《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第1节 程序基本常识

目录

一、算法的特征

二、算法的空间复杂度

三、算法的时间复杂度

算法特征

- **有穷性**：执行有穷步，在有穷的时间内完成。
- **确切性**：每一条指令必须有确切的含义，不会产生歧义。在任何条件下算法只有唯一的一条执行路径。
- **可行性**：算法中的操作可以通过执行有限次来实现。
- **输入**：一个算法有零个或者多个输入。
- **输出**：一个算法有一个或者多个输出。

算法的复杂度

- **空间复杂度：**指执行算法所需占用的内存空间。
- **时间复杂度：**指算法执行时所需消耗时间，通常用算法执行次数来衡量，记作T(n)=O(f(n))，其中f(n)是算法执行次数的函数。

时间复杂度举例

- 例1：求出以下算法的时间复杂度。

```
void fundemo(int n){
    int i=1,j=100;
    while(i<n){
        i+=2;
        ++j;
    }
}
```

分析：该算法基本操作为i+=2和++j，规模为n。i最后的值为1+2m(m代表执行次数)，则1+2m+k=n(k是足够小的数)，m=n-1-k/2，故f(n)=n-1-k/2，此时f(n)增长最快的那项是n，故T(n)=O(f(n))=O(n)。

时间复杂度计算规则

- 加法规则

  T(n,m) = T1(n) + T2(m) = O(max{f(n), g(m)})　　//并列算法

- 乘法规则

  T(n,m) = T1(n) * T2(m) = O(max{f(n)*g(m)})　　//嵌套算法

- 比较规则

$$O(1) < O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < \ldots < O(n^K) < O(2^n) < O(n!) < O(n^n)$$

【课堂练习】

课堂练习

1.【NOIP2011】在使用高级语言编写程序时，一般提到的“空间复杂度”中的“空间”是指(　　)。

A.程序运行时理论上所占的内存空间　　B.程序运行时理论上所占的数组空间

C.程序运行时理论上所占的硬盘空间　　D.程序源文件理论上所占的硬盘空间

2.【NOIP2013】斐波那契数列的定义如下：$F_1 = 1$，$F_2 = 1$，$F_n = F_{n-1} + F_{n-2}$ $(n \geq 3)$。如果用下面的函数计算斐波那契数列的第 n 项，则其时间复杂度为(　　)。

```
int F(int n){
if (n <= 2)
return 1;
else
return F(n - 1) + F(n - 2);
}
```

A.O(1)　　B.O(n)　　C.O(n2)　　D.O(Fn)

3.【NOIP2013】T(n)表示某个算法输入规模为 n 时的运算次数。如果 T(1)为常数，且有递归式 T(n) =2*T(n / 2) + 2n，那么 T(n) = (　　)。

A.O(n)　　B.O(n log n)　　C.$O(n^2)$　　D.$O(n^2 \log n)$

课堂练习

4.【NOIP2015】设某算法的计算时间表示为递推关系式 T(n) = T(n - 1) + n(n 为正整数)及 T(0) = 1，则该算法的时间复杂度为(　　)。

A.O(log n)　　B.O(n log n)　　C.O(n)　　D.O(n^2)

5.【NOIP2018】设某算法的时间复杂度函数的递推方程是T(n) = T(n - 1) + n（n 为正整数）及T(0) = 1，则该算法的时间复杂度为（ ）。

A. O(log n)　　B. O(n log n)　　C. O(n)　　D. O(n^2)

6.【NOIP2016】假设某算法的计算时间表示为递推关系式

$$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$$

T(1) = 1

则算法的时间复杂度为（）。

A. O(n)　　B. O($\sqrt{n}$)　　C. O($\sqrt{n}\log n$)　　D. O(n^2)

7.【NOIP2017】若某算法的计算时间表示为递推关系式：

T(N) = 2T(N / 2) + N log N

T(1) = 1

则该算法的时间复杂度为（）。

A. O(N)　　B. O(N log N)　　C. O(N log2 N)　　D. O(N2)

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第2节 C++语言基础

目录

一、程序的基本构成

二、变量及数据类型

三、类

四、程序的基本结构

五、函数

六、递归函数

02 程序的基本构成

```
#include <iostream>

using namespace std;

int main() {//main()是程序开始执行的地方

    cout <<"He11o world";//输出Hello world

    return 0;

}
```

① 头文件 <iostream>包含了程序中如输入输出即cin、cout的定义。
② using namespace std; 告诉编译器使用 std 命名空间。命名空间是为了作为附加信息来区分不同库中相同名称的函数、类、变量等。本质上，命名空间就是定义了一个范围。
③ 下一行 int main() 是主函数，程序从这里开始执行。
④ // 是一个单行注释，用于解释说明。单行注释以 // 开头，在行末结束。
⑤ cout << "Hello World"; 用于输出，在屏幕上显示消息 "Hello World"。
⑥ return 0; 终止 main( )函数，并向调用进程返回值 0。
⑦ 分号是语句结束符。也就是说每个语句必须以分号结束。
⑧ {}称为块，其中的所有语句是一个语句块。

变量及数据类型

- 数据类型
  - 基本数据类型
    - 布尔型bool
    - 字符型char
    - 整型int
    - 实型
      - 单精度型/浮点型float
      - 双精度型double
  - 非基本数据类型
    - 数组[]
    - 指针*
    - 空类型void
    - 结构体struct
    - 联合体union
    - 枚举enum
    - 类class

类

```
class Box { //方法一
    public: //public是类型修饰符，具体意义见“类访问修饰符”
        double length;//长度
        double width;//宽度
        double height;//高度
        double getVolume() {
            return length * width * height;
        }
};
```

类的数据成员,也就是类中的变量

类的成员函数,也就是类中的函数

程序的基本结构

一般一个程序可以用顺序、选择和循环三种基本结构组合而成。

1.顺序结构

顺序结构是最简单、最常用的结构，语句与语句之间按从上到下的顺序执行即可。

2.选择结构

选择结构是先根据条件做出判断，再决定执行哪一种操作的算法结构，它必须包含判断框。当条件P成立（或称为真）时执行A，否则执行B，不可能两者同时执行，但A或B两个框中可以有一个是空的，即不执行任何操作。常用的是if、if...else、switch。

3.循环结构

在一些算法中，经常会出现从某处开始，按照一定条件，反复执行某一处理步骤的情况，这就是循环结构。常用的有while、for、do...while

函数

函数是用来实现某些特定功能而封装成的一个模块。在创建函数时，必须编写其定义。所有函数定义包括以下组成部分：

```
返回类型 函数名（形参列表）{

    函数主体

}
```

递归函数

递归，就是在运行的过程中调用自己。

构成递归需具备的条件：

1. 子问题须与原始问题为同样的事，且更为简单；

2. 不能无限制地调用本身，须有个出口，化简为非递归状况处理。

【课堂练习】

课堂练习

1.【NOIP2014】若有如下程序段，其中 s、a、b、c 均已定义为整型变量，且 a、c 均已赋值，c > 0。

```
s = a;
for (b = 1; b <= c; b++)
s += 1;
```

则与上述程序段功能等价的赋值语句是(　　)。

A.s = a + b　　B.s = a + c　　C.s = s + c　　D.s = b + c

2.【NOIP2014】要求以下程序的功能是计算：s = 1 + 1/2 + 1/3 + ... + 1/10。

```
#include <iostream>
using namespace std;
int main() {
    int n;
    float s;
    s = 1.0;
    for (n = 10; n > 1; n--)
        s = s + 1 / n;
    cout << s << endl;
    return 0;
}
```

程序运行后输出结果错误，导致错误结果的程序行是(　　)。

A.s = 1.0;　　B.for (n = 10; n > 1; n--)

C.s = s + 1 / n;　　D.cout << s << endl;

02 课堂练习

3.【NOIP2014】有以下程序：

```
#include <iostream>
using namespace std;
int main() {
    int s, a, n;
    s = 0;
    a = 1;
    cin >> n;
    do {
        s += 1;
        a -= 2;
    } while (a != n);
    cout << s << endl;
    return 0;
}
```

若要使程序的输出值为 2，则应该从键盘给 n 输入的值是（　　）。

A.-1　　B.-3　　C.-5　　D.0

4.【NOIP2016】有以下程序：

```
#include <iostream>
using namespace std;
int main() {
    int k = 4, n = 0;
    while (n < k) {
        n++;
        if (n % 3 != 0)
            continue;
        k--;
    }
    cout << k <<","<< n << endl;
    return 0;
}
```

程序运行后的输出结果是（）。

A. 2,2　　B. 2,3　　C. 3,2　　D. 3,3

02 课堂练习

5.【NOIP2018】为了统计一个非负整数的二进制形式中1的个数，代码如下：

```
int CountBit(int x){
        int ret = 0;
        while (x){
                ret++;
                ____________;
        }
        return ret;
}
```

则空格内要填入的语句是（　　）。

A.x>>=1　　B.x&=x-1

C.x|=x>>1　　D.x<<=1

6、【NOIP2013】下列程序中，正确计算 1, 2, …, 100 这 100 个自然数之和 sum(初始值为 0)的是(　　)。

A.	`i = 1;`
	`do {`
	`  sum += i;`
	`  i++;`
	`} while (i <= 100);`
		B.	`i = 1;`
			`do {`
			`  sum += i;`
			`  i++;`
			`} while (i > 100);`

C.	`i = 1;`
	`while (i < 100) {`
	`  sum += i;`
	`  i++;`
	`}`
		D.	`i = 1;`
			`while (i >= 100) {`
			`  sum += i;`
			`  i++;`
			`}`

02 课堂练习

7.【NOIP2014】若有变量 int a, float x, y, 且 a=7, x=2.5, y=4.7, 则表达式 x+a%3*(int)(x+y)%2/4的值大约是( ).

A. 2.500000　　B. 2.750000

C. 3.500000　　D. 0.000000

8、【NOIP2014】设变量 x 为 float 型且已赋值，则以下语句中能将 x 中的数值保留到小数点后两位，并将第三位四舍五入的是(　　)。

A.x = (x * 100) + 0.5 / 100.0;

B.x = (x * 100 + 0.5) / 100.0;

C.x = (int) (x * 100 + 0.5) / 100.0;

D.x = (x / 100 + 0.5) * 100.0;

9.【NOIP2014】以下程序段实现了找第二小元素的算法。输入是 n 个不等的数构成的数组 S，输出 S 中第二小的数 SecondMin。在最坏情况下，该算法需要做(　　)次比较。

```
if(S[1]<S[2]){
    FirstMin=S[1];
    SecondMin=S[2];
}else{
    FirstMin=S[2];
    SecondMin=S[1];
}
for(i=3;i<=n;i++)
    if(S[i]<SecondMin)
    if(S[i]<FirstMin){
        SecondMin=FirstMin;
        FirstMin=S[i];
    }else{
        SecondMin=S[i];
}
```

A.2n　　B.n-1　　C.2n-3　　D.2n-2

02 课堂练习

10. 【NOIP2008】递归过程或函数调用时，处理参数和返回地址，通常使用一种称为（）的数据结构。

A. 队列　B. 多维数组　C. 线性表　D. 栈

11.【NOIP2012】在程序运行过程中，如果递归调用的层数过多，会因为(　　)引发错误。

A.系统分配的栈空间溢出

B.系统分配的堆空间溢出

C.系统分配的队列空间溢出

D.系统分配的链表空间溢出

【不定项选择题】

02 课堂练习

1.【NOIP2013】下列程序中，正确计算 1, 2, …, 100 这 100 个自然数之和 sum(初始值为 0)的是(　　)。

A.
```
for (i = 1;i<=100;i++)
    sum += i;
```

B
```
i = 1;
while (i > 100) {
    sum += i;
    i++;
}
```

C.
```
i = 1;
do{
    sum += i;
    i++;
} while (i <= 100);
```

D
```
i = 1;
do{
   sum+=i;
   i++;
}while (i > 100);
```

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第3节 排序算法

目录

一、内排序和外排序的定义

二、衡量效率的方法

三、排序稳定性

四、常用排序算法复杂度

内排序和外排序的定义

由于待排序的记录数量不同，使得排序过程中涉及的存储器不同，可将排序方法分为两大类：内排序与外排序。

1.内排序：待排序记录存放在计算机内存中进行的排序过程。插入排序、快速排序、选择排序、归并排序、基数排序等目前竞赛研究的算法基本上都是内排序方法。

2.外排序：待排序记录的数量很大，以致于内存不能一次容纳全部记录，所以在排序过程中需要对外存进行访问的排序过程。

衡量效率的方法

1.内排序：比较次数，也就是时间复杂度。

2.外排序：IO次数，也就是读写外存的次数。

排序稳定性

排序前后相同元素的相对位置不变，则称排序算法是稳定的，否则排序算法是不稳定的。如原序列ri=rj且ri位于rj之前，排序后ri仍在rj之前，则称该排序是稳定的。

常用排序算法复杂度

排序法	最坏时间	平均时间复杂度	稳定性	空间复杂度
插入排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
选择排序	$O(n^2)$	$O(n^2)$	不稳定	O(1)
冒泡排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
希尔排序	$O(n^s)$,$s < 1 < 2$，取决于增量序列	$O(n\log_2 n)$	不稳定	O(1)
快速排序	$O(n^2)$	$O(n\log_2 n)$	不稳定	$O(\log_2 n)$
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	不稳定	O(1)
归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	稳定	$O(n)$

【课堂练习】

03 课堂练习

1.【NOIP2011】体育课的铃声响了，同学们都陆续地奔向操场，按老师的要求从高到矮站成一排。每个同学按顺序来到操场时，都从排尾走向排头，找到第一个比自己高的同学，并站在他的后面。这种站队的方法类似于(　　)算法。

