学院 专业 级 学号 姓名 密 封 线

5. 在下列 5 阶 B-树中首先插入关键字 85，然后删除关键字 70，画出插入元素和删除元素后的 B-树。

山东大学 2018–2019 学年 一 学期 数据结构 课程试卷 A

题号	一	二	三	四	五	六	七	八	九	十	总分	阅卷人
得分												

得分	阅卷人

一、线性结构（30 分）。

1、已知线性表：(8, 9, 2, 13, 0, 7, 1, 6, 5)，请完成以下题目。

(1) 请描述公式化描述及链表描述的空间需求。如果需要删除元素 13，请描述各自的时间复杂度。

(2) 请分别进行选择排序、插入排序、快速排序（以 8 为轴），并给出第一轮排序结束后各自的结果。

(3) 设计散列表，散列函数为 H (k)=k%7，散列表长度为 11，请给出线性开型寻址的散列表。

(4) 基于以上散列表，查找元素 1，给出需要的查找次数。

(5) 若使用单链表存储上述线性表，请阅读以下程序，并给出程序运行结果及其时间复杂度。

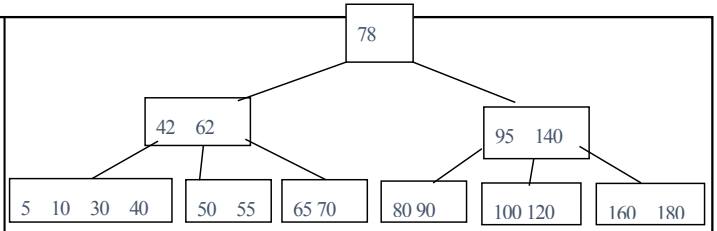
```
template<class T>
Chain<T>& Chain<T>::R ()
{  ChainNode<T> *last = 0,  // last node
                *current = first,
                                // current node
                *next;      // next node
   while (current) {

      next = current->link;
      current->link = last;
      last = current;
      current = next;
      }
   first = last;
   return *this;
}
```

得分	阅卷人

二、层次结构（35 分）。

1. 二叉树的层次遍历序列为 ABCDEFGHIJ，中序遍历序列为 DBGEHJACIF，写出该二叉树的前序遍历序列。
2. 一个最大堆为（66, 37, 41, 30, 25, 40, 35, 18），依次从中删除两个元素，写出最后得到的堆。
3. 有一份电文中共使用 6 个字符：A、B、C、D、E、F，它们的出现频率依次为 10、6、5、2、15、4，试画出对应的赫夫曼树（请按左子树根节点的权小于等于右子树根节点的权的次序构造，左 0 右 1），并求出每个字符的赫夫曼编码。
4. 对给定输入序列{ 19, 5, 7, 11, 26, 18, 16, 17 },构建 AVL 树。



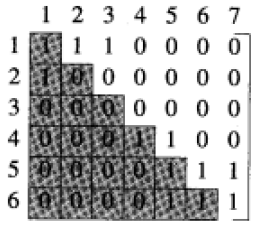
得分	阅卷人

三、网状结构（35 分）。

1. 请给出从加权无向图中生成最小耗费生成树的两种方法，请分别描述其算法思想，并给出各自的时间复杂度。
2. 下面是某有向加权图(顶点 A,B,C,D,E)的耗费邻接矩阵，先给出一个拓扑序列，然后，使用 Dijkstra 算法依次计算出顶点 A 至其它各顶点的最短路径和最短路径长度。

	A	B	C	D	E
A		6		40	50
B				10	
C					20
D			30		10
E					

3. a 是一个(n- 1)×n 的数组，用来描述一个 n 顶点图的邻接矩阵 A（如下图所示）。a 中没有描述矩阵的对角线。
   1) 编写两个函数 Store 和 Retrieve 分别存储和搜索 A(i, j) 的值，每个函数的复杂性应为Θ (1)。
   2) 编写函数 indegree（i），计算顶点 i 的入度，并分析其复杂度。


1 2 3 4 5 6 7
1 1 1 1 0 0 0 0
2 1 0 0 0 0 0 0
3 0 0 0 0 0 0 0
4 0 0 0 1 1 0 0
5 0 0 0 0 1 1 1
6 0 0 0 0 1 1 1

