13.14

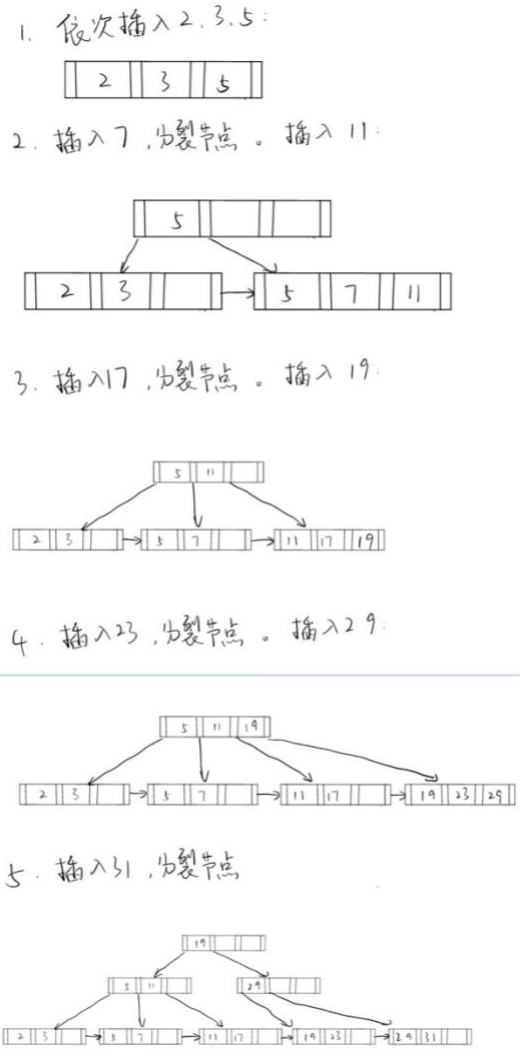
- 考虑对象引用率：引用率高的对象应尽可能保存在缓冲区，优先移出对象引用率低的对象的块
- 考虑对象读取代价：读取代价高的对象应尽可能保存在缓冲区，优先移出代价低的对象的块
- 考虑缓冲区空间：要考虑变长对象占用缓冲区可用空间，优先移出占用空间大的对象的块

例如，可以对每个缓冲区中的对象计算并持续更新得分：$score = \frac{size}{reference\_rate \times cost}$

优先移出得分高的对象的块。

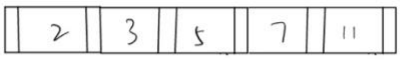
14.3

a. 4

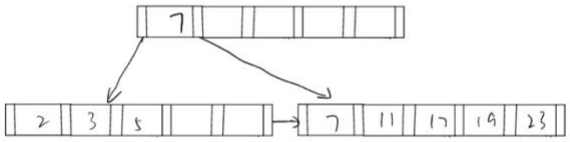


b. 6

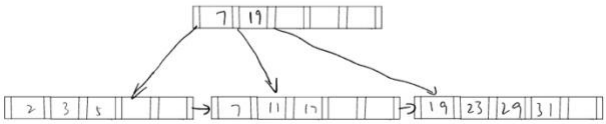
1. 依次插入 2、3、5、7、11：



2. 插入 17，分裂结点，插入 19、23：

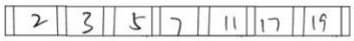


3. 插入 29，分裂结点，插入 31

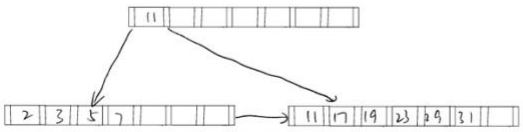


c. 8

1. 依次插入 2、3、5、7、11、17、19：



2. 插入 23，分裂节点。依次插入 29、31：



14.13

a.位图索引如下，S1~S4分别表示salary小于50000、50000到60000以下、60000到70000以下、70000及以上。

S1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
S2	0	0	0	0	0	0	0	0	0	0	0	0
S3	1	0	0	0	1	0	0	1	0	0	0	0
S4	0	1	0	1	0	1	1	0	1	1	1	1

b.在a的基础上修改S4为70000到80000以下，构建S5为80000及以上（仅列出有更改的两行）：

S4	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**
S5	0	1	0	1	0	1	0	0	1	0	1	1

