

实做题

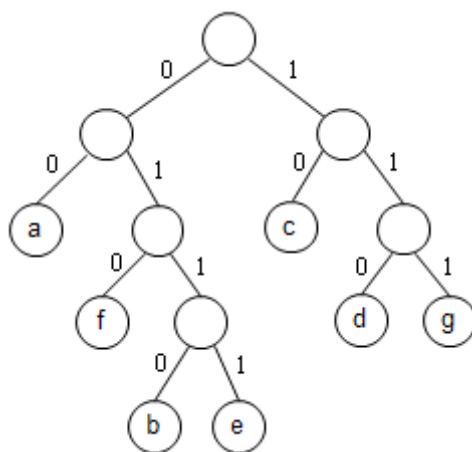
1、已知 a、b、c、d、e、f、g 的哈夫曼编码分别为：00，0110，10，110，0111，010，111

(1) 画出此哈夫曼树；

(2) a、b、c、d、e、f、g 出现的概率分别为：0.30，0.07，0.20，0.09，0.08，0.15，0.11，求加权路径长度 WPL。

【参考答案】

(1) 哈夫曼树如下：



(2) $WPL = 2 * (0.30 + 0.20) + 3 * (0.09 + 0.15 + 0.11) + 4 * (0.07 + 0.08) = 2.65$

【他山之石】

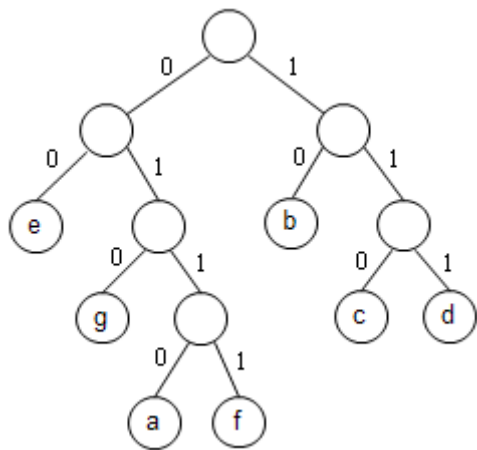
已知 a、b、c、d、e、f、g 的哈夫曼编码分别为：0110，10，110，111，00，0111，010

(1) 画出此哈夫曼树；

(2) a、b、c、d、e、f、g 出现的概率分别为：0.03，0.35，0.13，0.15，0.20，0.05，0.09，求带权路径长度。

【参考答案】

(1) 哈夫曼树如下：



(2) $WPL=2*(0.35+0.20)+3*(0.13+0.15+0.09)+4*(0.03+0.05)=2.53$

2、给出如下关键字序列：29，18，25，47，58，12，51，10

- (1) 给出快速排序每一趟的结果
- (2) 按照所给关键字序列建立平衡二叉树

【参考答案】

(1) 快速排序：

第一趟：（10 18 25 12）29（58 51 47）

第二趟：10（18 25 12）29（47 51）58

第三趟：10（12）18（25）29 47（51）58

第四趟：10 12 18 25 29 47 51 58

(2)

插入元素	所生成的 AVL 树
29	
18	
25	

47	<pre> graph TD 25((25)) --> 18((18)) 25 --> 29((29)) 29 --> 47((47)) </pre>
58	<pre> graph LR subgraph Left 25L((25)) --> 18L((18)) 25L --> 29L((29)) 29L --> 47L((47)) 47L --> 58L((58)) end subgraph Right 25R((25)) --> 18R((18)) 25R --> 47R((47)) 47R --> 29R((29)) 47R --> 58R((58)) end Left --> Right </pre>
12	<pre> graph TD 25((25)) --> 18((18)) 25 --> 47((47)) 18 --> 12((12)) 47 --> 29((29)) 47 --> 58((58)) </pre>
51	<pre> graph TD 25((25)) --> 18((18)) 25 --> 47((47)) 18 --> 12((12)) 47 --> 29((29)) 47 --> 58((58)) 29 --> 51((51)) </pre>
10	<pre> graph LR subgraph Left 25L((25)) --> 18L((18)) 25L --> 47L((47)) 18L --> 12L((12)) 12L --> 10L((10)) 47L --> 29L((29)) 29L --> 51L((51)) 29L --> 58L((58)) end subgraph Right 25R((25)) --> 12R((12)) 25R --> 47R((47)) 12R --> 10R((10)) 12R --> 18R((18)) 47R --> 29R((29)) 29R --> 51R((51)) 29R --> 58R((58)) end Left --> Right </pre>