1、已知线性表：(8, 9, 2, 13, 0, 7, 1, 6, 5)，请完成以下题目。

(1) 请描述公式化描述及链表描述的空间需求。如果需要删除元素 13，请描述各自的时间复杂度。

(2) 请分别进行选择排序、插入排序、快速排序（以 8 为轴），并给出第一轮排序结束后各自的结果。

(3) 设计散列表，散列函数为 H (k)=k%7，散列表长度为 11，请给出线性开型寻址的散列表。

(4) 基于以上散列表，查找元素 1，给出需要的查找次数。

(5) 若使用单链表存储上述线性表，请阅读以下程序，并给出程序运行结果及其时间复杂度。

```
template<class T>
Chain<T>& Chain<T>::R ()
{  ChainNode<T> *last = 0,   // last node
                *current = first,
                           // current node
                *next;     // next node
   while (current) {
                n
      next = current->link;
      current->link = last;
      last = current;
      current = next;
      }
   first = last;
   return *this;
}
```

(1) 公式化描述空间需求为 9　删除13复杂度为 O(n)

链表描述空间需求为 18　删除13复杂度为 O(n)

(2) 8 9 2 13 0 7 1 6 5

选择排序 0 8 9 2 13 7 1 6 5 X　　0 9 2 13 8 7 1 6 5

改错!!!

插入排序 8 9 2 13 0 7 1 6 5

快速排序 8 9 2 13 0 7 1 6 5

8 5 2 6 0 7 1 13 9

1 5 2 6 0 7 8 13 9

(3)

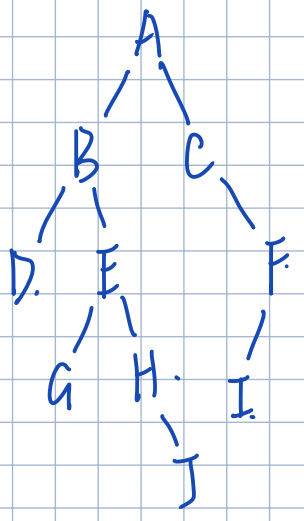
0	1	2	3	4	5	6	7	8	9	10
0	8	9	2	7	1	13	6	5		
1	1	1	2	5	5	1	1	4		

(4) 查找1需查找1次

(5) 结果(5, 6, 1, 7, 0, 13, 2, 9, 8)的单向链表

复杂度 O(n)

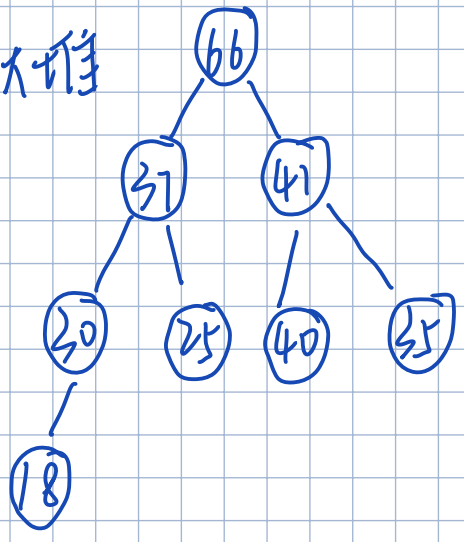
1. 二叉树的层次遍历序列为 ABCDEFGHIJ，中序遍历序列为 DBGEHJACIF，写出该二叉树的前序遍历序列。