A.快速排序　　B.插入排序　　C.冒泡排序　　D.归并排序

2.【NOIP2018】以下排序算法中，不需要进行关键字比较操作的算法是(　　)。

A.基数排序　　B.冒泡排序　　C.堆排序　　D.直接插入排序

3.【NOIP2009】排序算法是稳定的意思是关键码相同的记录排序前后相对位置不发生改变，下列哪种排序算法是不稳定的：

A.冒泡排序　　B.插入排序　　C.归并排序　　D.快速排序

4.【NOIP2009】快速排序最坏情况下的算法复杂度为：

A.$O(\log_2 n)$　　B.$O(n)$　　C.$O(n\log_2 n)$　　D.$O(n^2)$

03 课堂练习

5.【NOIP2009】快速排序平均情况和最坏情况下的算法时间复杂度分别为：

A.平均情况 $O(n\log_2 n)$，最坏情况 $O(n^2)$　　B.平均情况 $O(n)$，最坏情况 $O(n^2)$

C.平均情况 $O(n)$，最坏情况 $O(n\log_2 n)$　　D.平均情况 $O(\log_2 n)$，最坏情况 $O(n^2)$

6.【NOIP2012】如果不在快速排序中引入随机化，有可能导致的后果是(　　)。

A.数组访问越界　　B.陷入死循环

C.排序结果错误　　D.排序时间退化为平方级

7.【NOIP2010】基于比较的排序时间复杂度的下限是(　　),其中 n 是待排的元素个数。

A. $O(n)$　　B. $O(n \log n)$　　C. $O(\log n)$　　D. $O(n^2)$

8.【NOIP2013】(　　)的平均时间复杂度为 $O(n \log n)$，其中 n 是待排序的元素个数。

A.快速排序　　B.插入排序　　C.冒泡排序　　D.基数排序

9.【NOIP2014】以下时间复杂度不是 $O(n^2)$的排序方法是(　　)。

A.插入排序　　B.归并排序　　C.冒泡排序　　D.选择排序

03 课堂练习

10.【NOIP2017】对于给定的序列$\{a_k\}$，我们把 (i, j) 称为逆序对当且仅当 i < j 且 $a_i > a_j$。那么序列 1, 7, 2, 3, 5, 4 的逆序对数为（）个。

A. 4　　B. 5　　C. 6　　D. 7

11.【NOIP2012】使用冒泡排序对序列进行升序排序，每执行一次交换操作将会减少 1 个逆序对，因此序列 5, 4, 3, 2, 1 需要执行(　　)次交换操作，才能完成冒泡排序。

A.0　　B.5　　C.10　　D.15

12.【NOIP2017】设 A 和 B 是两个长为 n 的有序数组，现在需要将 A 和 B 合并成一个排好序的数组，任何以元素比较作为基本运算的归并算法在最坏情况下至少要做（）次比较。

A. n^2　　B. n log n　　C. 2n　　D. 2n-1

【不定项选择题】

课堂练习

1.【NOIP2009】排序算法是稳定的意思是关键码相同的记录排序前后相对位置不发生改变，下列哪些排序算法是稳定的：

A.插入排序　　B.基数排序　　C.归并排序　　D.冒泡排序

2.【NOIP2017】下列算法中，(　　)是稳定的排序算法。

A．快速排序　　B．堆排序　　C．希尔排序　　D．插入排序

3.【NOIP2010】原地排序是指在排序过程中(除了存储待排序元素以外的)辅助空间的大小与数据规模无关的排序算法。以下属于原地排序的有(　　)。

A.冒泡排序　　B.插入排序　　C.基数排序　　D.选择排序

4.【NOIP2016】下列算法中运用分治思想的有(　　)。

A．快速排序　　B．归并排序　　C．冒泡排序　　D．计数排序

03 课堂练习

5.【NOIP2013】(　　)的平均时间复杂度为 O(n log n)，其中 n 是待排序的元素个数。

A.快速排序　　B.插入排序　　C.冒泡排序　　D.归并排序

6.【NOIP2017】以下排序算法在最坏情况下时间复杂度最优的有(　　)。

A．冒泡排序　　B．快速排序　　C．归并排序　　D．堆排序

7.【NOIP2011】对于序列“7、5、1、9、3、6、8、4”，在不改变顺序的情况下，去掉(　　)会使逆序对的个数减少 3。

A．7　　B．5　　C．3　　D．6

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第4节 基础算法

基础算法

一、高精度计算

高精度数值处理是采用模拟算法对位数达到上百位甚至更多位数的数字进行各种运算，包括加法、减法、乘法、除法等基础运算。核心思想是对参与运算的两个数分别用两个数组进行存储。常见问题有“汉诺塔的步数”、“国王的米粒”等。

二、穷举算法

穷举法即枚举法，是从可能的解的集合中一一枚举各元素， 用题目给定的检验条件判定哪些是无用的，哪些是有用的。能使命题成立即为其解。常见问题有“百钱买百鸡”、“素数判断”等。

三、递推算法

递推法是一种重要的数学方法，在数学的各个领域中都有广泛的运用，也是计算机用于数值计算的一个重要算法。递推算法的首要问题是得到相邻的数据项间的关系（即递推关系）。递推算法避开了求通项公式的麻烦，把一个复杂的问题的求解，分解成了连续的若干步简单运算。一般说来，可以将递推算法看成是一种特殊的迭代算法。常见问题有“斐波那契数列”、“过河卒”等。

四、递归算法

递归程序设计是C++语言程序设计中的一种重要的方法，它使许多复杂的问题变得简单容易解决了。递归特点是函数或过程调用它自己本身 ，其中直接调用自己称为直接递归，而将A 调用B，B 以调用 A 的递归叫做间接递归。 常见问题有“汉诺塔问题”、“Ackermann函数”等。

基础算法

五、搜索与回溯算法

搜索与回溯是计算机解题中常用的算法，很多问题无法根据某种确定的计算法则来求解，可以利用搜索与回溯的技术求解。回溯是搜索算法中的一种控制策略。它的基本思想是：为了求得问题的解，先选择某一种可能情况向前探索，在探索过程中，一旦发现原来的选择是错误的，就退回一步重新选择，继续向前探索，如此反复进行，直至得到解或证明无解。常见问题有“八皇后问题”、“骑士游历问题”等。

六、贪心算法

贪心算法是指对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，它所做出的仅是在某种意义上的局部最优解。

贪心算法没有固定的算法框架，算法设计的关键是贪心策略的选择。必须注意的是，贪心算法不是对所有问题都能得到整体最优解，选择的贪心策略必须具备无后效性，即某个状态以后的过程不会影响以前的状态，只与当前状态有关。所以对所采用的贪心策略一定要仔细分析其是否满足无后效性。

常见问题有“删数问题”、“排队打水问题”等。

七、分治算法

分治就是指的分而治之，即将较大规模的问题分解成几个较小规模的问题，通过对较小规模问题的求解达到对整个问题的求解。将问题分解成两个较小问题求解时的分治方法称之为二分法。常见问题有“归并排序”、“比赛日程安排”等。

基础算法

八、广度优先搜索

广度优先算法的核心思想是：从初始节点开始，应用算符生成第一层节点，检查目标节点是否在这些后继节点中，若没有，再用产生式规则将所有第一层的节点逐一扩展，得到第二层节点，并逐一检查第二层节点中是否包含目标节点。若没有，再用算符逐一扩展第二层的所有节点……，如此依次扩展，检查下去，直到发现目标节点为止。这种搜索的次序体现沿层次向横向扩展的趋势，所以称之为广度优先搜索。

常见问题有“分油问题”、“八数码问题”等。

九、动态规划

动态规划程序设计是解决多阶段决策过程最优化问题的一种途径。动态规划的设计方法对不同的问题，有各具特色的解题方法，而不存在一种万能的动态规划算法，可以解决各类最优化问题。需要满足“最优子结构”和“无后效性”的两项基本条件。动态规划解决问题的步骤：

1. 确定状态：状态是一个数学形式；
2. 确定状态转移方程和边界条件：递归或递推；
3. 程序实现。

动态规划大致可以分为以下几类：线性、区间、背包型、树型、数位、状态压缩等。常见问题有“最长不下降序列”、“最长公共子序列”等。

【课堂练习】

课堂练习

1.【NOIP2016】周末小明和爸爸妈妈三个人一起想动手做三道菜。小明负责洗菜、爸爸负责切菜、妈妈负责炒菜。假设做每道菜的顺序都是：先洗菜 10 分钟，然后切菜 10 分钟，最后炒菜 10 分钟。那么做一道菜需要 30 分钟。注意：两道不同的菜的相同步骤不可以同时进行。例如第一道菜和第二道的菜不能同时洗，也不能同时切。那么做完三道菜的最短时间需要(　　)分钟。

A. 90　　B. 60　　C. 50　　D. 40

2.【NOIP2017】2017 年 10 月 1 日是星期日，1999 年 10 月 1 日是（）。

A. 星期三　　B. 星期日　　C. 星期六　　D. 星期二

3.【NOIP2018】下面的故事与(　　)算法有着异曲同工之妙。

从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事：从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事：从前有座山，山里有座庙，庙里有个老和尚给小和尚讲故事....

A.枚举　　B.递归　　C.贪心　　D.分治

04 课堂练习

4.【NOIP2011】(　　)是一种选优搜索法，按选优条件向前搜索，以达到目标。当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择。

A.回溯法　　B.枚举法　　C.动态规划　　D.贪心法

5.【NOIP2013】将(2, 6, 10, 17)分别存储到某个地址区间为 0~10 的哈希表中，如果哈希函数 h(x) = (　　)，将不会产生冲突，其中 a mod b 表示 a 除以 b 的余数。

A.x mod 11　　B.x^2 mod 11

C.2x mod 11　　D.$\lfloor\sqrt{x}\rfloor$ mod 11，其中$\lfloor\sqrt{x}\rfloor$表示$\sqrt{x}$下取整

6.【NOIP2012】(　　)就是把一个复杂的问题分成两个或者更多的相同或相似的子问题，再把子问题分成更小的子问题……直到最后的子问题可以简单的直接求解。而原问题的解就是子问题解的并。

A.动态规划　　B.贪心　　C.分治　　D.搜索

04 课堂练习

7.【NOIP2016】给定含有 n 个不同的数的数组 L=<x1, x2, ..., xn>。如果 L 中存在 x（1 < i < n）使得 x1 < x2 < ... < xi-1 < xi > xi+1 > ... > xn，则称 L 是单峰的，并称 xi 是 L 的“峰顶”。现在已知 L 是单峰的，请把 a-c 三行代码补全到算法中使得算法正确找到 L 的峰顶。

a. Search(k+1, n)　　b. Search(1, k-1)　　c. return L[k]

```
Search(1, n)
1. k←⌊n/2⌋
2. if L[k] > L[k-1] and L[k] > L[k+1]
3. then __________
4. else if L[k] > L[k-1] and L[k] < L[k+1]
5. then __________
6. else __________
```

正确的填空顺序是(　　)。

A. c, a, b　　B. c, b, a　　C. a, b, c　　D. b, a, c

04 课堂练习

8.【NOIP2017】在 n（n ≥ 3）枚硬币中有一枚质量不合格的硬币（质量过轻或质量过重），如果只有一架天平可以用来称重且称重的硬币数没有限制，下面是找出这枚不合格的硬币的算法。请把 a-c 三行代码补全到算法中。

（a） A←X ∪ Y

（b） A←Z

（c） n←|A|

算法 Coin(A, n)

（1） k ←⌊n/3⌋

（2） 将 A 中硬币分成 X，Y，Z 三个集合，使得$|X| = |Y| = k$，$|Z| = n - 2k$

（3） if$W(X) \neq W(Y)$ //W(X)，W(Y)分别为 X 或 Y 的重量

（4） then _____

（5） else _____

（6） _____

（7） if n>2 then goto 1

（8） if n=2 then 任取 A 中 1 枚硬币与拿走硬币比较，若不等，则它不合格；若相等，则 A 中剩下的硬币不合格.

（9） if n=1 then A 中硬币不合格

正确的填空顺序是（）。

A. b, c, a　　B. c, b, a　　C. c, a, b　　D. a, b, c

04 课堂练习

9.【NOIP2011】应用快速排序的分治思想，可以实现一个求第 K 大数的程序。假定不考虑极端的最坏情况，理论上可以实现的最低的算法时间复杂度为(　　)。

A. $O(n^2)$　　B. O(n log n)　　C. O(n)　　D. O(1)

10.【NOIP2014】设有 100 个数据元素，采用折半搜索时，最大比较次数为(　　)。

A.6　　B.7　　C.8　　D.10

11.【NOIP2009】有一个由 4000 个整数构成的顺序表，假定表中的元素已经按升序排列，采用二分查找定位一个元素。则最多需要几次比较就能确定是否存在所查找的元素：

A.11 次　　B.12 次　　C.13 次　　D.14 次

课堂练习

12.【NOIP2008】对有序数组{5，13，19，21，37，56，64，75，88，92，100}进行二分查找，成功查找元素 19 的查找长度（比较次数）是（）。

A．1　　B．2　　C．3　　D．4

13.【NOIP2008】对有序数组{5，13，19，21，37，56，64，75，88，92，100}进行二分查找，等概率的情况下查找成功的平均查找长度（平均比较次数）是（ ）。

A.35/11　　B.34/11　　C.33/11　　D.32/11　　E.34/10

14.【NOIP2015】在数据压缩编码的应用中，哈夫曼(Huffman)算法是一种采用了(　　)思想的算法。

A.贪心　　B.分治　　C.递推　　D.回溯

15.【NOIP2008】将数组{8，23，4，16，77，-5，53，100}中的元素按从大到小的顺序排列，每次可以交换任意两个元素，最少需要交换（）次。

A．4　　B．5　　C．6　　D．7

04 课堂练习

16.【NOIP2018】给定一个含 N 个不相同数字的数组，在最坏情况下，找出其中最大或最小的 数，至少需要 N - 1 次比较操作。则最坏情况下，在该数组中同时找最大与 最小的数至少需要（ ）次比较操作。（⌈⌉表示向上取整，⌊⌋表示向下取整）