【他山之石】

给出如下关键字序列：12，11，13，49，26，14，8，7

(1) 给出快速排序每一趟的结果

(2) 建立 AVL 树

【参考答案】

(1) 快速排序：


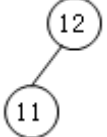
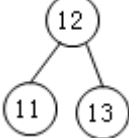
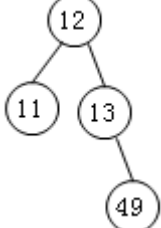
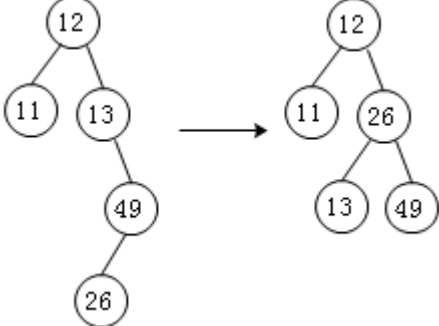
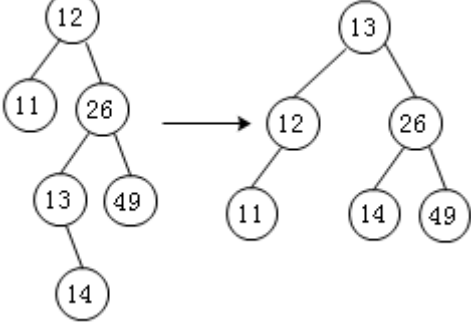
第一趟：（7 11 8）12（26 14 49 13）

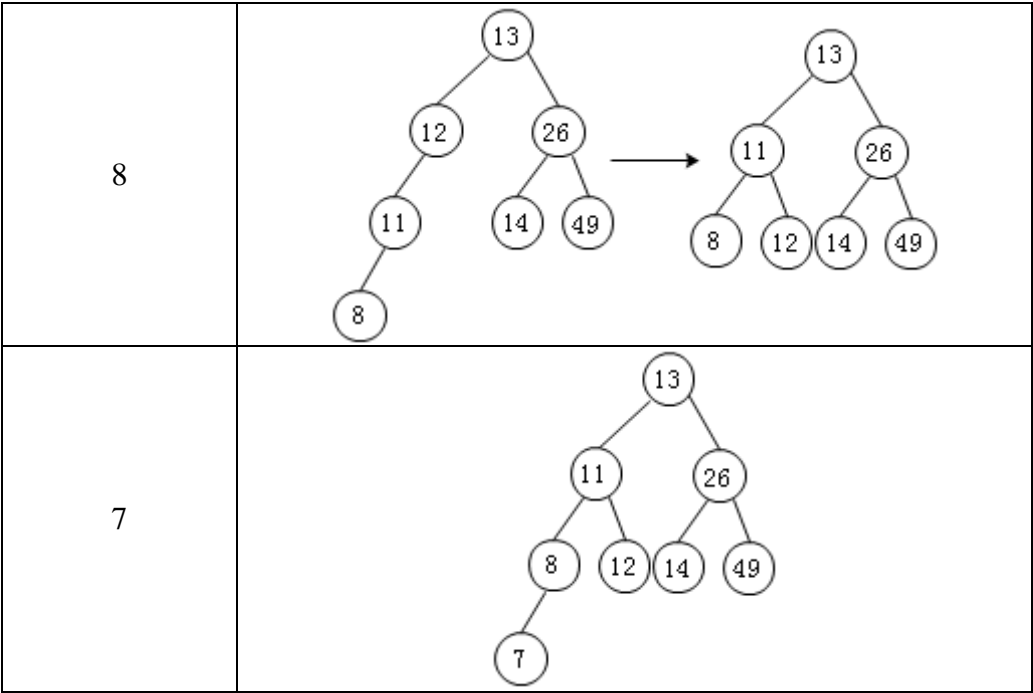
第二趟：7（11 8）12（13 14）26（49）

第三趟：7（8）11 12 13（14）26 49

第四趟：7 8 11 12 13 14 26 49

(2)

