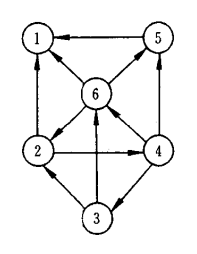
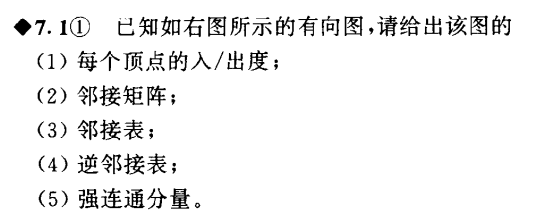
**7.1**



（1） OD(1)=0 ID(1)=3 OD(2)=2 ID(2)=2

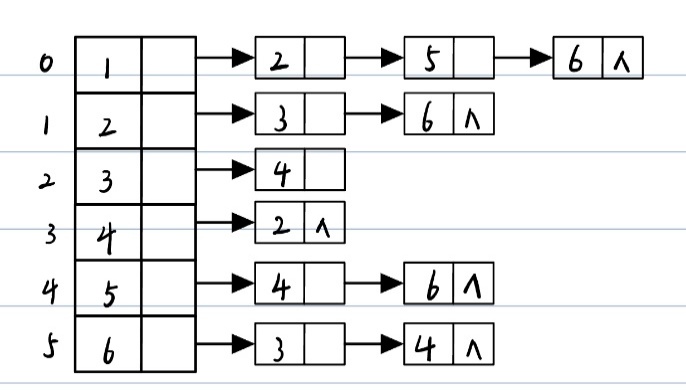
OD(3)=2 ID(3)=1 OD(4)=3 ID(4)=1

OD(5)=1 ID(5)=2 OD(6)=3 ID(6)=2

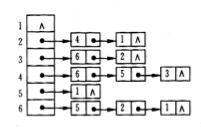
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 1 | 0 |

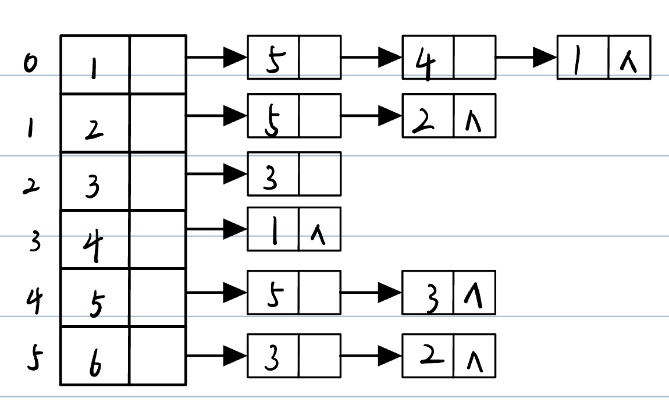
（2） 终点

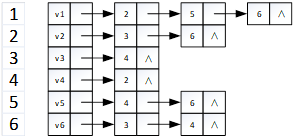
起点

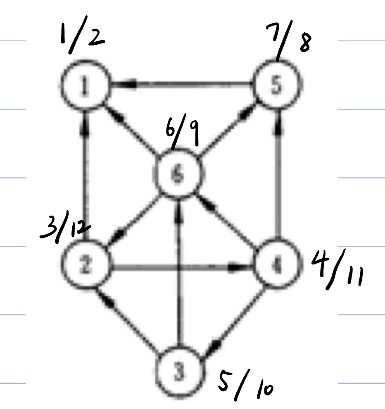


（3）正邻接表



（4）逆邻接表

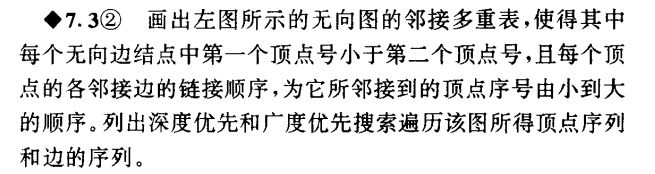
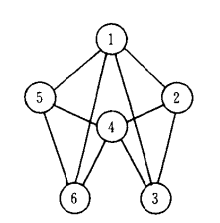


