1.答案如下：

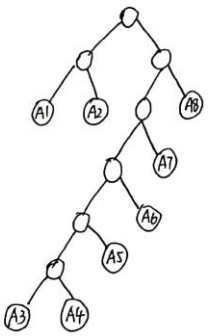
1. 假设矩阵A1～A8的规模如下：

A1	A2	A3	A4	A5	A6	A7	A8
10×100	100×5	5×50	50×30	30×20	20×60	60×45	45×50

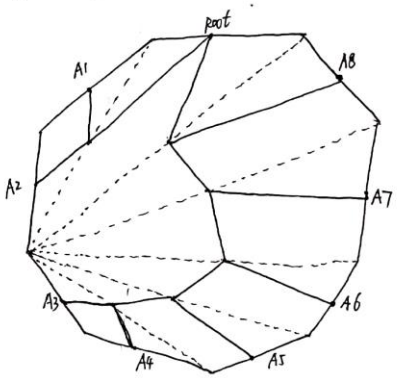
(1) 最优完全加括号方式为：

((A1×A2)(((((A3×A4)A5)A6)A7)A8))

(2) 该问题语法树为：



最优三角剖分图为：



2.答案如下：

（1）最优二叉树搜索树的动态规划算法：

给定关键字序列$\{k_i,\dots,k_j\}(1\le i\le j\le n)$，定义$E[i,j]$为期望搜索代价，$w[i,j]$为期望搜索代价的增量。

假设k_r是包含序列的一棵最优子树的根，则根k_r的左子树包含$k_i,\dots,k_{r-1}$(以及$d_i,\dots,d_{r-1}$)，右子树包含$k_{r+1},\dots,k_j$(以及$d_r,\dots,d_j$)

当$j=i-1$时，$E[i,j]=q_{i-1}$

当$i\le j$时，从关键字序列中选择一个节点作为根k_r，分别构造左子树和右子树，递归查找最优的构造方式，$E[i,j]$和$w[i,j]$的递归表达式如下：

$$w[i,j]=w[i,j-1]+p_j+q_j$$

$$E[i,j]=w[i,j]+\min\{E[i,r-1]+E[r+1,j]\}$$

使用$R[i,j]$保留以k_r为根的最优子树$T[i,j]$的根节点。

（2）最优值概率表格如下：

0.05	0.45	0.90	1.25	1.75	2.75
0	0.10	0.40	0.70	1.20	2.0
0	0	0.05	0.25	0.60	1.30
0	0	0	0.05	0.30	0.90
0	0	0	0	0.05	0.05
0	0	0	0	0	0.10

子树概率表格如下：

0.05	0.30	0.45	0.55	0.70	1.00
0	0.10	0.25	0.35	0.50	0.80
0	0	0.05	0.15	0.30	0.60
0	0	0	0.05	0.20	0.50
0	0	0	0	0.05	0.35
0	0	0	0	0	0.10

根节点标识表格：

1	1	2	2	2
0	2	2	2	4
0	0	3	4	5
0	0	0	4	5
0	0	0	0	5

3.代码如下：

```
public class zuoye2_3 {
    public static void main(String[] args){
        activity act=new activity();
        act.s=new int[]{0,2,2,3,4,6,7,9,10,13};
        act.f=new int[]{0,3,4,5,7,8,11,12,15,17};
        act.a=new int[act.s.length];
        act.count=1;

        act.select(act.s, act.f, act.a, act.count);
    }
}
class activity{
    int[] s;   //活动开始
    int[] f;   //活动结束
```

```
    int[] a;   //活动是否被选中,表示被选中，0表示未被选中
    int count; //记录活动安排时间
    //选择活动
    public void select(int[] s,int[] f,int[] a,int count){
        timeSort(s, f, 0, f.length-1);

        a[1]=1;
        int j=1;
        for(int i=2;i<s.length;i++){
            if(s[i]>=f[j]){
                a[i]=1;
                j=i;
                count++;
            }
            else{
                a[i]=0;
            }
        }
        System.out.println("总共有活动"+count+"个可以安排");
        showSelect(s, f, a);
    }
    //找到主元位置
    public int partitionWithFirst(int[] s,int[] f,int low,int
high){
        int k=f[low];
        int m=s[low];
        int i=low;
        int j=high;
        while(j>i){
            //--j反向查找小于k的元素
            while(j>i&&f[j]>k){
                j--;
            }
            //找到小于k值之后进行替换
            if(i<j){
                f[i]=f[j];
                s[i]=s[j];
                i++;
            }


            //++i方向查找大于k的值进行替换
            while(j>i&&f[i]<k){
                i++;
```

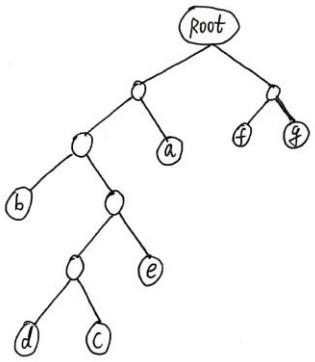
```
            }
            if(i<j){
                f[j]=f[i];
                s[j]=s[i];
                j--;
            }
        }
        f[i]=k;
        s[i]=m;
        return i;
    }

    //对结束时间进行排序
    public void timeSort(int[] s,int[] f,int low,int high){
        if(low<high){
            int k=partitionWithFirst(s, f, low, high);
            timeSort(s, f, low, k-1);
            timeSort(s, f, k+1, high);
        }

    }
    //输出结果
    public void showSelect(int[] s,int[] f,int[] a){
        for(int i=1;i<s.length;i++){
            if(a[i]==1){
                System.out.println("活动开始时间: "+s[i]+"\t"+"活动结束时
间: "+f[i]);
            }
        }
    }
}
```

输出结果如下：

```
总共有活动5个可以安排：
活动1    开始时间：2      结束时间：3
活动2    开始时间：3      结束时间：5
活动3    开始时间：6      结束时间：8
活动4    开始时间：9      结束时间：12
活动5    开始时间：13     结束时间：17
```

4.答案如下：

4. 哈夫曼树为：



各字编码为：

a	b	c	d	e	f	g
01	000	00101	00100	0011	10	11

平均码长：

$$L = 2\times31\% + 3\times12\% + 5\times5\% + 5\times2\% + 4\times10\% + 2\times18\% + 2\times22\%$$

$$= 2\times71\% + 3\times12\% + 4\times10\% + 5\times7\%$$

$$= 2.53$$