A. ⌈3N/2⌉ - 2　　B. ⌊3N/2⌋ - 2　　C. 2N - 2　　D. 2N - 4

17.【NOIP2017】有正实数构成的数字三角形排列形式如图所示。

第一行的数为 a_{11}；第二行的数从左到右依次为 a_{21}，a_{22}；…第 n 行的数为 a_{n1}，a_{n2}，…，a_{nn}。从 a_{11} 开始，每一行的数 a_{ij} 只有两条边可以分别通向下一行的两个数 $a_{(i+1)j}$ 和 $a_{(i+1)(j+1)}$。用动态规划算法找出一条从 a_{11} 向下通到 a_{n1}，a_{n2}，…，a_{nn} 中某个数的路径，使得该路径上的数之和达到最大。

$$\begin{array}{ccccc} & & a_{11} & & \\ & a_{21} & & a_{22} & \\ a_{31} & & a_{32} & & a_{33} \\ & & \cdots\cdots & & \\ a_{n1} & & a_{n2} & \cdots\cdots & a_{nm} \end{array}$$

令 C[i,j]是从 a_{11} 到 a_{ij} 的路径上的数的最大和，并且 C[i,0]=C[0,j]=0,则 C[i,j]=（）。

A. max{C[i-1,j-1], C[i-1,j]} + a_{ij}　　B. C[i-1,j-1] + C[i-1,j]

C. max{C[i-1,j-1], C[i-1,j]} + 1　　D. max{C[i,j-1],C[i-1,j]} + a_{ij}

【不定项选择题】

04 课堂练习

1.【NOIP2009】散列表的地址区间为 0-10,散列函数为 H(K)=K % 11。采用开地址法的线性探查法处理冲突，并将关键字序列 26，25，72，38，8，18，59 存储到散列表中，这些元素存入散列表的顺序并不确定。假定之前散列表为空，则元素 59 存放在散列表中的可能地址有：

A.5　　B.7　　C.9　　D.10

2.【NOIP2012】从顶点 A_0 出发，对有向图(　　)进行广度优先搜索(BFS)时，一种可能的遍历顺序是 A_0，A_1，A_2，A_3，A_4。

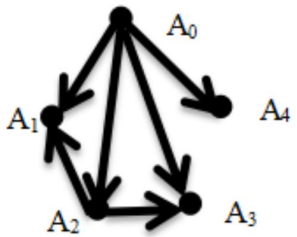


图 A

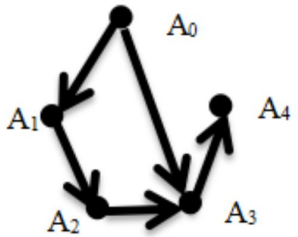


图 B

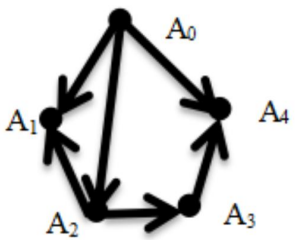


图 C

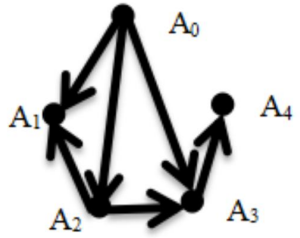
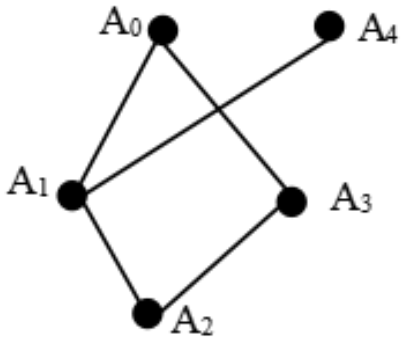


图 D

04 课堂练习

3.【NOIP2013】以 A_0 作为起点，对下面的无向图进行**深度**优先遍历时(遍历的顺序与顶点字母的下标无关)，最后一个遍历到的顶点可能是(　　)。



A.A_1　　B.A_2　　C.A_3　　D.A_4

4.【NOIP2011】现有一段文言文，要通过二进制哈夫曼编码进行压缩。简单起见，假设这段文言文只由 4 个汉字“之”、“乎”、“者”、“也”组成，它们出现的次数分别为 700、600、300、400。那么，“也”字的编码长度可能是(　　)。

A．1　　B．2　　C．3　　D．4

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第5节 字符数组与字符

目录

一、C风格字符串

二、C++引入的string类

三、相关函数总结

C风格字符串

1.字符数组

```
#include<cstdio>
using namespace std;
int main(){
    char  str[]="I love china";
    printf(“%s\n”,str);
    return 0;
}
```

2.字符指针

```
#include<cstdio>
using namespace std;
int main(){
    char  *str="I love china";
    printf("%s\n",str);
    return 0;
}
```

C++引入的string类

C++ 标准库提供了 string 类类型，支持上述所有 C 风格字符串操作。

```
#include <iostream>
#include <string>
using namespace std;
int main () {
    string s1 = "He1lo";
    string s2 = "wor1d";
    string s3;
    int len;
    s3 = s1;//复制 s1 到 s3
    cout << "s3 : " << s3 << endl;
    s3 = s1 +s2;//连接 s1 和 s2
    cout << "s1 +s2 : " << s3 << endl;
    len = s3.size();//连接后 s3 的总长度
    cout << "s3.size() : " << len << endl;
    return 0;
}
```

字符数组相关函数总结

函数	全拼	头文件	功能	实例
memset	memory set	cstring	内存填充	memset(a,'b',sizeof(a))
gets	get string	cstdio	得到字符数组，允许字符数组里面有空格	gets(str)
puts	put string	cstdio	输出字符数组	puts(str)
atoi	ascii to integer	cstdlib	字符数组转数字	atoi(str)
strcat	String Catenate	cstring	连接字符串	strcat(d,s);
strcpy	string copy	cstring	复制字符串	strcpy(char* des,const char* source)
memcpy	memory copy	cstring	复制字节	memcpy(d,s,(strlen(s)+1));
strcmp	string compare	cstring	比较字符串	strcmp(s1,s2);
strlen	string length	cstring	计算字符串长度	strlen(str);
strstr	string string	cstring	子串在母串的位置	strstr(s,sub);
strncpy	string n copy	cstring	求子串	strncpy(char d [], const char s [], int numchars);
strncat	String n Catenate	cstring	结尾追加 n 个字符	strncat(char d [], const char s [], int numchars);

字符串（stl容器）相关函数总结

函数	全拼	头文件	功能	实例
empty	empty	string	判断字符串是否为空	s.empty()
getline	Get line	string	带空格的字符串输入	getline(cin,s)
size	Size	string	求字符串的长度	s.size()
insert	insert	string	在母串某个位置插入子串	s.insert(pos,s2);
substr	substring	string	求子串	s.substr(pos,len)
erase	erase	string	删除特定长度字符	s.erase(pos,len)
find	find	string	查找子串的位置	s.find(sub)
replace	replace	string	替换子串	s.replace(pos,len,news)
c_str		string	获取 C 语言风格的字符数组首地址指针	s.c_str()

【课堂练习】

课堂练习

1.【NOIP2008】设字符串S=”Olympic”，S的非空子串的数目是（）。

A．28　　B．29　　C．16　　D．17

2.【NOIP2012】原字符串中任意一段连续的字符组成的新字符串称为子串。则字符串“AAABBBCCC”共有(　　)个不同的非空子串。

A.3　　B.12　　C.36　　D.45

3.【NOIP2017】若串 S = “copyright”，其子串的个数是（）。

A．72　　B．45　　C．46　　D．36

4.【NOIP2016】以下关于字符串的判定语句中正确的是(　　)。

A．字符串是一种特殊的线性表　B．串的长度必须大于零

C．字符串不可以用数组来表示　D．空格字符组成的串就是空串

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第6节 链表

目录

一、顺序表

二、链表

顺序表

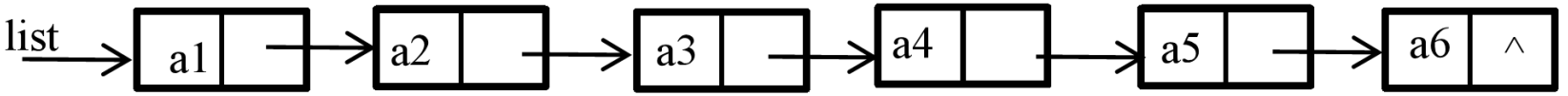
用一组地址连续的存储单元依次存储线性表中的数据元素，此时线性表是顺序表，数据元素间的逻辑关系通过元素下标反映出来。

0	1	…	i-1	i	….	n-1	…	maxsize-1
a_1	a_2	…	a_{i-1}	a_i	a_{i+1}	…	a_n	…	

1. 地址计算：$Loc(a_i)=Loc(a_1)+(i-1)*k$　　$Loc(a_{i+1})=Loc(a_i)+k$
2. 特点：逻辑上相邻的元素在物理位置上也相邻。
3. 优点：只需存放数据元素自身的信息，存储密度大，空间利用率高，存取速度快。
4. 缺点：需事先分配存储空间，容易造成空间浪费，插入删除操作时效率低。

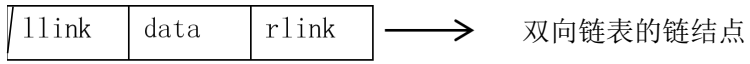
单向链表/线性链表

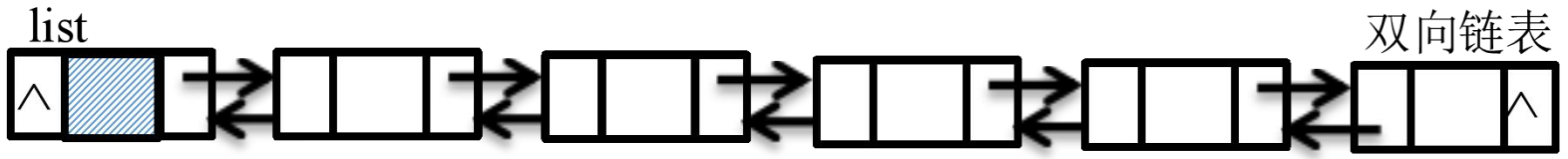
此链表中每个节点链结点由两部分构成：元素自身信息即数据域(用 data 表示)、指示其直接后继元素位置的信息即指针域(用 link 表示)。整个链表由一个称为外指针/头结点指针 list 指出，以表明链表的首地址，当链表为空时，list 为 null。用线性链表存储线性表时，数据元素间的逻辑关系通过指针反映出来。



06 双向链表

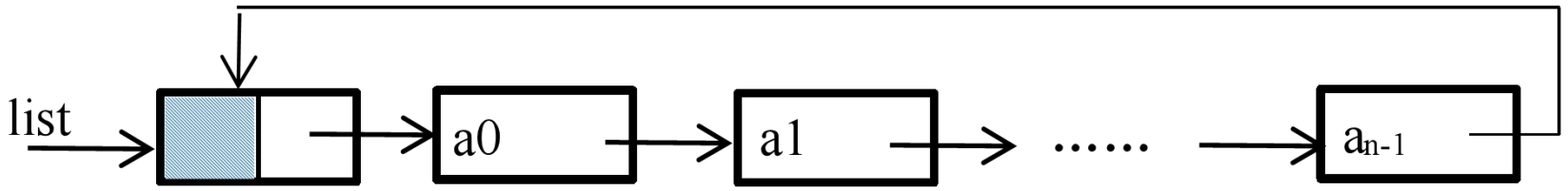
双向链表的每个链结点除了数据域 data 外设置两个指针域，一个 llink 指向直接前驱结点，一个 rlink 指向直接后继结点。双向链表有循环线性和非循环线性的，也可根据需要在链表前设置头结点 list。





循环链表

链表的最后一个链结点的指针指向链表的第 1 个链结点，整个链表形成一个环，从表中任意节点出发均可找到表中其他节点。



【课堂练习】

课堂练习

1.【NOIP2014】链表不具有的特点是(　　)。

A.不必事先估计存储空间　　B.可随机访问任一元素

C.插入删除不需要移动元素　　D.所需空间与线性表长度成正比

2.【NOIP2015】链表不具备的特点是(　　)。

A.可随机访问任何一个元素　　B.插入、删除操作不需要移动元素

C.无需事先估计存储空间大小　　D.所需存储空间与存储元素个数成正比

3.【NOIP2015】线性表若采用链表存储结构，要求内存中可用存储单元地址(　　)。

A.必须连续　　B.部分地址必须连续

C.一定不连续　　D.连续不连续均可

06 课堂练习

4.【NOIP2011】在含有 n 个元素的双向链表中查询是否存在关键字为 k 的元素，最坏情况下运行的时间复杂度是(　　)。

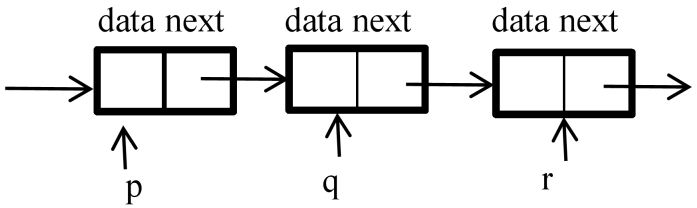
A.O(1)　　B.O(log n)　　C.O(n)　　D.O(n log n)

5.【NOIP2014】对长度为 n 的有序单链表，若检索每个元素的概率相等，则顺序检索到表中任一元素的平均检索长度为(　　)。

A.n/2　　B.(n+1)/2　　C.(n-1)/2　　D.n/4

6.【NOIP2014】有以下结构体说明和变量定义，如图所示，指针 p、q、r 分别指向一个链表中的三个连续结点。

```
struct node {
    int data;
    node*next;
}*p,*q,*r;
```



现要将 q 和 r 所指结点的先后位置交换，同时要保持链表的连续，以下程序段中错误的是(　　)。

A.q->next=r->next;p->next=r;r->next=q;

B.p->next=r;q->next=r->next;r->next=q;

C.q->next=r->next;r->next=q;p->next=r;

D.r->next=q;q->next=r->next;p->next=r;

课堂练习

7.【NOIP2010】双向链表中有两个指针域 llink 和 rlink,分别指向该结点的前驱及后继。设 p 指向链表中的一个结点，它的左右结点均非空。现要求删除结点 p,则下面语句序列中错误的是( )。

A.p->rlink->llink=p->rlink; p->llink->rlink=p->llink; delete p;

B.p->llink->rlink=p->rlink; p->rlink->llink=p->llink; delete p;

C.p->rlink->llink=p->llink; p->rlink->llink->rlink=p->rlink; delete p;

D.p->llink->rlink=p->rlink; p->llink->rlink->llink=p->llink; delete p;