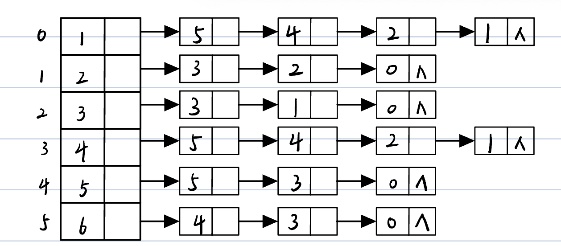
（5） ①DFS遍历G，如图，按访问顺序生成逆后序：**243651**

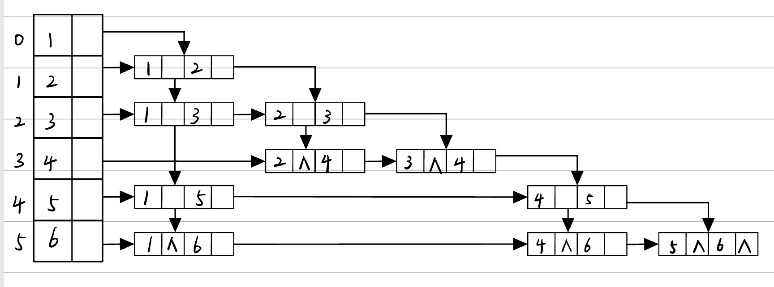
（标注数对为：发现顺序/访问顺序）

②按逆后序出发对逆向图G’进行DFS得到以下强 连通分量：**1、5、2346**

**7.3**

邻接表：

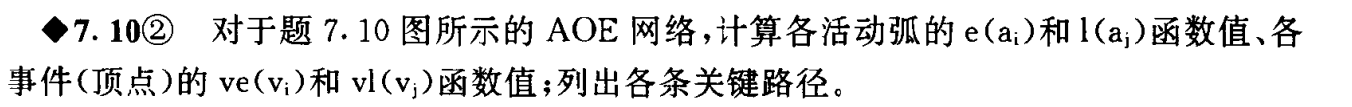


邻接多重表：

深度优先遍历 顶点序列：~~156432 边序列：15、56、64、43、32~~  123456

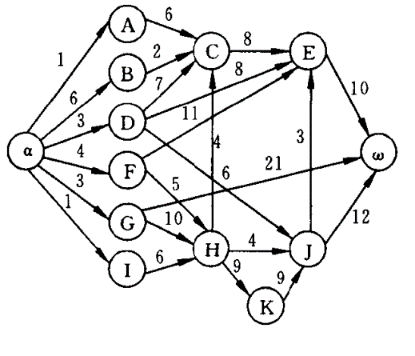
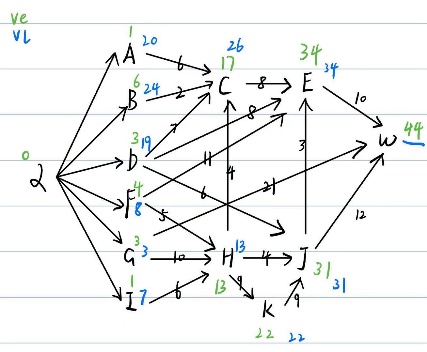
广度优先遍历 顶点序列：~~156324 边序列：15、16、13、12、54~~  123564

**7.10**



对顶点进行拓扑排序得到顶点序列：

α>A>B>D>F>G>I>H>C>K>J>E>ω



按拓扑顺序，依次计算每个事件的最早发生时间，再按拓扑排序的逆序，依次计算每个事件的最晚发生时间

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **顶点v** | **α** | **A** | **B** | **D** | **F** | **G** | **I** | **H** | **C** | **K** | **J** | **E** | **ω** |
| **ve(v)** | **0** | **1** | **6** | **3** | **4** | **3** | **1** | **13** | **17** | **22** | **31** | **34** | **44** |
| **vl(v)** | **0** | **20** | **24** | **19** | **8** | **3** | **7** | **13** | **26** | **22** | **31** | **34** | **44** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **v** | **A** | **B** | **D** | **F** | **G** | **I** |
| **e(<α, v>)** | **0** | **0** | **0** | **0** | **0** | **0** |
| **l(<α, v>)** | **19** | **18** | **16** | **4** | **0** | **6** |

