块, 那么在使用(a)中选择的算法执行 $R \bowtie S$ 时, $R \bowtie S$ 的结果是否需要物化(materialize)到文件中? 说明理由。
- 如果 $R \bowtie S$ 的结果不超过 49 块, 那么使用什么算法将 $R \bowtie S$ 的结果与 T 进行自然连接最合适? 说明理由。
- 使用(d)中选择的算法计算连接结果的 I/O 代价是多少?
- 如果 $R \bowtie S$ 的结果大于 49 块, 那么使用什么算法将 $R \bowtie S$ 的结果与 T 进行自然连接最合适? 说明理由。
- 使用(f)中选择的算法计算连接结果的 I/O 代价是多少?

- Grace 哈希连接算法最合适, 因为 R 和 S 的块数都超过了 M , 一趟算法不可用; R 和 S 无索引, 索引连接不可用; R 和 S 未排序, 排序归并连接不可用。
- $3B(R) + 3B(S) = 45000$ 。
- 不需要。使用 Grace 哈希连接算法执行 $R \bowtie S$, R 和 S 都被分到 100 个桶中, 因此 R 的每个桶大约 50 块, S 的每个桶大约 100 块。在执行 $R_i \bowtie S_i$ 时, 可以使用一趟连接算法, 需要使用内存缓冲区 51 个页面, 还剩 50 个, 能够存放 $R \bowtie S$ 的结果。
- 如果 $R \bowtie S$ 的结果不超过 49 块, 那么在执行 $R \bowtie S$ 时, 结果可以存放在剩余可用缓冲区中, 因此使用一趟连接算法执行 $(R \bowtie S) \bowtie T$ 最合适。



使用一趟连接(one-pass join)执行 $(R \bowtie S) \bowtie U \bowtie T$

- 一趟连接使用51页内存(不计输出缓冲)

$T \bowtie U$ 的缓冲 1页

$R \bowtie S$ 的结果 k 页

输出缓冲 $100 - k$ 页

- I/O代价: $B(U) = 10000$ ($R \bowtie S$ 的结果已在内存中, 无需I/O)

- 策略如下:

- 如果 $k \leq 49$, one-pass join + pipelining
- 如果 $50 < k \leq 300$, nested-loop join + materialization
- 如果 $300 < k \leq 5000$, Grace hash join + pipelining
- 如果 $k > 5000$, Grace hash join + materialization