8.【NOIP2015】双向链表中有两个指针域，llink 和 rlink，分别指回前驱及后继，设 p 指向链表中的一个结点，q 指向一待插入结点，现要求在 p 前插入 q，则正确的插入为(　　)。

A.
```
p->llink = q;q->rlink = p;
p->llink->rlink = q;q->llink = p->llink;
```

B.
```
q->llink = p->llink;p->llink->rlink = q;
q->rlink = p;p->llink = q->rlink;
```

C.
```
q->rlink = p;p->rlink = q;
p->llink->rlink = q;q->rlink = p;
```

D.
```
p->llink->rlink = q;q->rlink = p;
q->llink = p->llink;p->llink = q;
```

【不定项选择题】

课堂练习

1.【NOIP2009】在带尾指针（链表指针 clist 指向尾结点）的非空循环单链表中每个结点都以 next 字段的指针指向下一个节点。假定其中已经有 2 个以上的结点。下面哪些说法是正确的：

A. 如果 p 指向一个待插入的新结点，在头部插入一个元素的语句序列为：
```
p->next = clist->next; clist->next = p;
```
B. 如果 p 指向一个待插入的新结点，在尾部插入一个元素的语句序列为：
```
p->next = clist; clist->next = p;
```
C. 在头部删除一个结点的语句序列为：
```
p = clist->next; clist->next = clist->next->next; delete p;
```
D. 在尾部删除一个结点的语句序列为：
```
p = clist; clist = clist ->next; delete p;
```

2.【NOIP2010】双向链表中有两个指针域 llink 和 rlink，分别指向该结点的前驱及后继。设 p 指向链表中的一个结点，他的左右结点均为非空。现要求删除结点 p，则下列语句序列中正确的是(　　)。

```
A.p->rlink->llink=p->rlink;p->llink->rlink=p->llink; delete p;
B.p->llink->rlink=p->rlink;p->rlink->llink = p->llink; delete p;
C.p->rlink->llink = p->llink;p->rlink->llink->rlink= p->rlink;delete p;
D.p->llink->rlink = p->rlink;p->llink->rlink->link = p->llink;delete p;
```

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第7节 栈和队列

目录

一、栈

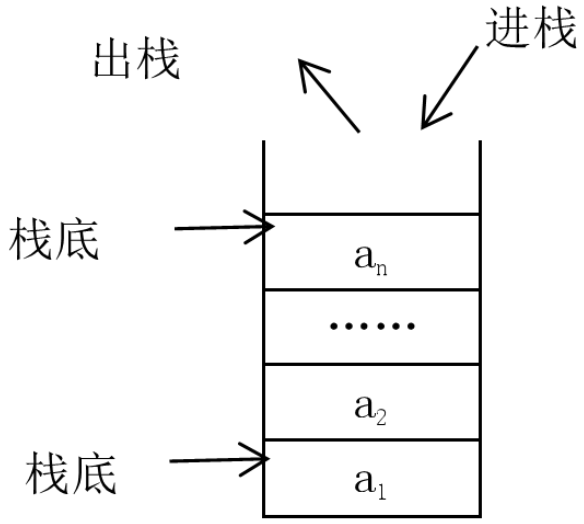
二、队列

栈

栈是只能在某一端插入和删除的特殊线性表。

用桶堆积物品，先堆进来的压在底下，随后一件一件往上堆。取走时，只能从上面一件一件取。堆和取都在顶部进行，底部一般是不动的。栈就是一种类似桶堆积物品的数据结构，进行删除和插入的一端称栈顶，另一堆称栈底。插入一般称为进栈（PUSH），删除则称为退栈（POP）。栈也称为后进先出表（LIFO 表）。

一个栈可以用定长为 n 的数组 s 来表示，用一个栈指针 top 指向栈顶。若 top＝0，表示栈空，top=n 时栈满。进栈时 top 加 1 。退栈时 top 减 1 。当 top<0 时为下溢。栈指针在运算中永远指向栈顶。



进栈（PUSH）、退栈（POP）算法

1. 进栈（PUSH）算法

①若 top≥ n 时，则给出溢出信息，作出错处理（进栈前首先检查栈是否已满，满则溢出；不满则作②）；

②top++（栈指针加 1 ，指向进栈地址）；

③s[top]=x，结束（x 为新进栈的元素）；

2. 退栈（POP）算法

①若 top≤0，则给出下溢信息，作出错处理(退栈前先检查是否已为空栈，空则下溢；不空则作②)；

②x=s[top]，（退栈后的元素赋给 x）；

③top--，结束（栈指针减 1 ，指向栈顶）。

进栈、出栈的 c++实现过程程序：

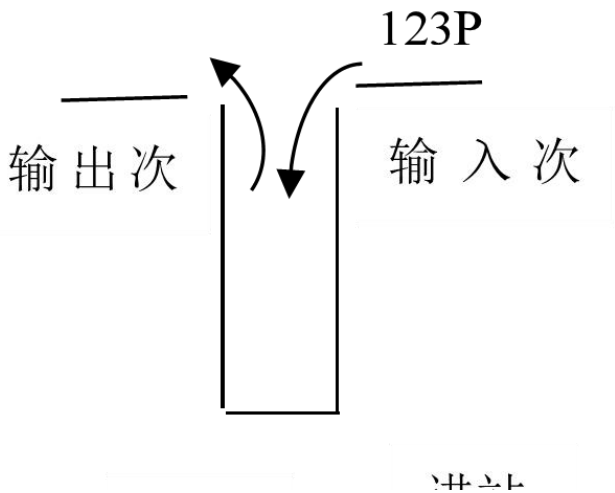
07 例题1

例题 1．设输入元素为 1、2、3、P 和 A，输入次序为 123PA，如图所示。元素经过栈后到达输出序列，当所有元素均到达输出序列后，有哪些序列可以作为高级语言的变量名？

【解答】

高级语言变量名的定义规则：以字母开头，字母与数字的组合。

由于必须以字母开头，所以，第一个可能出现的字母是 P，那么必然要求 123 已经先后入栈，这样便可得到以 P 开头的、并且 123 按逆序排列的（即 321）、同时 A 位于 P 后任一位置的变量名序列。此外，还需要考虑以 A 开头的可能情况，这只有一种情形：AP321 所以 AP321，PA321，P3A21，P32A1，P321A 可以作为高级语言的变量名。

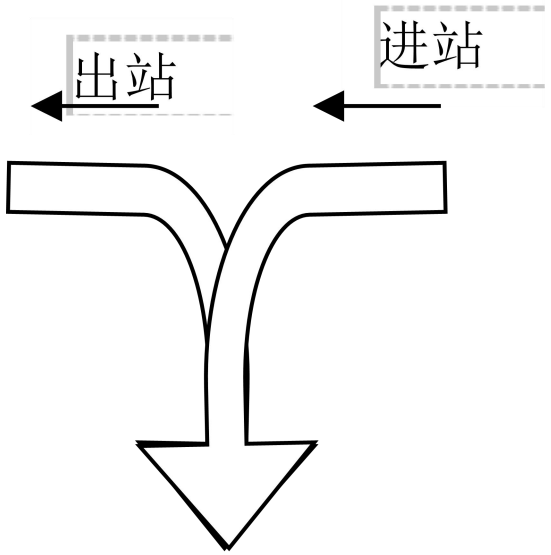


例题2

例题 2. 火车站列车调度示意图如下，假设调度站两侧的轨道为单向行驶轨道。

①如果进站的车厢序列为 123，则可能的出站车厢序列是什么？

②如果进站的车厢序列为 123456，问能否得到 135426 和 435612 的出站序列。



队列

队列是限定在一端进行插入，另一端进行删除的特殊线性表。就像排队买东西，排在前面的人买完东西后离开队伍（删除），而后来的人总是排在队伍末尾（插入）。通常把队列的删除和插入分别称为出队和入队。允许出队的一端称为队头，允许入队的一端称为队尾。所有需要进队的数据项，只能从队尾进入，队列中的数据项只能从队头离去。由于总是先入队的元素先出队（先排队的人先买完东西），这种表也称为先进先出（FIFO）表。

队列可以用数组 Q[m+1]来存储，数组的上界m即是队列所容许的最大容量。在队列的运算中需设两个指针：

head：队头指针，指向实际队头元素的前一个位置

tail：队尾指针，指向实际队尾元素所在的位置

一般情况下，两个指针的初值设为 0 ，这时队列为空，没有元素。图 1 (a)画出了一个由 6 个元素构成的队列，数组定义 Q[11]。

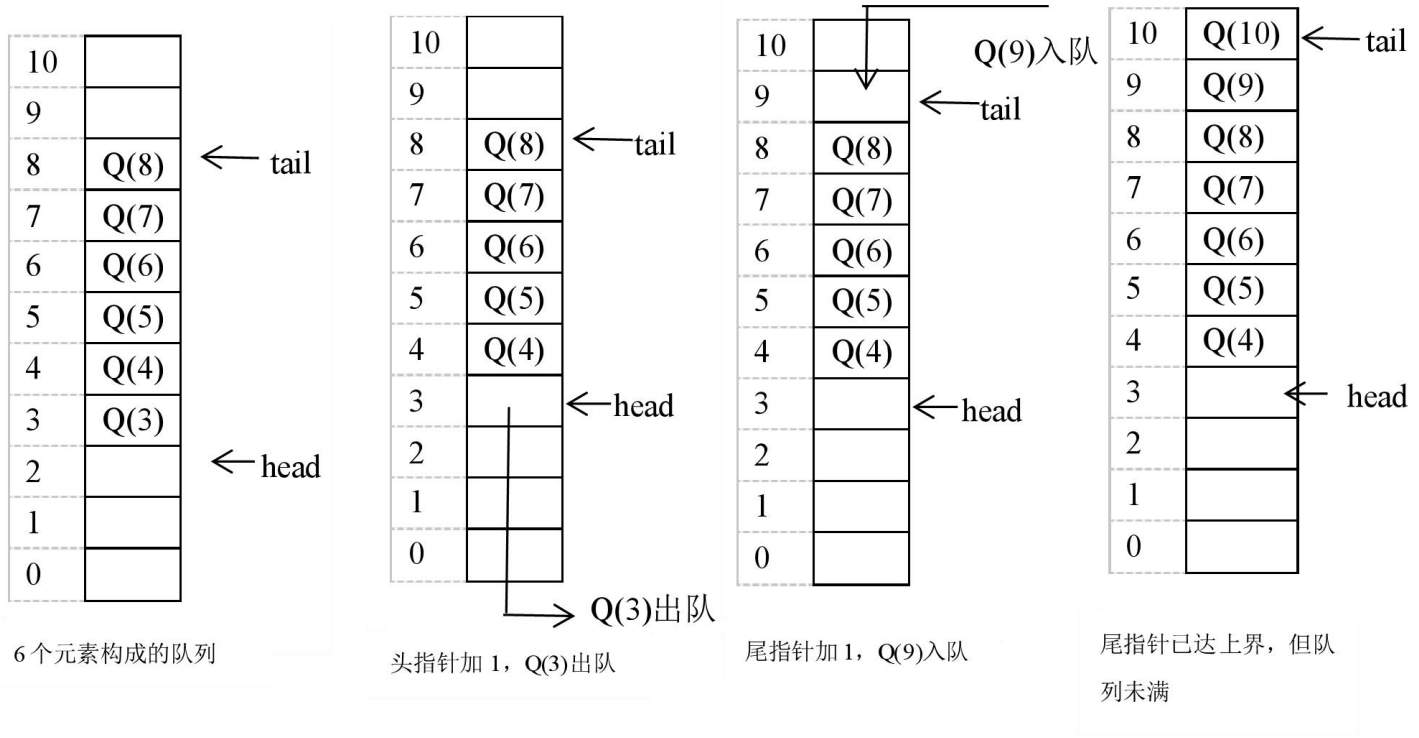
Q(i) i=3,4,5,6,7,8 头指针 head＝2，尾指针 tail＝8。

07 队列

队列中拥有的元素个数为:L=tail-head 现要让排头的元素出队，则需将头指针加 1，即 head++。这时头指针向上移动一个位置，指向 Q(3)，表示 Q(3)已出队。见图 1 (b)。

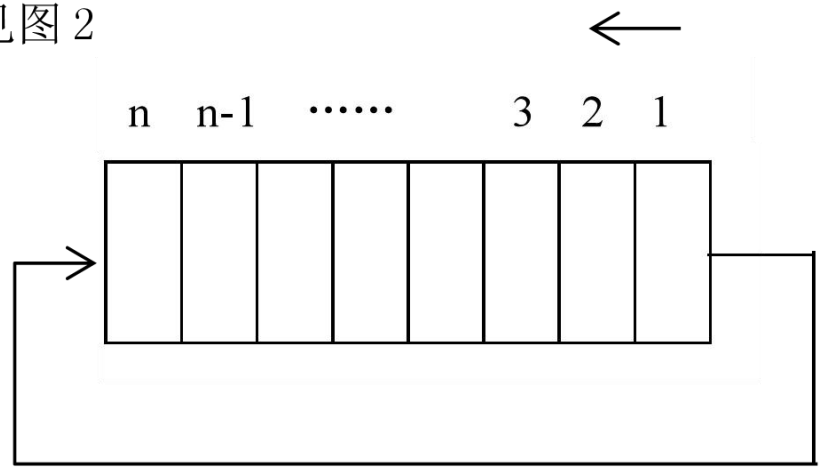
如果想让一个新元素入队，则需尾指针向上移动一个位置。即 tail++这时 Q(9)入队，见图 1 (c)。

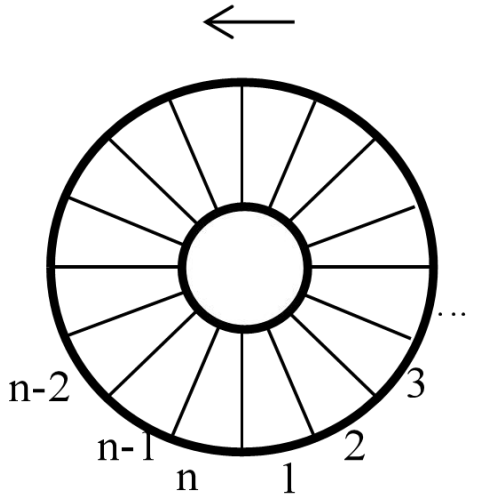
当队尾已经处理在最上面时，即 tail=10，见图 1 (d)，如果还要执行入队操作，则要发生"上溢"，但实际上队列中还有三个空位置，所以这种溢出称为"假溢出"。