a=<A, C>: e(a)=1, l(a)=26-6=20 a=<B, C>: e(a)=6, l(a)=26-2=24

a=<D, C>: e(a)=3, l(a)=26-7=19 a=<D, E>: e(a)=3, l(a)=34-8=26

a=<D, J>: e(a)=3, l(a)=31-6=25

a=<F, E>: e(a)=4, l(a)=34-11=23 a=<F, H>: e(a)=4, l(a)=34-5=29

a=<G,ω>: e(a)=43, l(a)=44-21=23 a=<G, H>: e(a)=4, l(a)=34-5=29

a=<I, H>: e(a)=1, l(a)=13-6=7

a=<H, J>: e(a)=13, l(a)=31-4=27 a=<H, K>: e(a)=13, l(a)=31-9=22

a=<C, E>: e(a)=17, l(a)=34-8=26

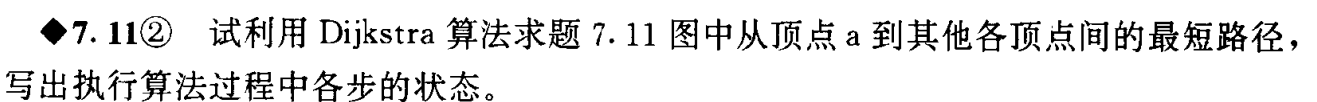
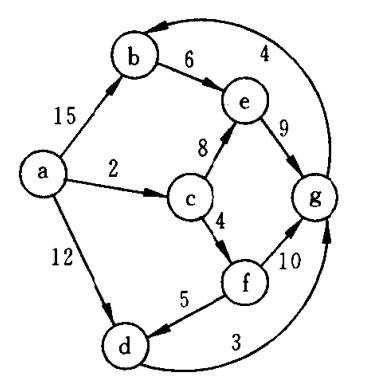
a=<K, J>: e(a)=22, l(a)=31-9=22

a=<J,ω>: e(a)=43, l(a)=44-12=32

a=<E,ω>: e(a)=43, l(a)=44-10=34

因此关键路径为：α-G-H-K-J-ω

**7.11**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 0 | 0 | 0 | 0 | 0 | 0 |
| dist | 15 | 2 | 12 | ∞ | ∞ | ∞ |

初始状态如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 0 | 1 | 0 | 0 | 0 | 0 |
| dist | 15 | 2 | 12 | 10 | 6 | ∞ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 0 | 1 | 0 | 0 | 1 | 0 |
| dist | 15 | 2 | 11 | 10 | 6 | 16 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 0 | 1 | 0 | 1 | 1 | 0 |
| dist | 15 | 2 | 11 | 10 | 6 | 16 |

⇒

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 0 | 1 | 1 | 1 | 1 | 0 |
| dist | 15 | 2 | 11 | 10 | 6 | 14 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 0 | 1 | 1 | 1 | 1 | 1 |
| dist | 15 | 2 | 11 | 10 | 6 | 14 |