插入元素	所生成的 AVL 树
12	
11	
13	
49	
26	
14	



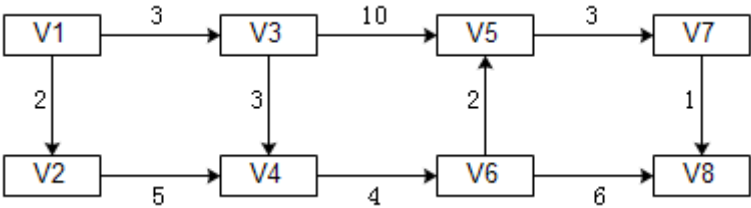
3、设有一组关键字 {9, 1, 23, 14, 55, 20, 84, 27}，采用哈希函数 $H(\text{key}) = \text{key} \bmod 7$ ，表长 $m=10$ ，采用线性探测再散列的方法解决冲突，对该关键字构造哈希表。

【参考答案】

散列地址	0	1	2	3	4	5	6	7	8	9
关键字	14	1	9	23	84		55	20	27	

key	$H(\text{key}) = \text{key} \bmod 7$	处理
9	2	无冲突，散列地址为 2
1	1	无冲突，散列地址为 1
23	2	冲突后移，散列地址为 3
14	0	无冲突，散列地址为 0
55	6	无冲突，散列地址为 6
20	7	冲突后移，散列地址为 7
84	0	冲突后移，散列地址为 4
27	6	冲突后移，散列地址为 8

4、

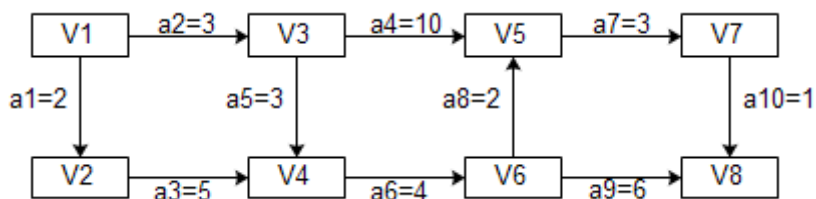


- (1) 给出关键路径
- (2) 求 V1 到其他顶点的最短路径。

【参考答案】

(1)

解：如图设置活动：



由题，用 $ve(k)$ 表示事件 k 的最早发生时间； $vl(k)$ ：事件 k 的最迟发生时间；

$e(ak)$ ：活动 ak 的最早发生时间； $l(ak)$ ：活动 ak 的最迟发生时间。

① $ve(k)$ 求法： $ve(k) = \max\{ve(j) + \langle j, k \rangle\}$ ，其中 j 为 k 的可列个前驱。

$vl(k)$ 求法： $vl(k) = \min\{vl(j) - \langle k, j \rangle\}$ ，其中 j 为 k 的可列个后继。

可得下表：

k	1	2	3	4	5	6	7	8
$ve(k)$	0	2	3	7	13	11	16	17
$vl(k)$	0	2	3	7	13	11	16	17

② $e(ak)$ ：活动 ak 的最早发生时间：与由此节点发出的活动的最早发生时间；

$l(ak)$ ：活动 ak 的最迟发生时间：减去以其为结束的活动的活动持续时间。

可得下表：

ak	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10
$e(ak)$	0	0	2	3	3	7	13	11	11	16
$l(ak)$	0	0	2	3	4	7	13	11	11	16
差值	0	0	0	0	1	0	0	0	0	0

因为关键路径的最早发生时间与最迟发生时间相等，即差值为 0。

故本图的关键路径为：

V1→V2→V4→V6→V8 或者

V1→V3→V5→V7→V8 或者

V1→V2→V4→V6→V5→V7→V8

(2)