在dept_name上构建位图索引：

Comp. Sci	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
Finance	0	1	0	0	0	0	0	0	1	0	0	0
Music	0	0	1	0	0	0	0	0	0	0	0	0
Physics	0	0	0	1	0	1	0	0	0	0	0	0
History	0	0	0	0	1	0	0	1	0	0	0	0
Biology	0	0	0	0	0	0	0	0	0	1	0	0
Elec. Eng.	0	0	0	0	0	0	0	0	0	0	0	1

取S5和Finance的交集：

S4	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**
Finance	0	1	0	0	0	0	0	0	1	0	0	0
$S4 \cap Finance$	0	1	0	0	0	0	0	0	1	0	0	0

检查instructor关系，利用$S4 \cap Finance$行得知金融系中工资为80000或更高的教师为Wu和Singh。

17.15

a.检查两个事务串行执行可能的结果。

- $T_{13} \rightarrow T_{14}$

1. 初始$A = B = 0$
2. $T_{13} : read(A) \Rightarrow A = B = 0$
3. $T_{13} : read(B) \Rightarrow A = B = 0$
4. $T_{13} : if\ A = 0\ then\ B := B + 1 \Rightarrow$ 满足条件，B会被增加
5. $T_{13} : write(B) \Rightarrow A = 0, B = 1$
6. $T_{14} : read(B) \Rightarrow A = 0, B = 1$
7. $T_{14} : read(A) \Rightarrow A = 0, B = 1$
8. $T_{14} : if\ B = 0\ then\ A := A + 1 \Rightarrow$ 不满足条件，A不会被增加
9. $T_{14} : write(A) \Rightarrow A = 0, B = 1$
10. 满足一致性统计$A = 0 \vee B = 0$

- $T_{14} \rightarrow T_{13}$

1. 初始$A = B = 0$
2. $T_{14} : read(B) \Rightarrow A = B = 0$
3. $T_{14} : read(A) \Rightarrow A = B = 0$
4. $T_{14} : if\ B = 0\ then\ A := A + 1 \Rightarrow$ 满足条件，A会被增加
5. $T_{14} : write(A) \Rightarrow A = 1, B = 0$
6. $T_{13} : read(A) \Rightarrow A = 1, B = 0$
7. $T_{13} : read(B) \Rightarrow A = 1, B = 0$
8. $T_{13} : if\ A = 0\ then\ B := B + 1 \Rightarrow$ 不满足条件，B不会被增加
9. $T_{13} : write(B) \Rightarrow A = 1, B = 0$
10. 满足一致性统计$A = 0 \vee B = 0$

综上所述，两个事务的每一个串行执行都保持了数据库一致性。

b. 下面是一个可能的并发执行过程：

- 初始$A = B = 0$
- $T_{13}: read(A) \Rightarrow A = B = 0$
- $T_{13}: read(B) \Rightarrow A = B = 0$
- $T_{14}: read(B) \Rightarrow A = B = 0$
- $T_{14}: read(A) \Rightarrow A = B = 0$
- $T_{13}: if\ A = 0\ then\ B := B + 1 \Rightarrow$ 满足条件，B会被增加
- $T_{14}: if\ B = 0\ then\ A := A + 1 \Rightarrow$ 满足条件，A会被增加
- $T_{13}: write(B) \Rightarrow A = 0, B = 1$
- $T_{14}: write(A) \Rightarrow A = 1, B = 1$

> 最终结果$A = 1, B = 1$，由a知两个事务的串行执行只可能产生满足$A = 0 \vee B = 0$的结果，因此这个并发执行产生了不可串行化的结果。
>
> 上面的调度之间存在数据依赖关系，都读取了对方可能修改的变量并根据其值执行不同操作，会导致冲突，因此这个调度不可串行化。

c. 不存在产生可串行化调度的T_{13}和T_{14}的并发执行。

因为T_{14}的第一条指令读的结果依赖于T_{13}最后一条指令写的结果，反之亦然。