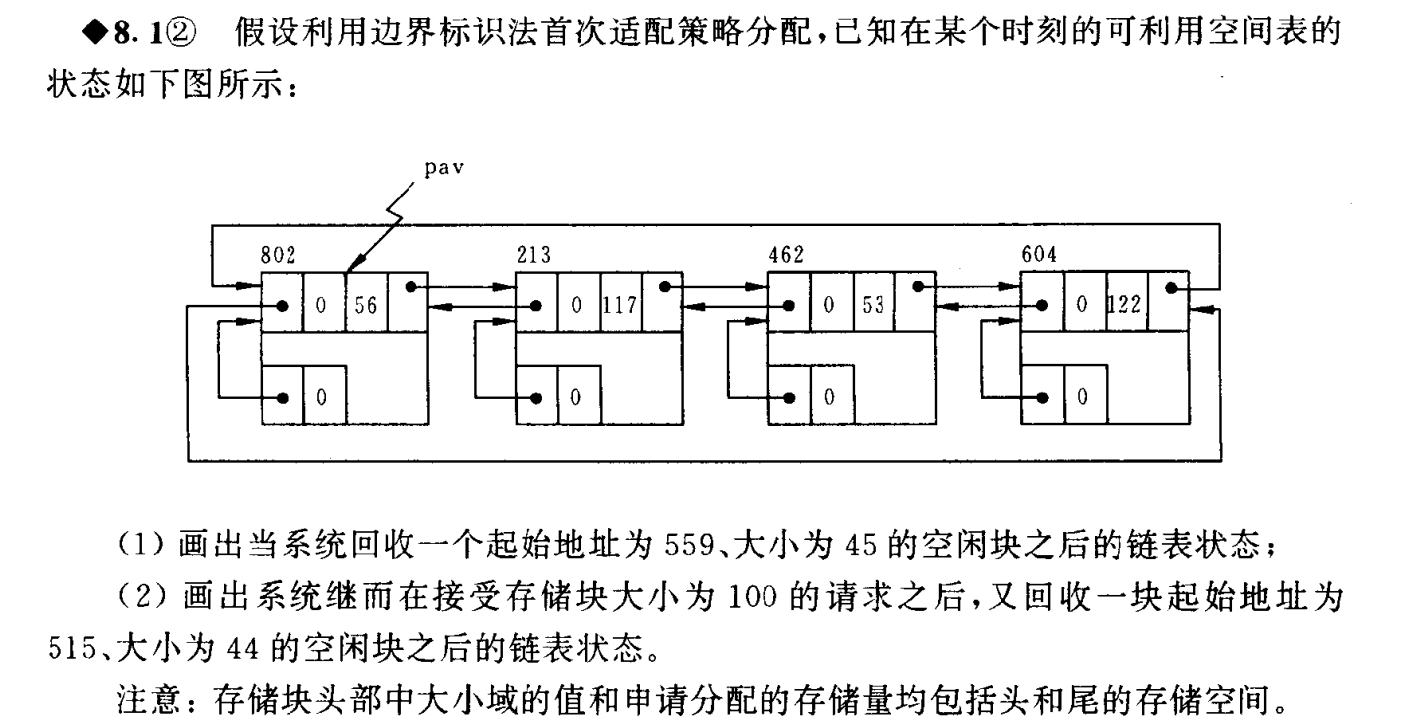
⇒

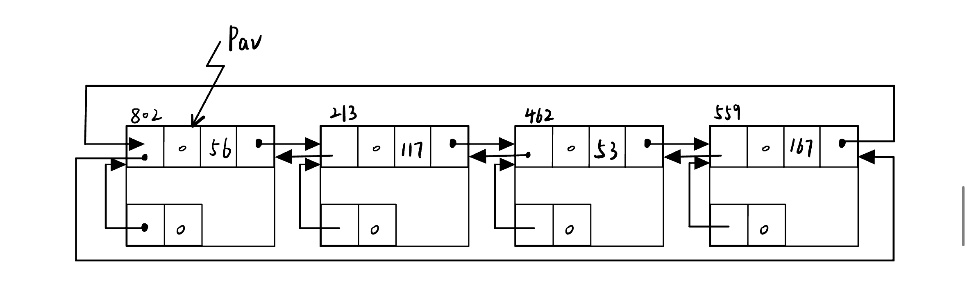
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | b | c | d | e | f | g |
| final | 1 | 1 | 1 | 1 | 1 | 1 |
| dist | 15 | 2 | 11 | 10 | 6 | 14 |

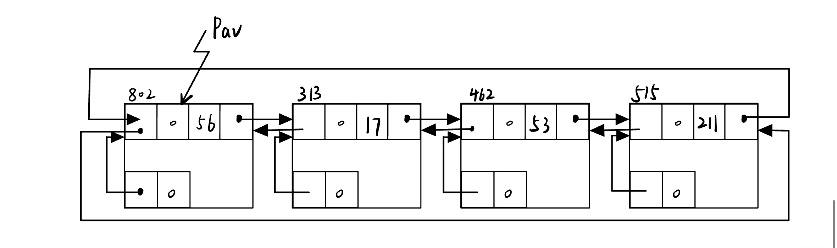
故a到其他结点的最短路径如下

|  |  |
| --- | --- |
| 终点 | 路径 |
| b | a-b |
| c | a-c |
| d | a-c-f-d |
| e | a-c-e |
| f | a-c-f |
| g | a-c-f-g |

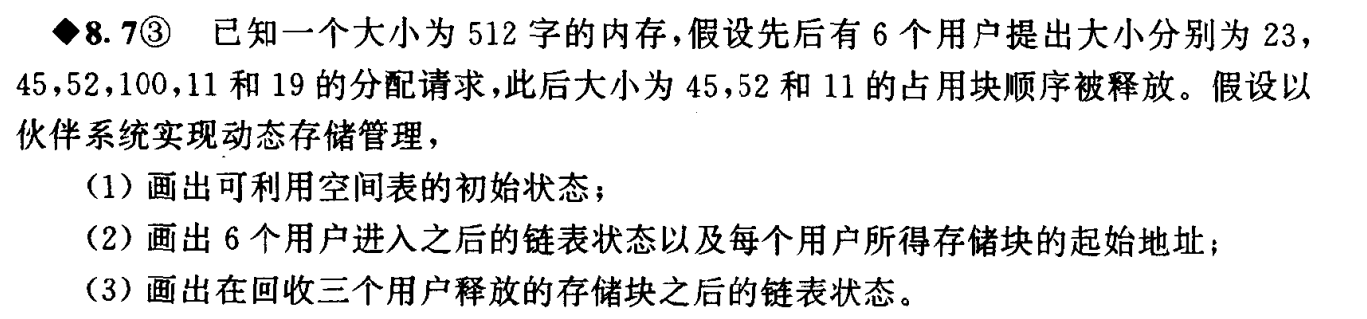
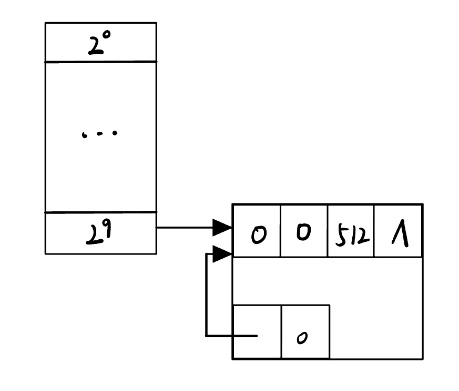
**8.1**