07 课堂练习

克服假溢出的方法有两种。一种是将队列中的所有元素均向低地址区移动，显然这种方法是很浪费时间的；另一种方法是将数组存储区看成是一个首尾相接的环形区域。当存放到 n 地址后，下一个地址就"翻转"为 1 。在结构上采用这种技巧来存储的队列称为循环队列，见图 2





循环队的入队算法如下：

1、tail++;

2、若 tail=n+1，则 tail=1;

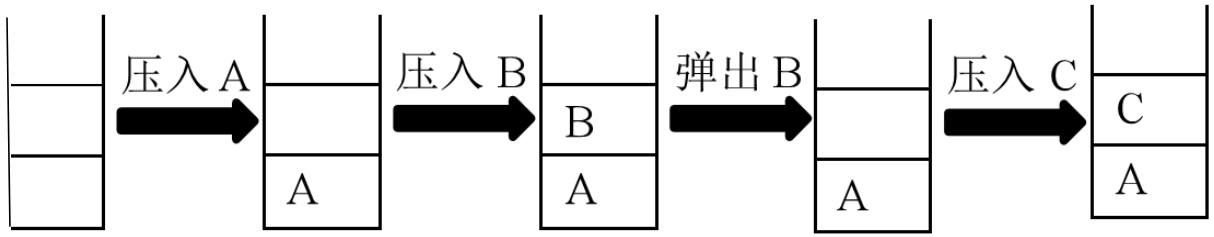
3、若 head=tail 尾指针与头指针重合了，表示元素已装满队列，则作上溢出错处理；

4、否则，Q(tail)=x，结束（x 为新入队元素）。

【课堂练习】

07 课堂练习

1.【NOIP2013】【NOIP2018】下图中所使用的数据结构是(　　)。



A.哈希表　　B.栈　　C.队列　　D.二叉树

2.【NOIP2008】设栈 S 的初始状态为空，元素 a，b，c，d，e，f 依次入栈 S，出栈的序列为 b，d，f，e，c，a，则栈 S 的容量至少应该是（）。

A．6　　B．5　　C．4　　D．3

3.【NOIP2009】有六个元素 FEDCBA 从左至右依次顺序进栈，在进栈过程中会有元素被弹出栈。问下列哪一个**不可能**是合法的出栈序列？

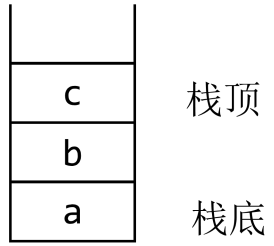
A.EDCFAB　　B.DECABF　　C.CDFEBA　　D．BCDAEF

07 课堂练习

4.【NOIP2010】元素 R1、R2、R3、R4、R5 入栈的顺序为 R1、R2、R3、R4、R5。如果第一个出栈的是 R3，那么第 5 个出栈的不可能是(　)。

A.R1　　B.R2　　C.R4　　D.R5

5.【NOIP2012】如果一个栈初始时为空，且当前栈中的元素从栈底到栈顶依次为 a，b，c(如右图所示)，另有元素 d 已经出栈，则可能的入栈顺序是(　　)。



A.a，d，c，b　B.b，a，c，d　C.a，c，b，d　D.d，a，b，c

6.【NOIP2015】今有一空栈 S，对下列待进栈的数据元素序列 a,b,c,d,e,f 依次进行进栈，进栈，出栈，进栈，进栈，出栈的操作，则此操作完成后，栈 S 的栈顶元素为(　　)。

A. f　　B. c　　C. a　　D. b

课堂练习

7.【NOIP2017】表达式 a * (b + c) * d 的后缀形式是（ ）。

A. a b c d * + *　　B. a b c + * d *

C. a * b c + * d　　D. b + c * a * d

8.【NOIP2017】 对于入栈顺序为 a，b，c，d，e，f，g 的序列，下列（）不可能是合法的出栈序列。

A. a，b，c，d，e，f，g　　B. a，d，c，b，e，g，f

C. a，d，b，c，g，f，e　　D. g，f，e，d，c，b，a

9.【NOIP2011】广度优先搜索时，需要用到的数据结构是( )。

A.链表　　B.队列　　C.栈　　D.散列表

课堂练习

10.【NOIP2017】 向一个栈顶指针为 hs 的链式栈中插入一个指针 s 指向的结点时，应执行（）。

A. hs->next = s;　　B. s->next = hs; hs = s;

C. s->next = hs->next; hs->next = s;

D. s->next = hs; hs = hs->next;

11.【NOIP2012】(　　)是一种先进先出的线性表。

A.栈　　B.队列　　C.哈希表(散列表)　　D.二叉树

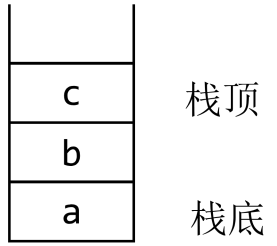
【不定项选择题】

07 课堂练习

1.【NOIP2010】元素 R1、R2、R3、R4、R5 入栈的顺序为 R1、R2、R3、R4、R5。如果第 1 个出栈的是 R3，那么第 5 个出栈的可能是(　　)。

A.R1　　B.R2　　C.R4　　D.R5

2.【NOIP2012】如果一个栈初始时为空，且当前栈中的元素从栈底到栈顶依次为 a，b，c(如右图所示)，另有元素 d 已经出栈，则可能的入栈顺序有(　　)。



A.a，b，c，d　　B.b，a，c，d

C.a，c，b，d　　D.d，a，b，c

3.【NOIP2017】对于入栈顺序为 a，b，c，d，e，f，g 的序列，下列(　　)不可能是合法的出栈序列。

A. a，b，c，d，e，f，g　　C. a，d，b，c，g，f，e

B. a，d，c，b，e，g，f　　D. g，f，e，d，c，b，a

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第8节 树

目录

一、树的定义

二、树的相关概念

三、树的性质

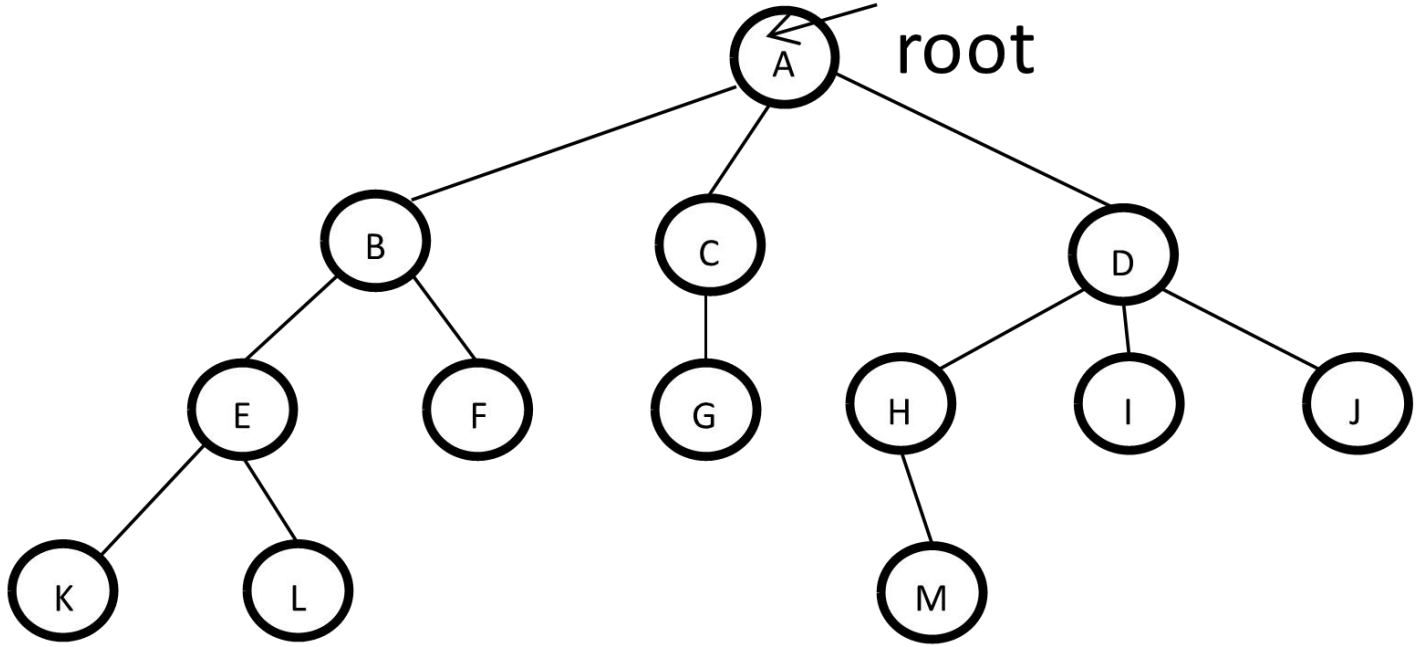
四、二叉树

树的定义

树是一种非线性的数据结构，是由 n（n >=0）个结点组成的有限集合。

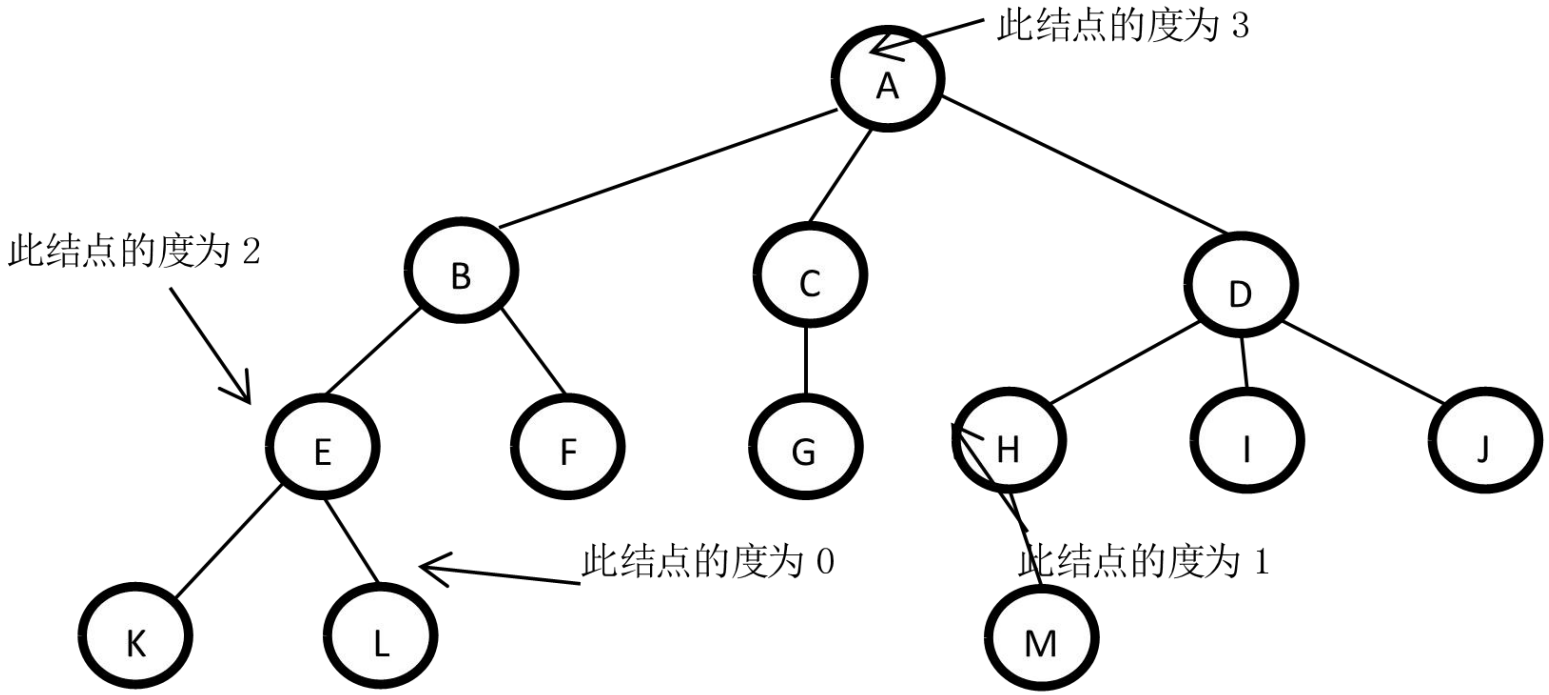
1. 如果 n==0，树为空树；
2. 如果 n>0，树有一个特定的结点—根结点。

根结点只有直接后继，没有直接前驱。除根结点以外的其他结点划分为 m（m>=0）个互不相交的有限集合 T0，T1，T2，...，Tm-1，每个集合是一棵树，称为根结点的子树。树的示例如下：