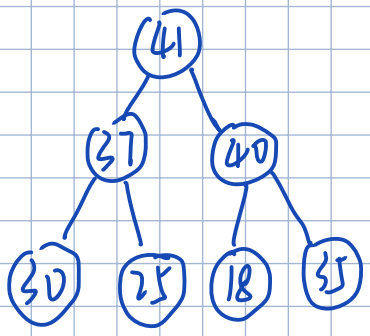
前序 A B D E G H J C F I

2. 一个最大堆为（66，37，41，30，25，40，35，18），依次从中删除两个元素，写出最后得到的堆。

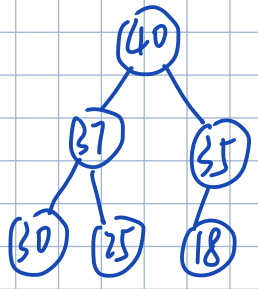
最大堆



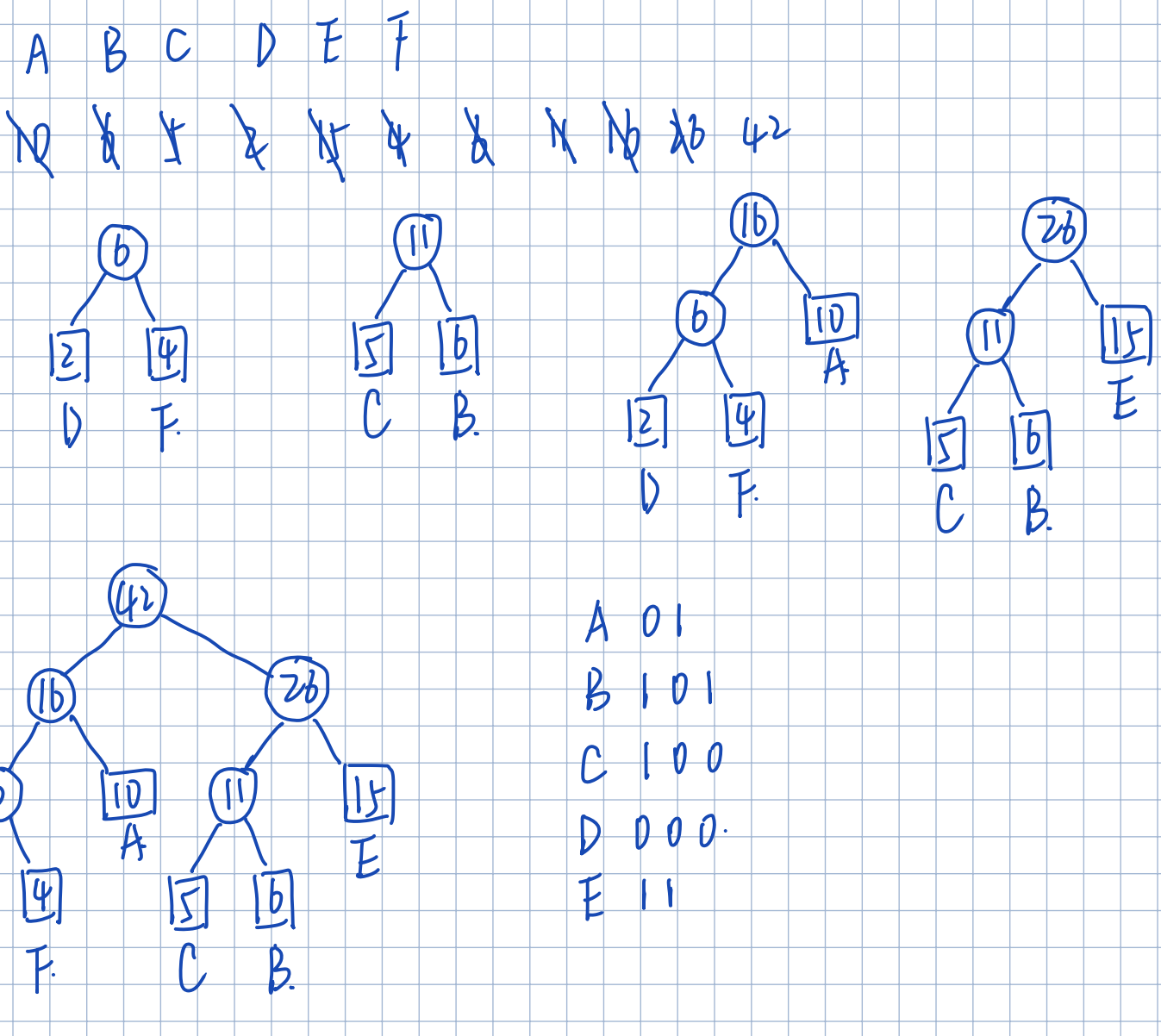
删66.