（1）

（2）

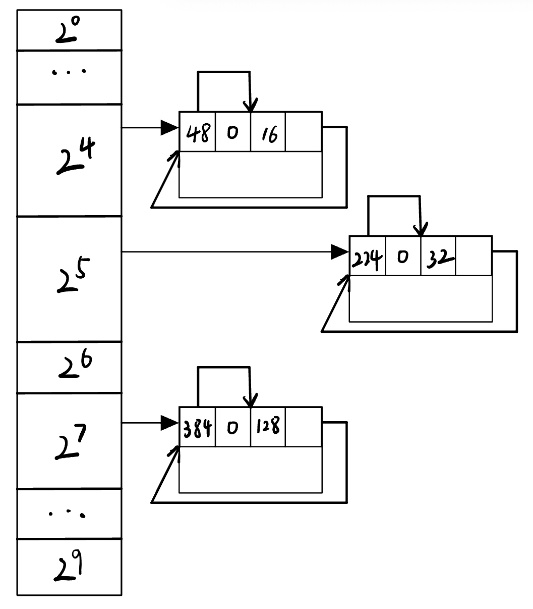
**8.7**



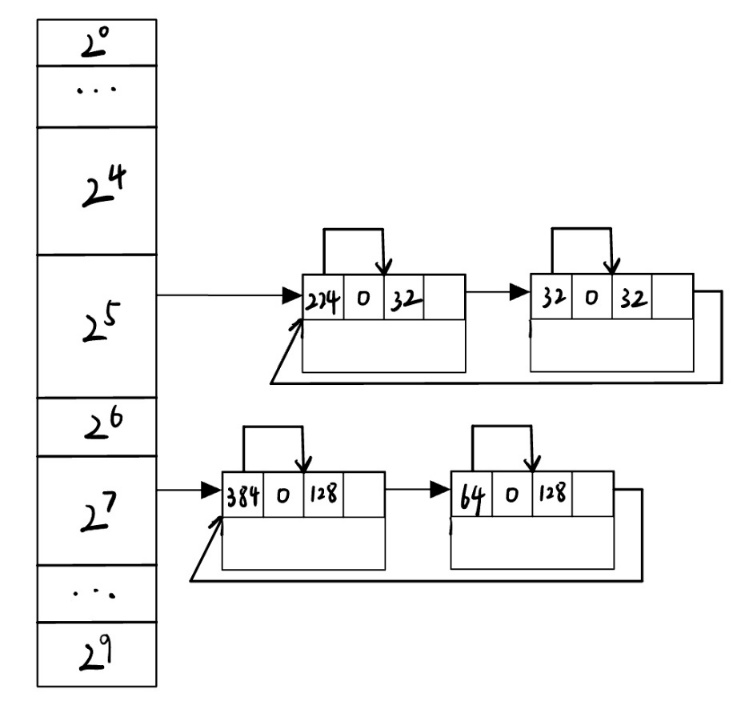
（1）

（2）

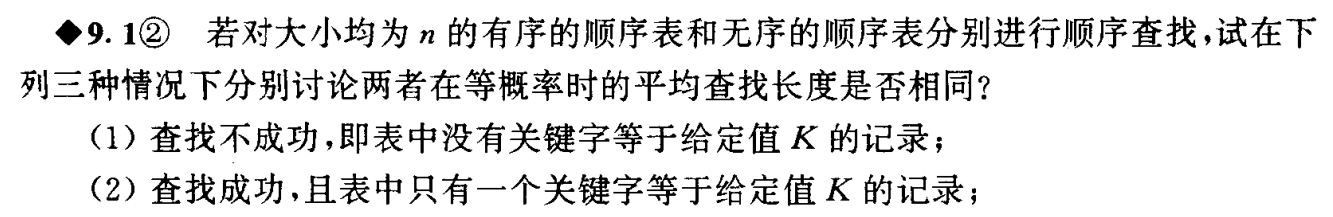
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用户序号 | 1 | 2 | 3 | 4 | 5 | 6 |
| 申请量 | 23 | 45 | 52 | 100 | 11 | 19 |
| 起始地址 | 0 | 64 | 128 | 256 | 32 | 192 |
| 分配量 | 32 | 64 | 64 | 128 |  |  |