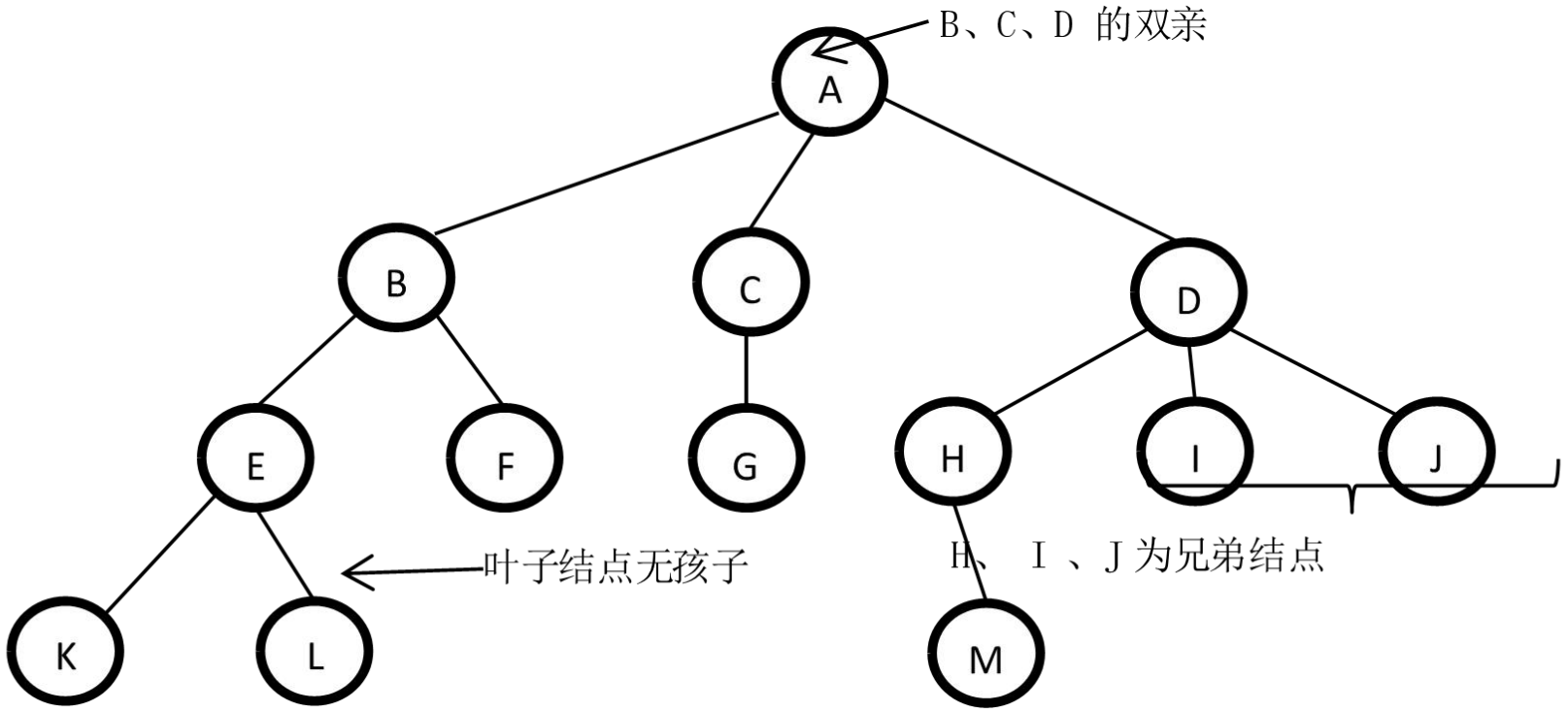
树的度

1. 树的度：结点拥有的子树的数量为结点的度，度为 0 的结点是叶结点，度不为 0 的结点为分支结点，树的度定义为树的所有结点中度的最大值。



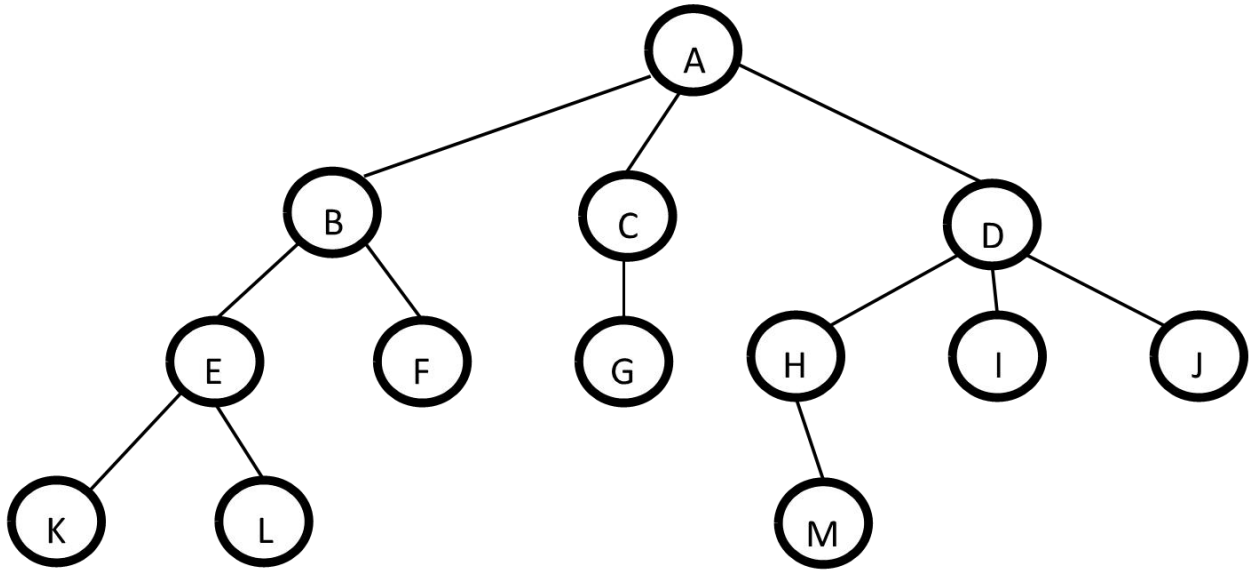
08 树的前驱和后继

2. 树的前驱和后继：除根节点没有前驱外，其余每个节点都有唯一的一个前驱节点。每个节点可以有 0 或多个后继节点。结点的直接后继称为结点的孩子，结点的直接前驱称为孩子的父亲。结点的孩子的孩子称为结点的孙子，结点称为子孙的祖先。同一个双亲的孩子之间互称兄弟。



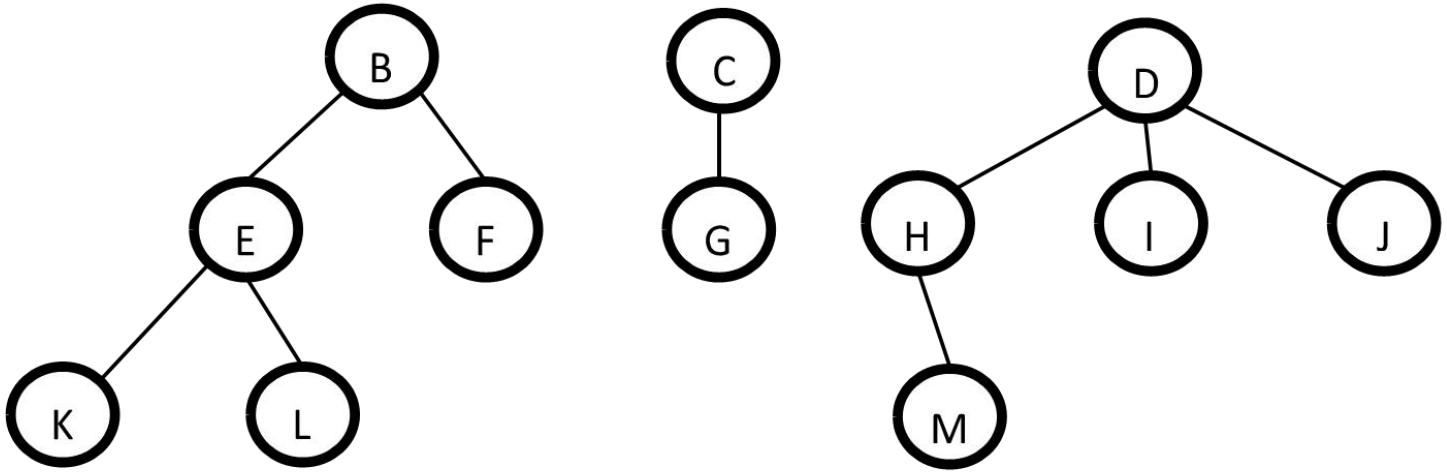
08 树中结点的层次

3. 树中结点的层次：树中根结点为第 1 层，根结点的孩子为第 2 层，依次类推。树中结点的最大层次称为树的深度或高度。



森林

4. 森林：是由 n(n>=0)棵互不相交的树组成的集合。



树的性质

1. 除根节点没有父节点外，其余节点有且仅有一个父节点。
2. n 个节点的树，有且仅有 n-1 条边。
3. 树中任意两个节点之间有且仅有一条简单路径（指路径上的顶点都不相同的路径，不存在自环和重边）。

二叉树

二叉树（binary tree,简写成 BT）是一种度数为 2 的树，即二叉树的每个节点最多有两个子节点。每个节点的子节点分别称为左孩子、右孩子，它的两棵子树分别称为左子树、右子树。

1.二叉树的遍历

所谓的遍历是指按一定的规律和次序访问树中的各个节点，而且每个节点仅被访问一次遍历一般按照从左到右的顺序，共有 4 种遍历方法：前序遍历、中序遍历、后序遍历、层次遍历。

(1)先序遍历：先访问根节点，再访问左子树，最后访问右子树。

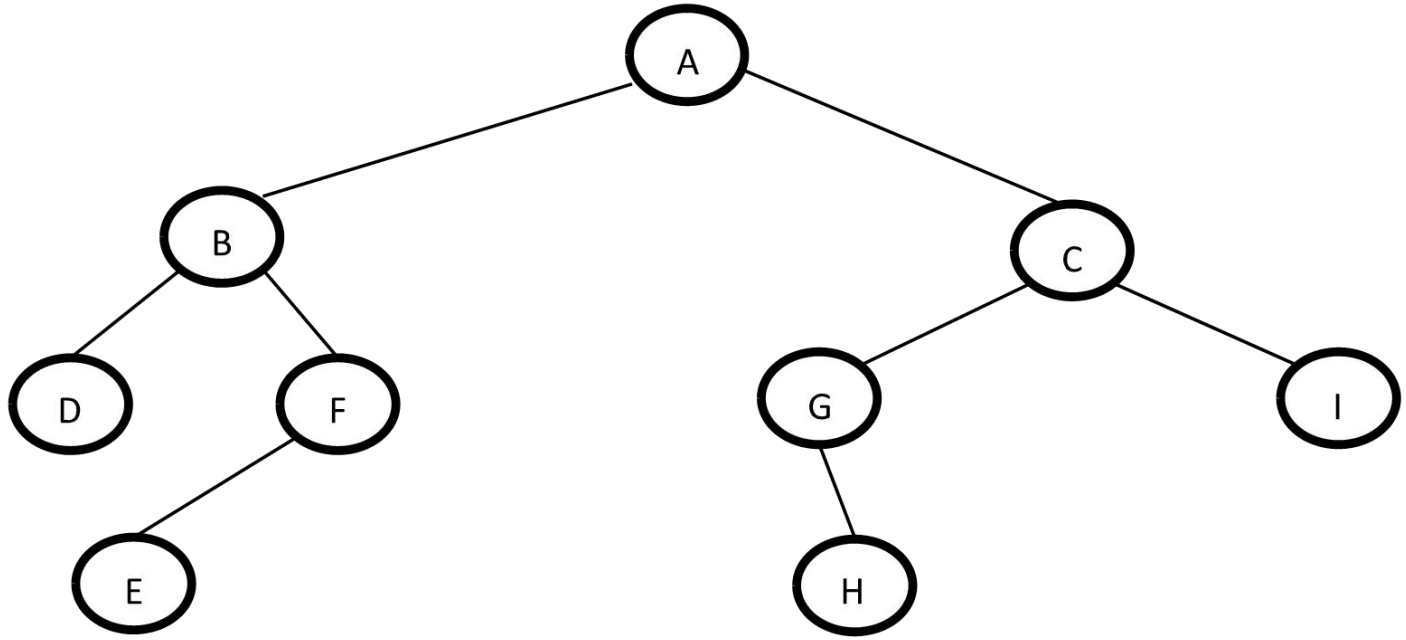
(2)后序遍历：先左子树，再右子树，最后根节点。

(3)中序遍历：先左子树，再根节点，最后右子树。

(4)层次遍历：每一层从左到右访问每一个节点。

例题

例 1：求下图所示树的各种遍历结果。



先序遍历结果：ABDFECGHI

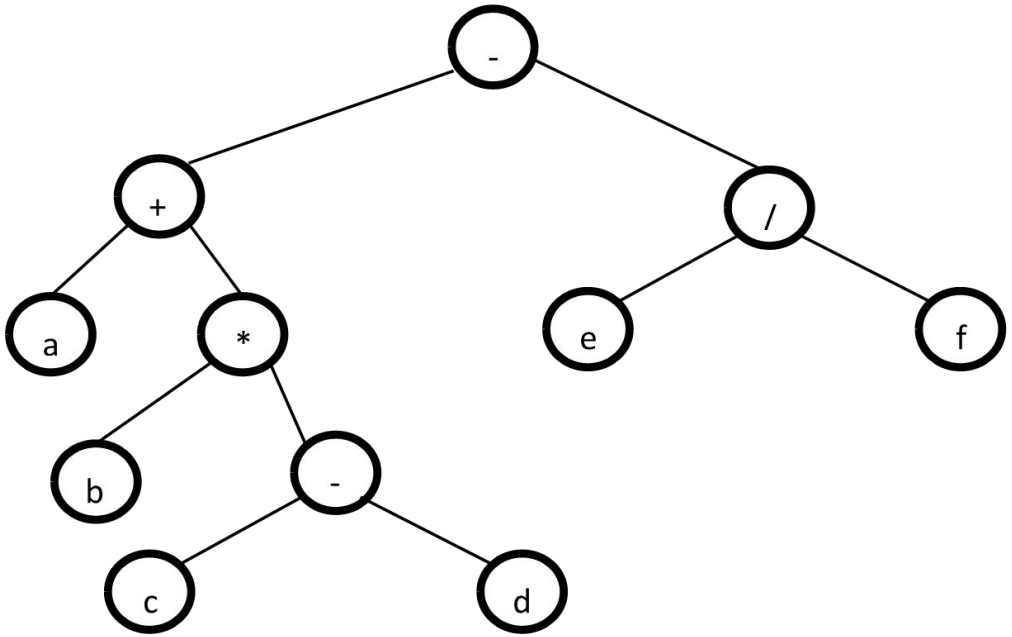
中序遍历结果：DBEFAGHCI

后序遍历结果：DEFBHGICA

层次遍历结果：ABCDFGIEH

例题

例 2：关于前面讲的表达式树，我们可以分别用先序、中序、后序的遍历方法得出完全不同的遍历结果，如对于**下图**遍历结果如下，它们正好对应着表达式的 3 种表示方法。