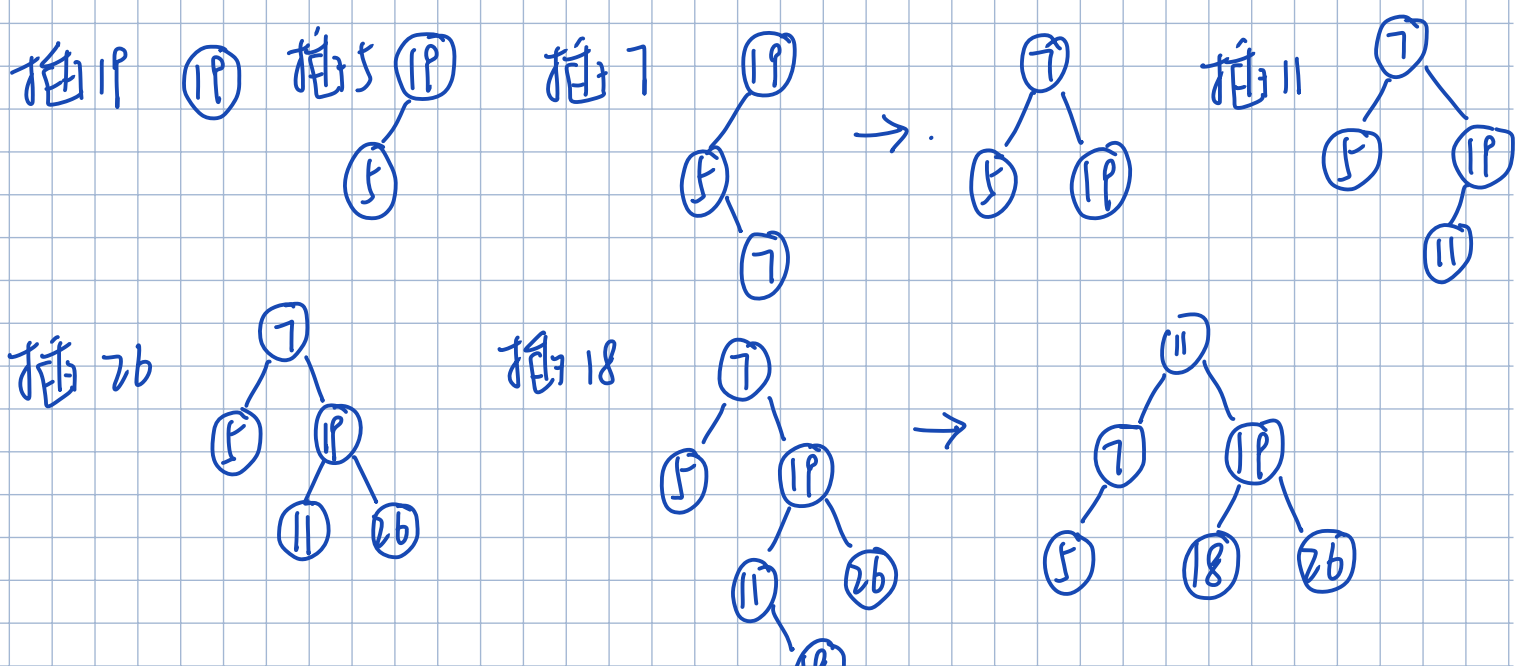
删41



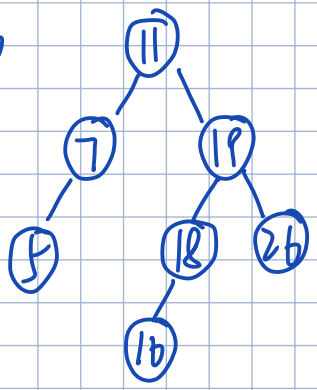
3. 有一份电文中共使用 6 个字符：A、B、C、D、E、F，它们的出现频率依次为 10、6、5、2、15、4，试画出对应的赫夫曼树（请按左子树根节点的权小于等于右子树根节点的权的次序构造，左 0 右 1），并求出每个字符的赫夫曼编码。



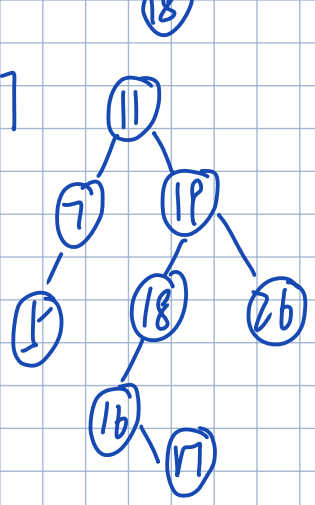
4. 对给定输入序列{ 19, 5, 7, 11, 26, 18, 16, 17 },构建 AVL 树。