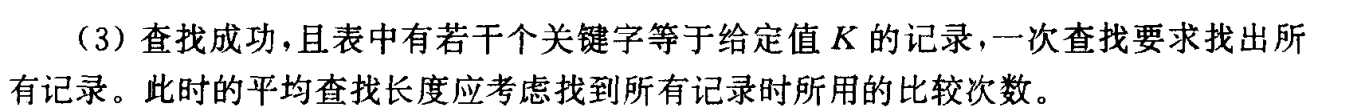


（3）



**9.1**





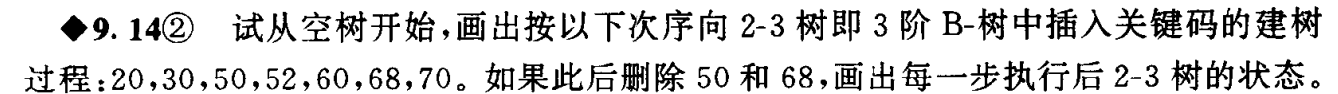
（1）不同。假设有序表按升序排列，则搜索过程中遇到比K大的值则说明没有K的记录，可以停止搜索；而无序表需要搜完n个元素后才能确定K的记录不存在；所以在查找不成功的情况下，有序表的平均查找长度小于无序表的平均查找长度。

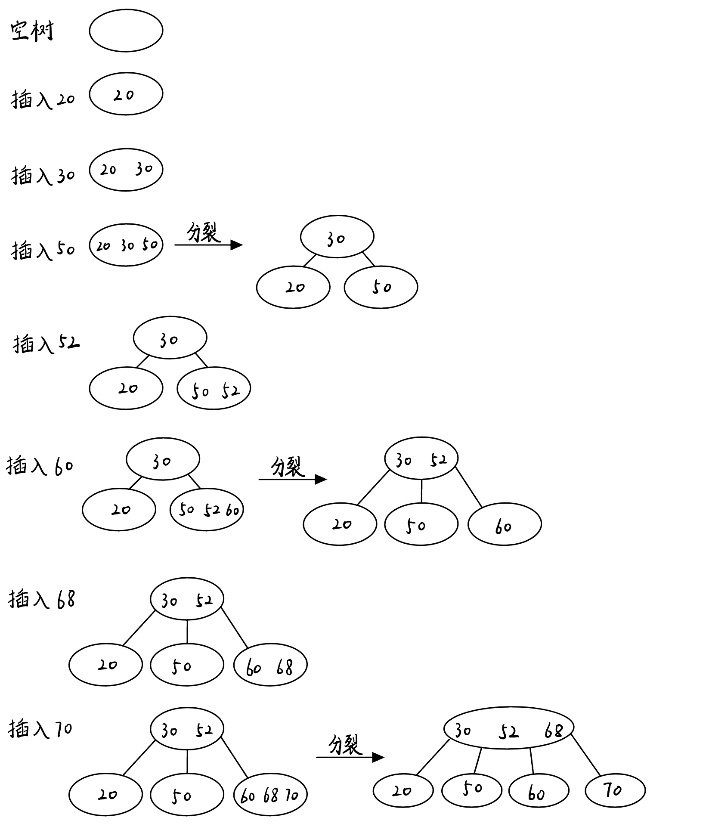
|  |  |  |  |
| --- | --- | --- | --- |
|  | 有序表 | 无序表 | 是否相同 |
| ASL | (1+2+…+n)/n = (n+1)/2 | (1+2+…+n)/n = (n+1)/2 | 相同 |