表达式（a+b*(c-d)-e/f）的二叉树

-+a*b-cd/ef　(前缀表示、波兰式)

a+b*c-d-e/f　(中缀表示)

abcd-*+ef/-　(后缀表示、逆波兰式)

结论：已知前序序列和中序序列可以确定出二叉树；已知中序序列和后序序列也可以确定出二叉树；但已知前序序列和后序序列却不可以确定出二叉树。

08 例题

例:3：有二叉树中序序列为A B C E F G H D，后序序列为：A B F H G E D C。请画出此二叉树，并求前序序列。

解答：根据后序序列知根节点为C

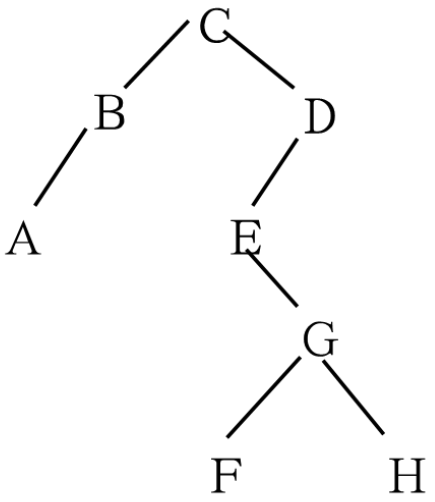
因此左子树：中序序列为A B

后序序列为A B

右子树：中序序列为E F G H D

后序序列为F H G E D

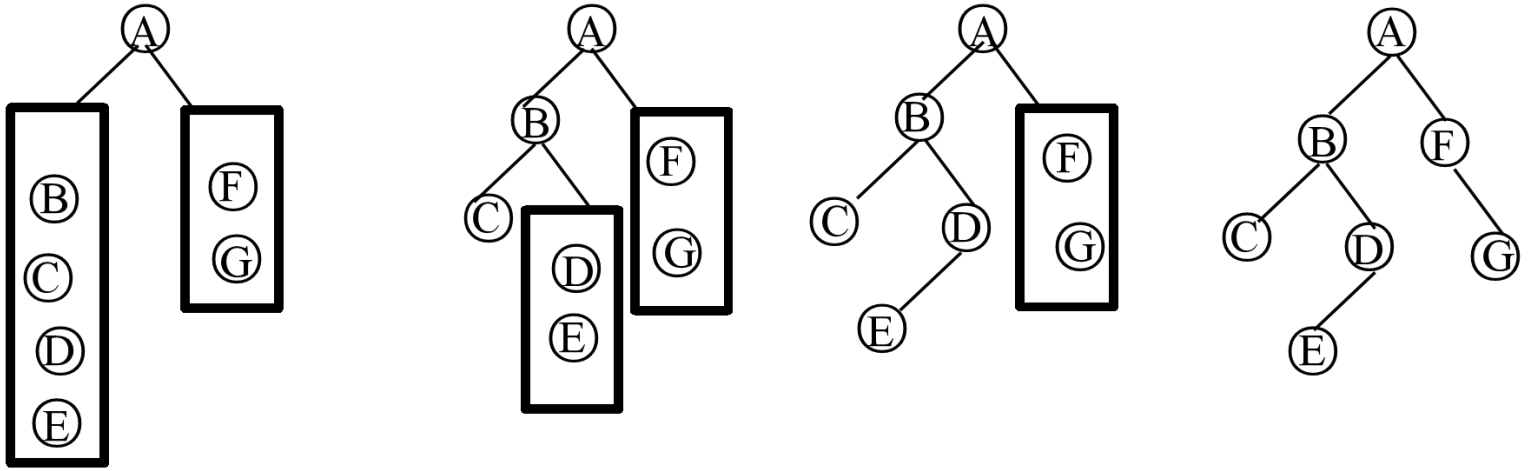
依次推得该二叉树的结构图。（如右图）



前序序列为：C B A D E G F H。

08 课堂练习

例 4：已知结点的前序序列为 ABCDEFG，中序序列为 CBEDAFG。构造出二叉树。过程见下图：



二叉树的性质

【性质 1】在二叉树的第 i 层上最多有 2^{i-1} 个结点（i>=1）。

证明：很简单，用归纳法：当 i=1 时，$2^{i-1}=1$ 显然成立；现在假设第 i-1 层时命题成立，即第 i-1 层上最多有 2^{i-2} 个结点。由于二叉树的每个结点的度最多为 2，故在第 i 层上的最大结点数为第 i-1 层的 2 倍，即 $2*2^{i-2}=2^{i-1}$。

二叉树的性质

【性质 2】深度为 k 的二叉树至多有 2^k-1 个结点（k>=1）。

证明：在具有相同深度的二叉树中，仅当每一层都含有最大结点数时，其树中结点数最多。因此利用性质 1 可得，深度为 k 的二叉树的结点数至多为：

$$2^0+2^1+\ldots+2^{k-1}=2^k-1$$

故命题正确。

特别：一棵深度为 k 且有 2^k-1 个结点的二叉树称为满二叉树。如下图 A 为深度为 4 的满二叉树，这种树的特点是每层上的结点数都是最大结点数。

可以对满二叉树的结点进行连续编号，约定编号从根结点起，自上而下，从左到右，由此引出完全二叉树的定义，深度为 k，有 n 个结点的二叉树当且仅当其每一个结点都与深度为 k 的满二叉树中编号从 1 到 n 的结点一一对应时，称为完全二叉树。

下图 B 就是一个深度为 4，结点数为 12 的完全二叉树。它有如下特征：叶结点只可能在层次最大的两层上出现；对任一结点，若其右分支下的子孙的最大层次为 m，则在其左分支下的子孙的最大层次必为 m 或 m+1。下图 C、D 不是完全二叉树，请大家思考为什么？

08 二叉树的性质

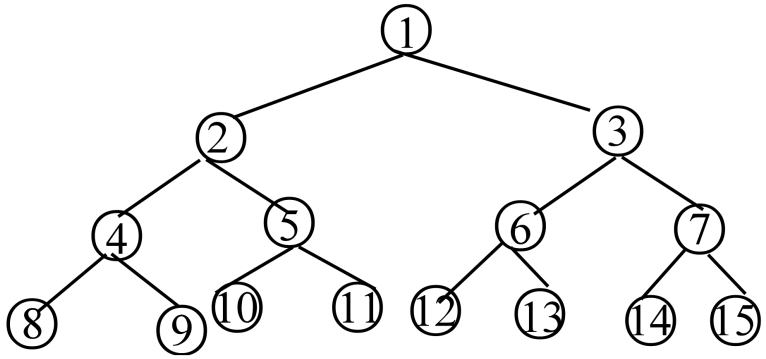


图 A

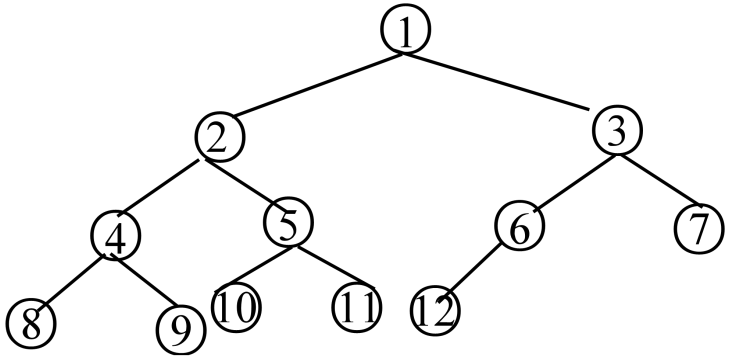


图 B

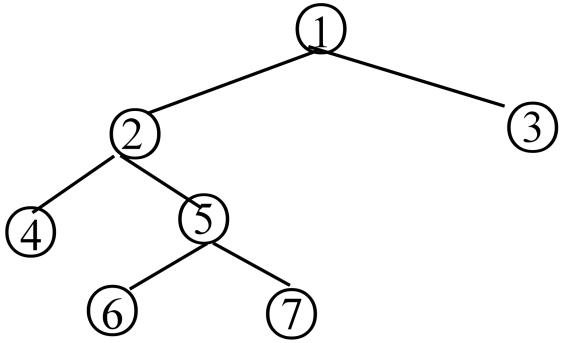


图 C

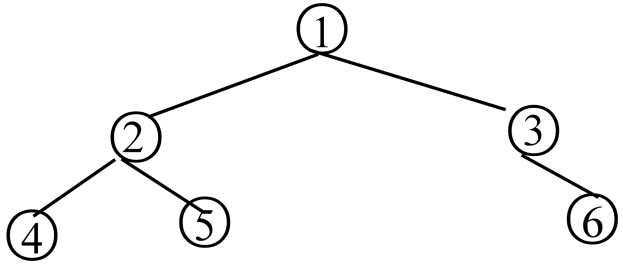


图 D

二叉树的性质

【性质 3】对任意一棵二叉树，如果其叶结点数为 n_0，度为 2 的结点数为 n_2，则一定满足：$n_0=n_2+1$。

证明：因为二叉树中所有结点的度数均不大于 2，所以结点总数(记为 n)应等于 0 度结点数 n_0、1 度结点 n_1 和 2 度结点数 n_2 之和：

$$n=n_o+n_1+n_2\ldots\ldots\text{(式子 1)}$$

另一方面，1 度结点有一个孩子，2 度结点有两个孩子，故二叉树中孩子结点总数是：

$$n_1+2n_2$$

树中只有根结点不是任何结点的孩子，故二叉树中的结点总数又可表示为：

$$n=n_1+2n_2+1\ldots\ldots\text{(式子 2)}$$

由式子 1 和式子 2 得到：

$$n_o=n_2+1$$

二叉树的性质

【性质 4】具有 n 个结点的完全二叉树的深度为 $floor(\log_2^n)+1$

证明：假设深度为 k，根据完全二叉树的定义，前面 k-1 层一定是满的，所以 $n>2^{k-1}-1$。但 n 又要满足 $n<=2^k-1$。所以，$2^{k-1}-1<n<=2^k-1$。变换一下为 $2^{k-1}<=n<2^k$。

以 2 为底取对数得到：$k-1<=\log_2^n<k$。而 k 是整数，所以 $k= floor(\log_2^n)+1$。

【性质 5】具有 n 个结点的二叉树的高度至少为 $\log_2 (n+1)$

证明：根据"性质 2"可知，高度为 k 的二叉树最多有 2k－1 个结点。反之，对于包含 n 个节点的二叉树的高度至少为 $\log_2(n+1)$。

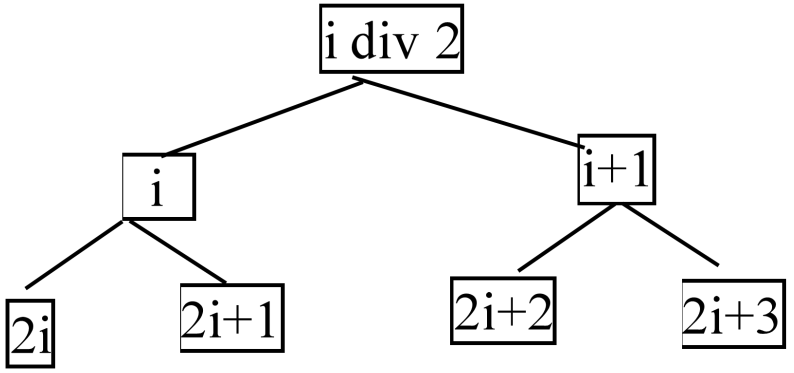
课堂练习

【性质 6】对于一棵 n 个结点的完全二叉树的任一个结点（编号为 i），有：

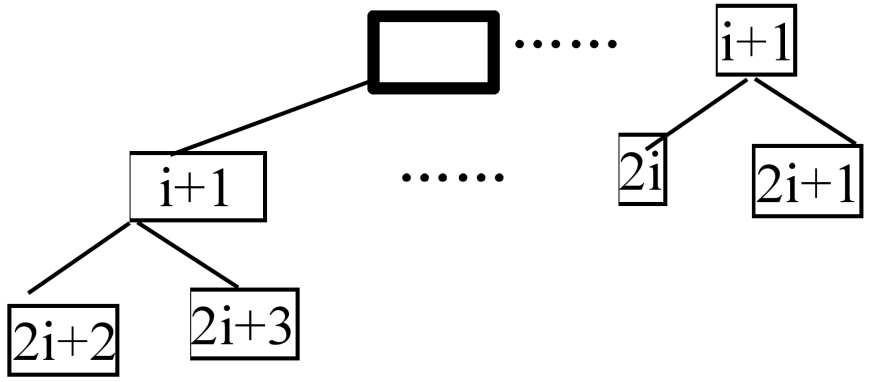
①如果 i=1,则结点 i 为根，无父结点；如果 i>1,则其父结点编号为 i/2。

②如果 2*i>n，则结点 i 无左孩子，否则左孩子编号为 2*i。如果 2*i+1>n，则结点 i 无右孩子，否则右孩子编号为 2*i+1。

证明：略，我们只要验证一下即可。总结如**图 3-5**:



结点 i 和 i+1 在同一层上



结点 i 和 i+1 不在同一层上

【课堂练习】

课堂练习

1.【NOIP2013】已知一棵二叉树有 10 个节点，则其中至多有(　　)个节点有 2 个子节点。

A.4　　B.5　　C.6　　D.7

2.【NOIP2013】已知一棵二叉树有 2013 个节点，则其中至多有(　　)个节点有 2 个子节点。

A.1006　　B.1007　　C.1023　　D.1024

3.【NOIP2011】如果根结点的深度记为 1，则一棵恰有 2011 个叶结点的二叉树的深度最少是(　　)。

A.10　　B.11　　C.12　　D.13

4.【NOIP2010】如果树根算是第一层，那么一颗 n 层的二叉树最多有(　)个结点。

A. 2^n-1　　B.2^n　　C.2^n+1　　D.2^{n+1}

课堂练习

5.【NOIP2009】一个包含 n 个分支结点（非叶结点）的非空二叉树，它的叶结点数目最多为：

A.2n + 1　　B.2n-1　　C.n-1　　D. n+1

6.【NOIP2009】最优前缀编码，也称 Huffman 编码。这种编码组合的特点是对于较频繁使用的元素给与较短的唯一编码，以提高通讯的效率。下面编码组合哪一组不是合法的前缀编码。