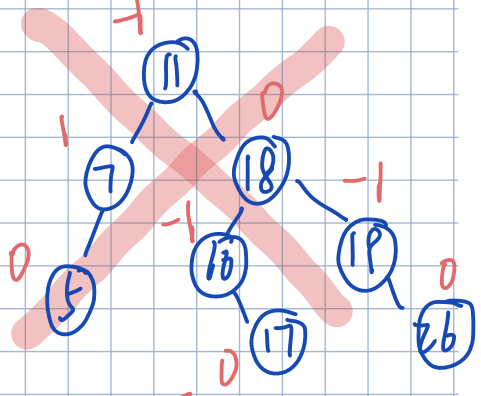


插16

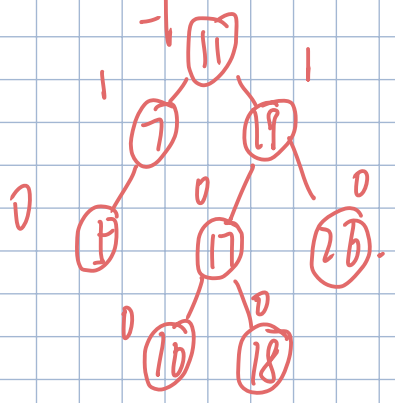


插17

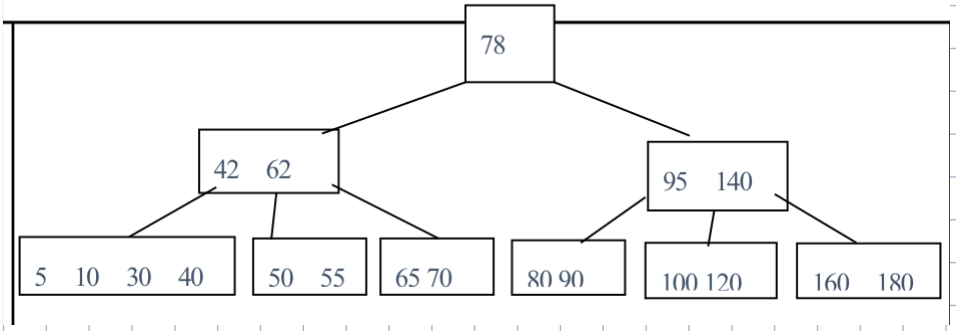




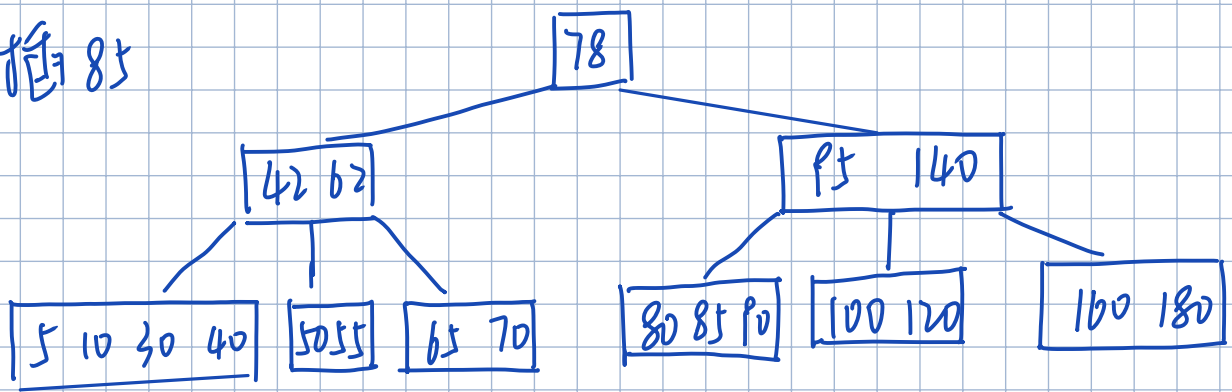
↓ 18是第一个不平衡点



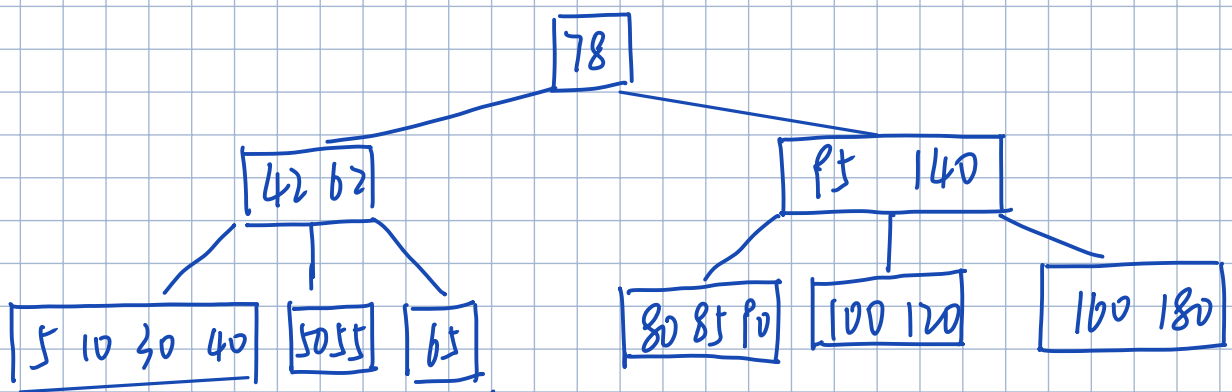
5. 在下列 5 阶 B-树中首先插入关键字 85, 然后删除关键字 70, 画出插入元素和删除元素后的 B-树。