迭代	Round1	Round2	Round3	Round4	Round5	Round6	Round7
V2	2 v1->v2						
V3	3 v1->v3	3 v1->v3					
V4	∞	7 v1->v2 ->v4	6 v1->v3 ->v4				
V5	∞	∞	13 v1->v3 ->v5	13 v1->v3 ->v5	12 v1->v3 ->v4->v6 ->v5		
V6	∞	∞	∞	10 v1->v3 ->v4->v6			
V7	∞	∞	∞	∞	∞	15 v1->v3 ->v4->v6 ->v5->v7	
V8	∞	∞	∞	∞	16 v1->v3 ->v4->v6 ->v8	16 v1->v3 ->v4->v6 ->v8	16 v1->v3 ->v4->v6 ->v8
顶点集 S	v1,v2	v1,v2,v3	v1,v2,v3,v4	v1,v2,v3,v4,v6	v1,v2,v3,v4,v5,v6	v1,v2,v3,v4,v5,v6,v7	v1,v2,v3,v4,v5,v6,v7,v8

5、已知一关键码序列为：72，87，61，23，04，16，05，58，根据堆排序原理

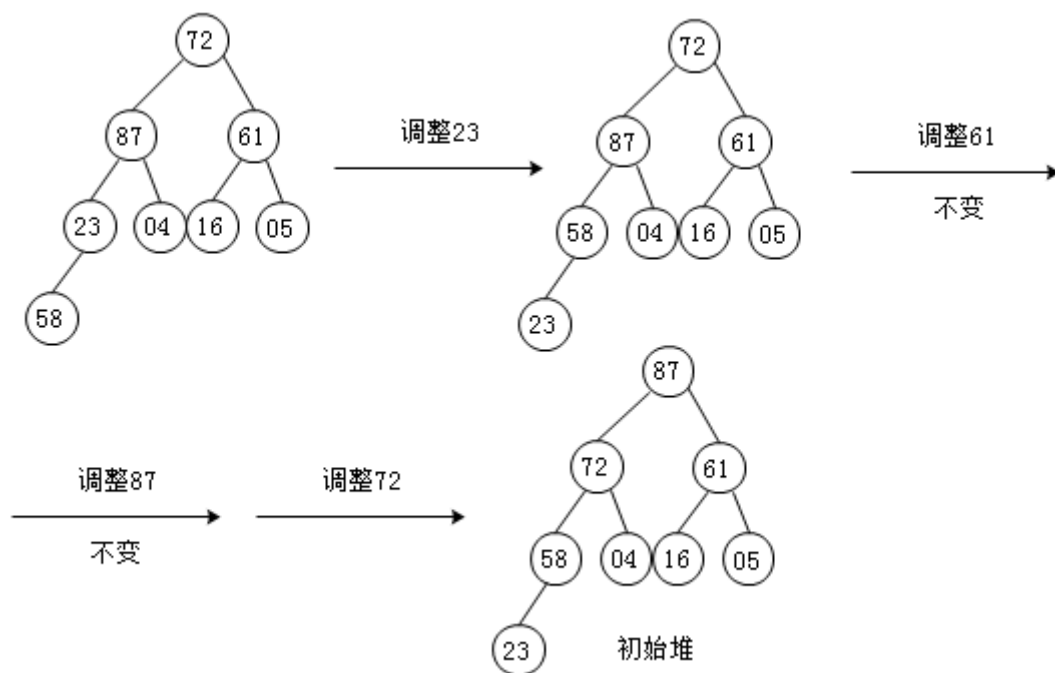
(1) 用图表示初始堆建立的过程

(2) 根据初始堆，得到排序后的前 3 个数，用图表示过程。

【参考答案】

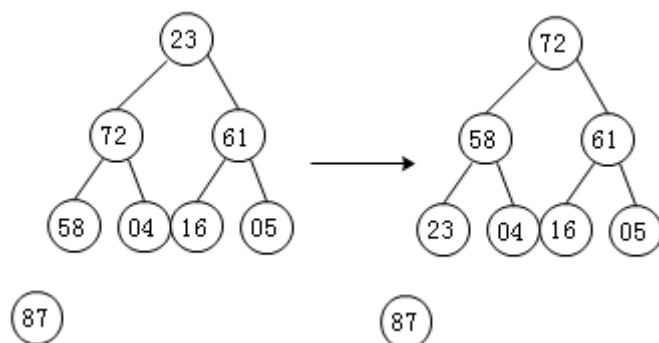
【注】在题目没有声明堆为大顶堆还是小顶堆的时候，可自行选择一个，我们以大顶堆为例作参考答案。