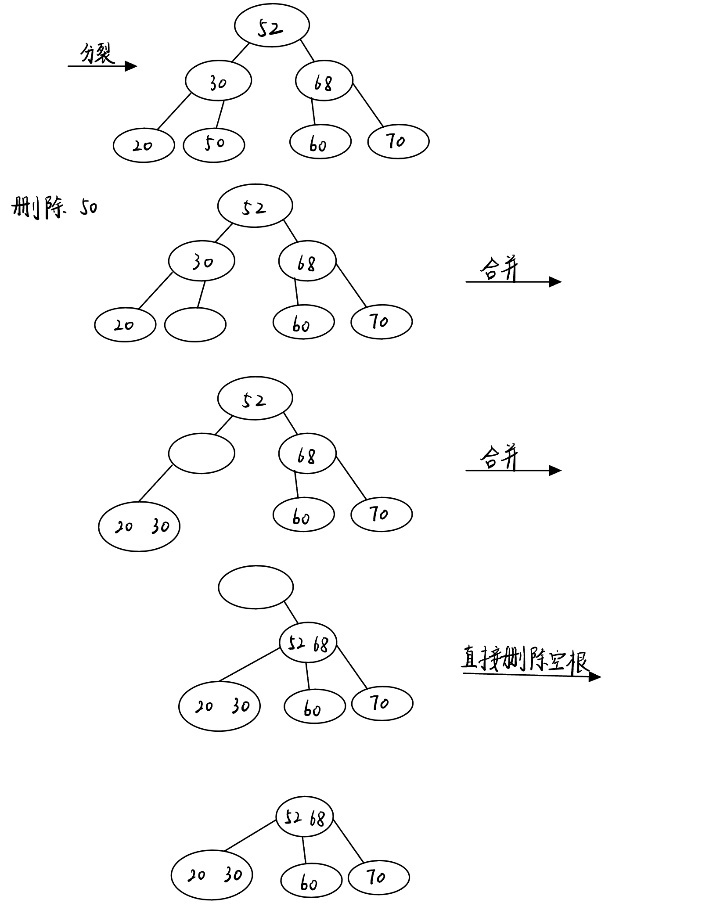
（2）

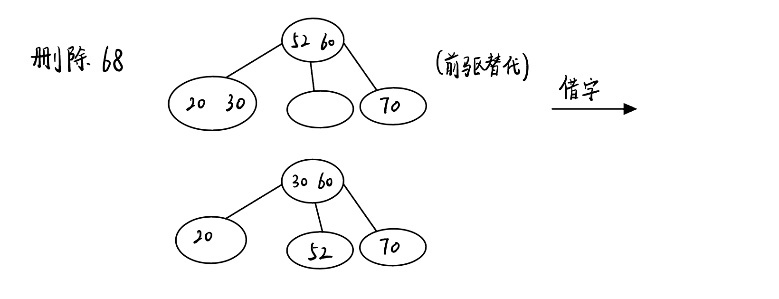
（3）不同，K在表中有多个记录时且查找成功的情况下，有序表只需要搜索到比K大的值时即可以确定找出了所有等于定值K的记录，平均查找长度<n。而无序表只有搜索完整个表后才能确定找出了所有等于定值K的记录，平均查找长度=n。故这种情况下有序表的平均查找长度小于无序表的平均查找长度。