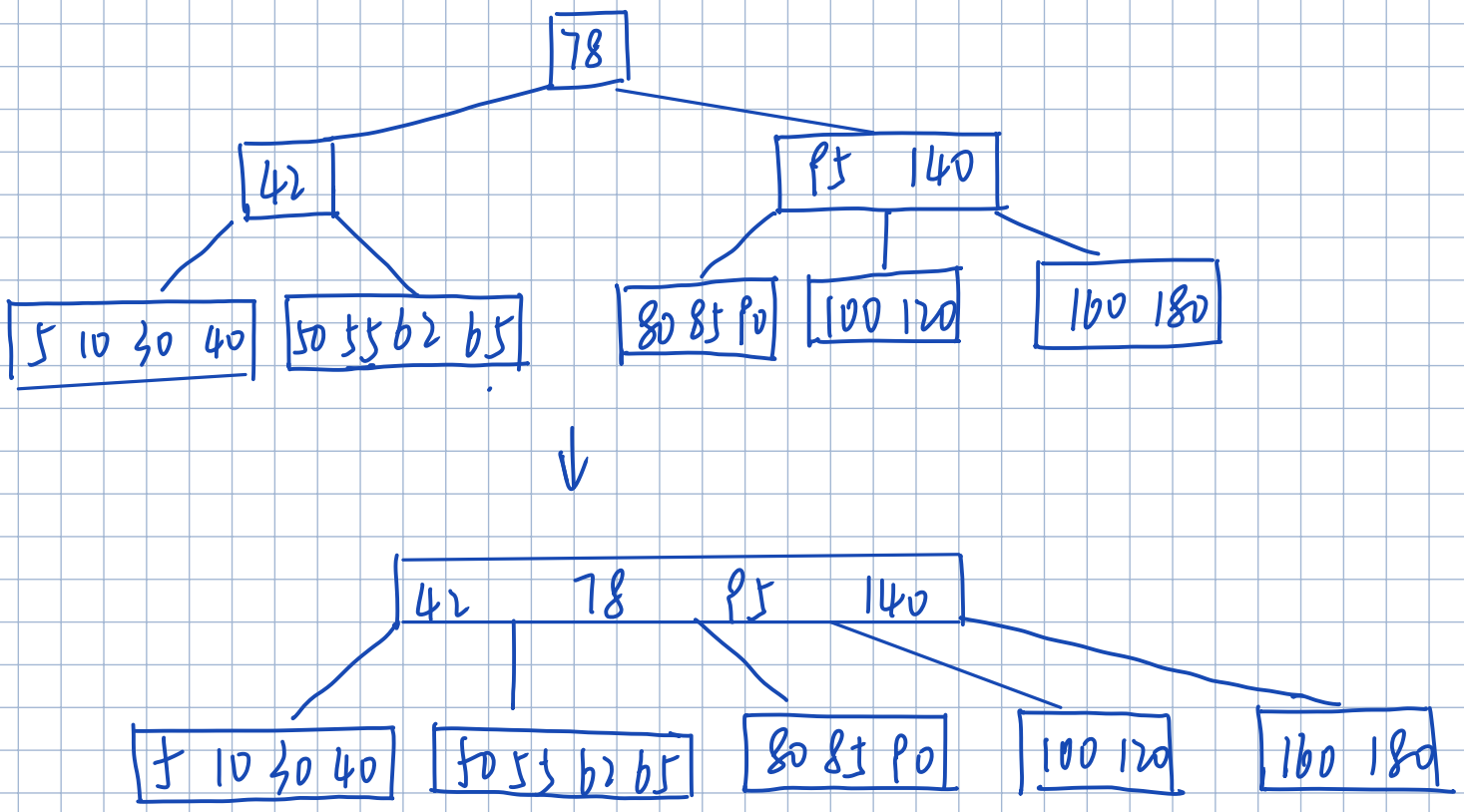
插85



删70



↓



1. 请给出从加权无向图中生成最小耗费生成树的两种方法，请分别描述其算法思想，并给出各自的时间复杂度。

Prim算法：从某一点开始，选取该点周围长度最小的边，将将边加入集合构成一棵数

当所有顶点都被加入集合后算法终止

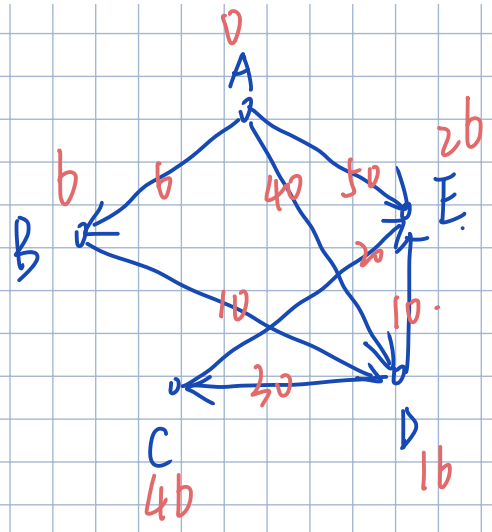
复杂度 $O(n^2)$

Kruscal算法：从所有的边集中挑选权最小的边并将该边加入集合A，每次都从未加入集合A的边中挑选权最小且不与A中边构成回路的边加入集合，直至所有顶点都被包含进集合A为止

复杂度 $O(n+e\log e)$ n为点数 e为边数

2. 下面是某有向加权图(顶点 A,B,C,D,E)的耗费邻接矩阵，先给出一个拓扑序列，然后，使用 Dijkstra 算法依次计算出顶点 A 至其它各顶点的最短路径和最短路径长度。

	A	B	C	D	E
A		6		40	50
B				10	
C					20
D			30		10
E					



拓扑序列：A B D C E

A → B 6.

A → B → D 16

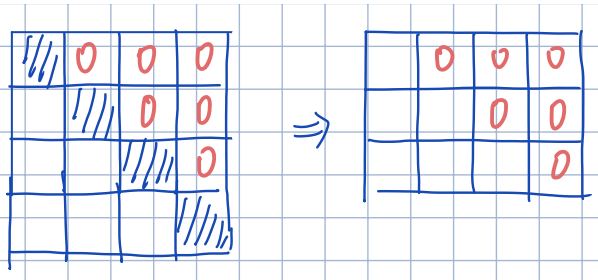
A → B → D → E 26.

A → B → D → C 46

3. a 是一个(n- 1)×n 的数组，用来描述一个 n 顶点图的邻接矩阵 A（如下图所示）。a 中没有描述矩阵的对角线。

1) 编写两个函数 Store 和 Retrieve 分别存储和搜索 A(i, j) 的值，每个函数的复杂性应为Θ (1)。

2) 编写函数 indegree (i)，计算顶点 i 的入度，并分析其复杂度。



	1	2	3	4	5	6	7
1	1	1	1	0	0	0	0
2	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	1	1	0	0
5	0	0	0	0	1	1	1
6	0	0	0	0	1	1	1

```
void Store(int* A, int i, int j, int m){
    int a,b; // 映射后行列坐标
    if(i==j){return;}
    if(i<j){a=i-1; b=j;}
    else if(i>j){a=i; b=j-1;}
    A[a][b]=m;
```

```
        return j;
}

int Retrieve(int* A, int i, int j){
    int a,b; // 映射后行列坐标.
    if(i==j){return j;}
    if(i>j){a=i-1; b=j;}
    else if(i<j){a=i; b=j;}
    return A[a][b];
}

int degree(int* A, int i){
    int count=0;
    for(int j=0; j<n-1; j++){
        if(A[j][i]==1){
            count++;
        }
    }
    return count;
}
```