(1) 初始堆建立的过程：

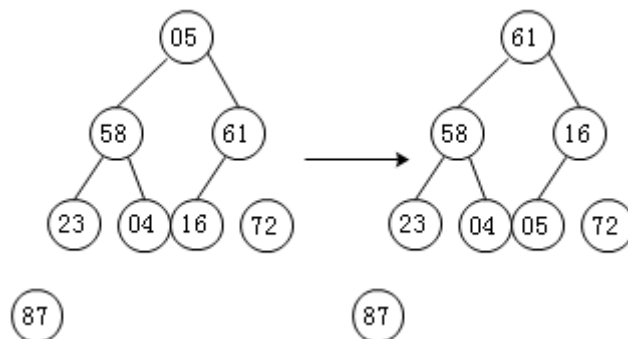


(2)

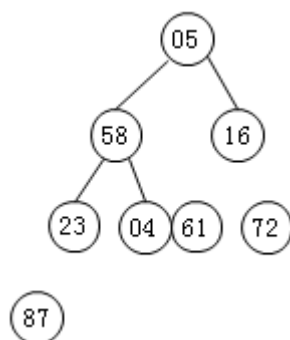
①由初始堆将 87 与 23 互换得第一个数 87，调整 23 如图所示：



②由①将 72 与 05 互换得第二个数 72，调整 05 如图所示：



③由②将 61 与 05 互换得第三个数 61，如图所示：



算法题

栈模拟队列，算法篇有提到。

编程题

1、已知未知数 x 和 y ， $x+y=A$ 、 $x-y=B$ ，给出 A 、 B 的值，解方程求 x 、 y 的值。

输入：输入样例量第一行给出的是测试的组数 T ，每组包含整数 A 和 B

$(-10000 \leq A, B \leq 100000)$

输出：对于每组输入、输出 x 和 y 以空格隔开，测试数据保证 x 和 y 都是整数。

输入样例：

2

5 1

48 22

输出样例：

3 2

35 13

【参考答案】

```
#include<iostream.h>
```

```
#define maxSize 10000
```

```
int main()
```

```
{
```

```
    int i,n;//n 为测试的组数
```

```
    cin>>n;

    int data[maxSize][2];

    for(i=0;i<n;i++)

        cin>>data[i][0]>>data[i][1];

    for(i=0;i<n;i++)

        cout<<(data[i][0]+data[i][1])/2<<" "

        <<(data[i][0]-data[i][1])/2<<endl;

    return 0;

}
```

2、判定子串

输入两个串 s 和 t ，判断 s 是否是 t 的子串（如果从 t 中删除任意字符可以得到 s ，则称 s 是 t 的子串）。

输入：输入样例包括若干组测试，每组包含两个以上空格隔开的字符串 s 和 t ，输入 EOF 终止。

输出：对于每组输入，如果 s 是 t 的子串，则输出 “Yes”，否则输出 “No”。

输入样例：

2

sequence subsequence

person compression

输出样例：

Yes

No

【参考答案】

```
#include<iostream>
```

```
#include<string>
```

```
#include<algorithm>
```

```
using namespace std;
```

```
bool IsSubstring(string s,string t)
```

```
{  
    char *it;  
    int i;  
    int x=s.length();  
    it=t.begin();  
    for(i=0;i<x;i++)  
    {  
        it=(find(it,t.end(),s[i]));  
        if(it==t.end())  
            return false;  
    }  
    return true;  
}
```

```
int main()
```

```
{  
    string s,t;  
    while (cin>>s>>t)  
    {  
        if (IsSubstring(s,t))  
            cout<<"Yes"<<endl;  
        else  
            cout<<"No"<<endl;  
    }  
}
```

```
    return 0;  
}
```