**9.14**

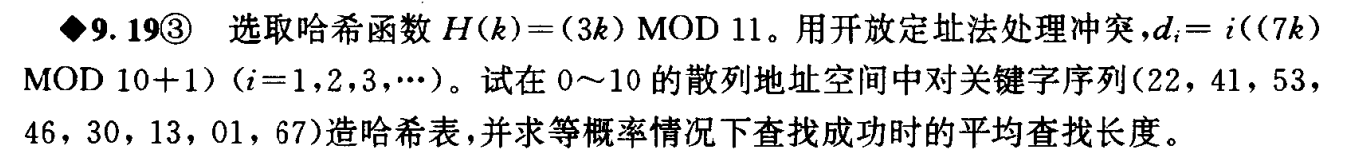








**9.19**



H0(22)=66 MOD11 = 0

H0(41)=123 MOD11 = 2

H0(53)=159 MOD11 = 5

H0(46)=138 MOD11 = 6

H0(30)=90 MOD11 = 2（冲突）

d1= 1\*(210 MOD10 +1 )=1 H1(30)=2+d1 = 3

H(13)=39 MOD11 = 6（冲突）

d1= 1\*(91 MOD10 +1 )=2 H2(13)=6+d1 = 8

H(01)=03 MOD11 = 3（冲突）

d1= 1\*(07 MOD10 +1 )=8 H1(01)=(6+d1)MOD11 = 3 （冲突）

d2=2\*(07 MOD10 +1 )=16 H2(01)=(6+d2)MOD11 = 0 （冲突）

d3=3\*(07 MOD10 +1 )=24 H3(01)=(6+d3)MOD11 = 8 （冲突）

d4=4\*(07 MOD10 +1 )=32 H4(01)=(6+d4)MOD11 = 5 （冲突）

d5=5\*(07 MOD10 +1 )=40 H5(01)=(6+d5)MOD11 = 2 （冲突）

d6=6\*(07 MOD10 +1 )=48 H6(01)=(6+d6)MOD11 = 10

H(67)=201 MOD11 = 3（冲突）

d1= 1\*(67\*7 MOD10 +1)=10 H1(67)=(3+d1)MOD11 = 2 （冲突）

d2= 2\*(67\*7 MOD10 +1)=20 H2(67)=(3+d2)MOD11 = 1

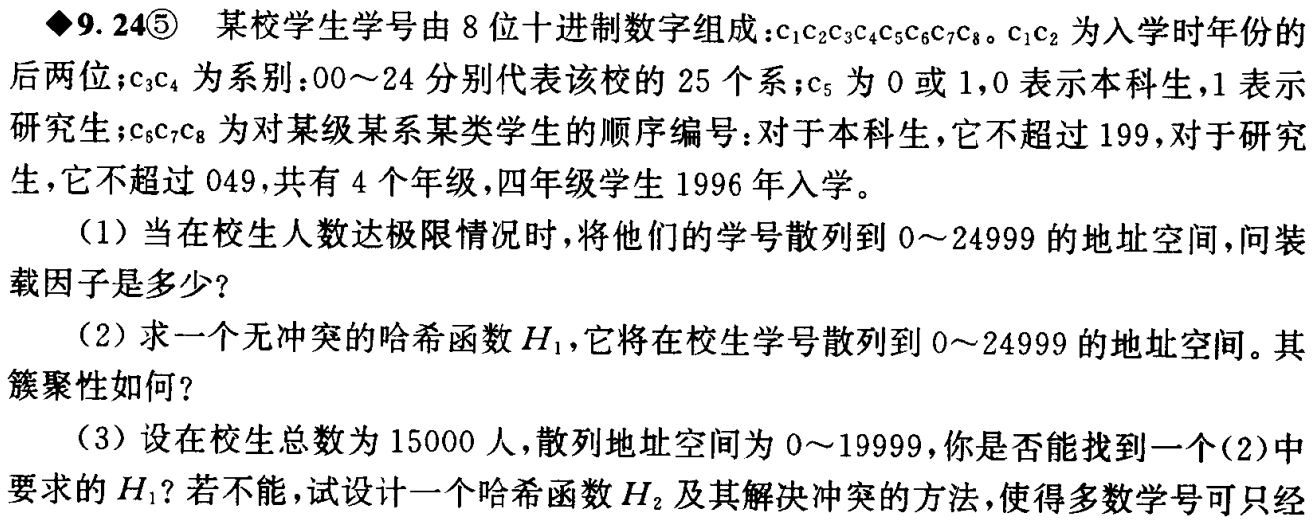
综上，最终哈希表为：

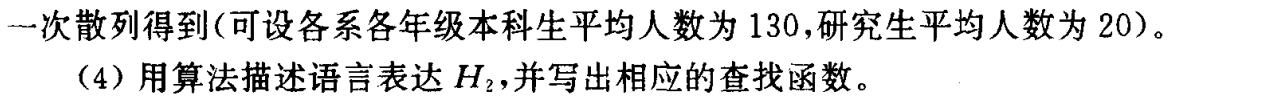
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 关键字 | 22 | 67 | 41 | 30 |  | 53 | 46 |  | 13 |  | 01 |
| 查找次数 | 1 | 3 | 1 | 2 |  | 1 | 1 |  | 2 |  | 6 |

（按记录的查找次数计算）ASL=(1+1+1+1+2+2+6+3)/8=17/8=2.125

（按公式计算）装填因子α=8/11，线性探测法成功时ASL≈0.5\*(1+1/(1-α))=7/3=2.333

**9.24**





（1）极限人数=4\*25\*(199+1+049+1)=25000

装载因子α=25000/25000=1

（2）H1(c1c2c3c4c5c6c7c8)=1000\*c3c4 + 250\*c1c2 + 200\*c5 + c6c7c8

簇聚性分析：同一系、同年级、同类别的学生地址连续，簇聚性较高。

(3)不能找到无冲突的哈希函数。

按系、年级分配地址块，每块有20000/(25\*4)=200个地址，

本科生用前200\*130/150≈174（进一法），研究生用175-199

H2(c1c2c3c4c5c6c7c8)=(c1c2×5000+c3c4×200+c5×175+c6c7c8) mod20,000

该哈希函数按照各系各年级的本科生与研究生人数之比为双方都预留了空间，尽可能地避免冲突，多数学号可以一次散列成功。

实际过程中若遇到冲突，可使用线性探测法：Hi(key)=(i×200+H2(key)) mod20,000 , i=1,2,3,…, 24进行跨地址块散列。

（4）哈希函数描述：

ADDRESS H2(key){

c1c2=key/1000000, c3c4=(key/10000)%100;

c5=(key/1000)%10, c6c7c8=key%1000;

return (c1c2×5000+c3c4×200+c5×175+c6c7c8) %20,000

}

查找算法描述：（设哈希表无记录的位置为NONE）

ADDRESS HashSearch(key, HList){

address = H2(key);

int i = 0；// 已进行的探测次数

while(HList[address] != key){

if(HList[address] == NONE || i>=24) return NONE; // 最多探测24次

i++;

address = (address + 200) % 20000；

}

return address；

}