A.（00，01，10，11）　　B.（0，1，00，11）

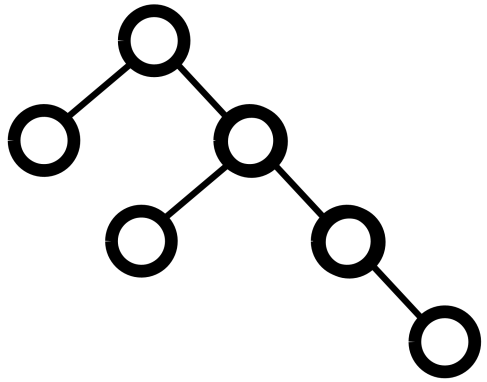
C.（0，10，110，111）　　D.（1，01，000，001）

7.【NOIP2011】现有一段文言文，要通过二进制哈夫曼编码进行压缩。简单起见，假设这段文言文只由 4 个汉字“之”、“乎”、“者”、“也”组成，它们出现的次数分别为 700、600、300、200。那么，“也”字的编码长度是(　　)。

A.1　　B.2　　C.3　　D.4

08 课堂练习

8.【NOIP2016】一棵二叉树如右图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点（根结点的下标为 1，若某结点的下标为 i ，则其左孩子位于下标 2i 处、右孩子位于下标(2i+1)处），则图中所有结点的最大下标为(　　)。



A．6　　B．10　　C．12　　D．15

9.【NOIP2016】一棵二叉树如右图所示，若采用二叉树链表存储该二叉树（各个结点包括结点的数据、左孩子指针、右孩子指针）。如果没有左孩子或者右孩子，则对应的为空指针。那么该链表中空指针的数目为（）。

A．6　　B．7　　C．12　　D．14

课堂练习

10.【NOIP2010】完全二叉树的顺序存储方案，是指将完全二叉树的结点从上到下、从左到右依次存放到一个顺序结构的数组中。假定根结点存放在数组的 1 号位置上，则第 k 号结点的父结点如果存在的话，应当存放在数组中的（）号位置。

A.2k　　B．2k+1　　C．k/2 下取整　　D．(k+1)/2

11.【NOIP2008】完全二叉树共有 2*N-1 个结点，则它的叶节点数是（）。

A．N-1　　B．N　　C．2*N　　D．2^N-1

12.【NOIP2009】一个包含 n 个分支结点（非叶结点）的非空满 k 叉树，k>=1，它的叶结点数目为：

A.nk + 1　　B.nk-1　　C.(k+1)n-1　　D.(k-1)n+1

13.【NOIP2015】如果根的高度为 1，具有 61 个结点的完全二叉树的高度为(　　)。

A.5　　B.6　　C.7　　D.8

08 课堂练习

14.【NOIP2014】一棵具有 5 层的满二叉树中结点数为(　　)。

A.31　　B.32　　C.33　　D.16

15.【NOIP2018】根节点深度为 0，一棵深度为 h 的满 k（k>1）叉树，即除最后一层无任何子 节点外，每一层上的所有结点都有 k 个子结点的树，共有（ ）个结点。

A. $(k^{h+1}-1)/(k-1)$　　B. k^{h-1}　　C. k^{h}　　D. $(k^{h-1})/(k-1)$

16.【NOIP2013】二叉树的(　　)第一个访问的节点是根节点。

A.先序遍历　　B.中序遍历　　C.后序遍历　　D.以上都是

17.【NOIP2013】二叉查找树具有如下性质：每个节点的值都大于其左子树上所有节点的值、小于其右子树上所有节点的值。那么，二叉查找树的(　　)是一个有序序列。

A.先序遍历　　B.中序遍历　　C.后序遍历　　D.宽度优先遍历

08 课堂练习

18.前序遍历序列与中序遍历序列相同的二叉树为(　　)。

A.根结点无左子树的二叉树

B.根结点无右子树的二叉树

C.只有根结点的二叉树或非叶子结点只有左子树的二叉树

D.只有根结点的二叉树或非叶子结点只有右子树的二叉树

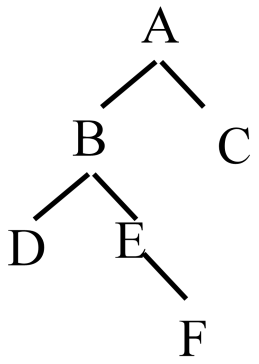
19.【NOIP2015】前序遍历序列与后序遍历序列相同的二叉树为(　　)。

A.非叶子结点只有左子树的二叉树　　B.只有根结点的二叉树

C.根结点无右子树的二叉树　　D.非叶子结点只有右子树的二叉树

20.【NOIP2011】右图是一棵二叉树，它的先序遍历是(　　)。

A. ABDEFC　　B. DBEFAC　　C. DFEBCA　　D. ABCDEF



08 课堂练习

21.【NOIP2012】如果一棵二叉树的中序遍历是 BAC，那么它的先序遍历不可能是(　　)。

A.ABC　　B.CBA　　C. ACB　　D.BAC

22.【NOIP2009】【NOIP2016】表达式 a*(b+c)-d 的后缀表达式是：

A.abcd*+-　　B.abc+*d-　　C.abc*+d-　　D.-+*abcd

23.【NOIP2018】表达式 a * d - b * c 的前缀形式是（ ）。

A.a d * b c * -　B.- * a d * b c　C.a * d - b * c　D.- * * a d b c

24.【NOIP2010】前缀表达式“+ 3 * 2 + 5 12”的值是( )。

A.23　　B.25　　C.37　　D.65

课堂练习

25.【NOIP2008】二叉树 T，已知其先根遍历是 1 2 4 3 5 7 6（数字为结点的编号，以下同），中根遍历是 2 4 1 5 7 3 6，则该二叉树的后根遍历是（）。

A. 4 2 5 7 6 3 1　　B. 4 2 7 5 6 3 1

C. 7 4 2 5 6 3 1　　D. 4 2 7 6 5 3 1

26.【NOIP2010】一棵二叉树的前序遍历序列是 ABCDEFG,后序遍历序列是 CBFEGDA,则根结点的左子树的结点个数可能是( )。

A.2　　B.3　　C.4　　D.5

【不定项选择题】

课堂练习

1.【NOIP2008】设 T 是一棵有 n 个顶点的树，下列说法正确的是(　　)。

A．T 是连通的、无环的　　B．T 是连通的，有 n-1 条边

C．T 是无环的，有 n-1 条边　　D．以上都不对

2.【NOIP2018】下列说法中，是树的性质的有(　　)。

A．无环　　B．任意两个结点之间有且只有一条简单路径

C．有且只有一个简单环　　D．边的数目恰是顶点数目减 1

3.【NOIP2015】下列有关树的叙述中，叙述正确的有(　　)。

A.在含有 n 个结点的树中，边数只能是(n-1)条

B.在哈夫曼树中，叶结点的个数比非叶结点个数多 1

C.完全二叉树一定是满二叉树

D.在二叉树的前序序列中，若结点 u 在结点 v 之前，则 u 一定是 v 的祖先

课堂练习

4.【NOIP2008】二叉树 T，已知其先根遍历是 1 2 4 3 5 7 6（数字为结点的编号，以下同），后根遍历是 4 2 7 5 6 3 1，则该二叉树的可能的中根遍历是(　　)。

A. 4 2 1 7 5 3 6　　B. 2 4 1 7 5 3 6

C. 4 2 1 7 5 6 3　　D. 2 4 1 5 7 3 6

5.【NOIP2011】如果根结点的深度记为 1，则一棵恰有 2011 个叶子结点的二叉树的深度可能是(　　)。

A. 10　　B. 11　　C. 12　　D. 2011

6.【NOIP2012】一棵二叉树一共有 19 个节点，其叶子节点可能有(　　)个。

A.1　　B.9　　C.10　　D.11

7.【NOIP2018】 2-3 树是一种特殊的树，它满足两个条件：

(1)每个内部结点有两个或三个子结点；

(2)所有的叶结点到根的路径长度相同。

如果一棵 2-3 树有 10 个叶结点，那么它可能有(　　)个非叶结点。

A. 5　　B. 6　　C. 7　　D. 8

《信息学奥赛一本通·初赛真题解析》

第二章　程序设计基本知识

第9节 图

目录

一、图的定义

二、图的相关概念

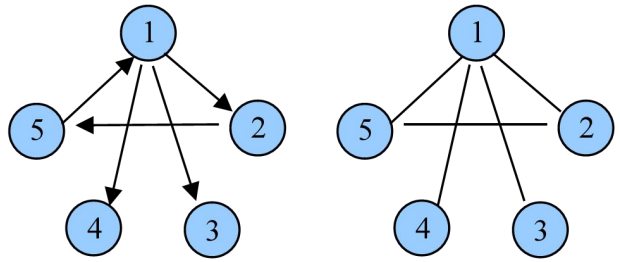
三、图的存储结构

图的定义

点用边连起来就叫做图，严格意义上讲，图是一种数据结构，定义为：graph=（V，E）。V是一个非空有限集合，代表顶点（结点），E代表边的集合。

09 图的相关概念

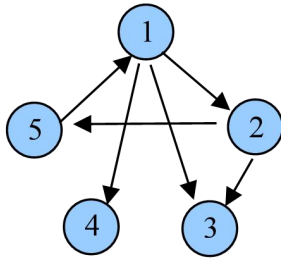
1.有向图：图的边有方向，只能按箭头方向从一点到另一点。(a)就是一个有向图。
2.无向图：图的边没有方向，可以双向。(b)就是一个无向图。



(a)

(b)

3.结点的度：无向图中与结点相连的边的数目，称为结点的度。
4.结点的入度：在有向图中，以这个结点为终点的有向边的数目。
5.结点的出度：在有向图中，以这个结点为起点的有向边的数目。
6.权值：边的“费用”，可以形象地理解为边的长度。
7.连通：如果图中结点U，V之间存在一条从U通过若干条边、点到达V的通路，则称U、V 是连通的。
8.回路：起点和终点相同的路径，称为回路，或“环”。
9.完全图：一个n 阶的完全无向图含有n*(n-1)/2 条边；一个n 阶的完全有向图含有n*(n-1)条边；
10.稠密图：一个边数接近完全图的图。
11.稀疏图：一个边数远远少于完全图的图。
12.强连通分量：有向图中任意两点都连通的最大子图。右图中，1-2-5构成一个强连通分量。特殊地，单个点也算一个强连通分量，所以右图有三个强连通分量：1-2-5，4，3。

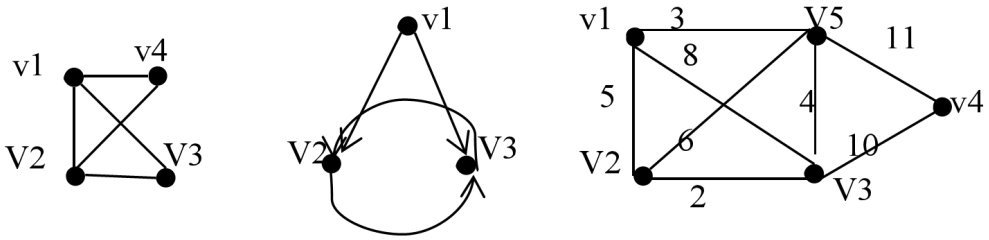


09 二维数组邻接矩阵存储

图的邻接矩阵存储方式是用一个二维数组（称邻接矩阵）存储图中的边或弧的信息。它适用于稠密图。

定义int G[101][101]，其中G[i][j]的值，表示从点i到点j的边的权值，定义如下：

$$G[i][j]=\begin{cases}1\text{ 或权值} & \text{当 } v_i \text{ 与 } v_j \text{ 之间有边或弧时，取值为 1 或权值}\\ 0\text{ 或}\infty & \text{当 } v_i \text{ 与 } v_j \text{ 之间无边或弧时，取值为 0 或}\infty\text{（无穷大）}\end{cases}$$

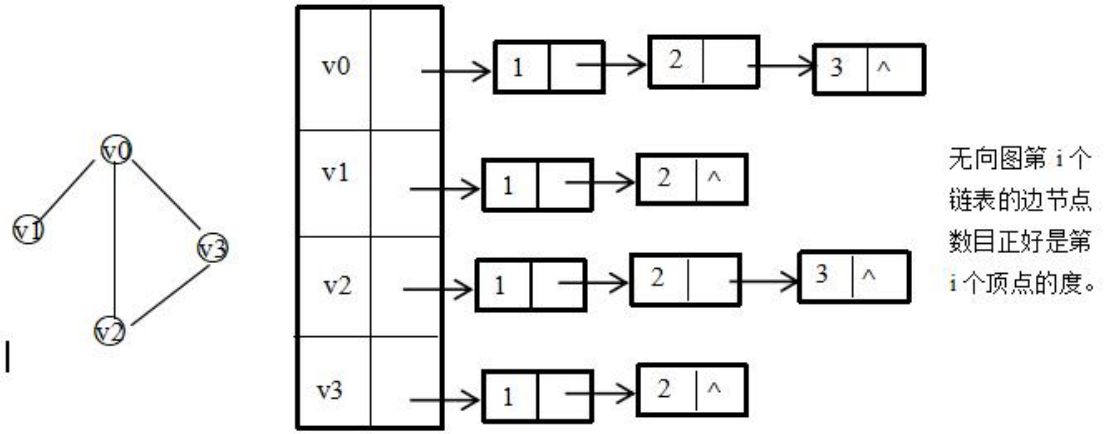


上图中的 3 个图对应的邻接矩阵分别如下：

$$G(A)=\begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&0\\1&1&0&0\end{bmatrix}\quad G(B)=\begin{bmatrix}0&1&1\\0&0&1\\0&1&0\end{bmatrix}\quad G(C)=\begin{bmatrix}\infty&5&8&\infty&3\\5&\infty&2&\infty&6\\8&2&\infty&10&4\\\infty&\infty&10&\infty&11\\3&6&4&11&\infty\end{bmatrix}$$

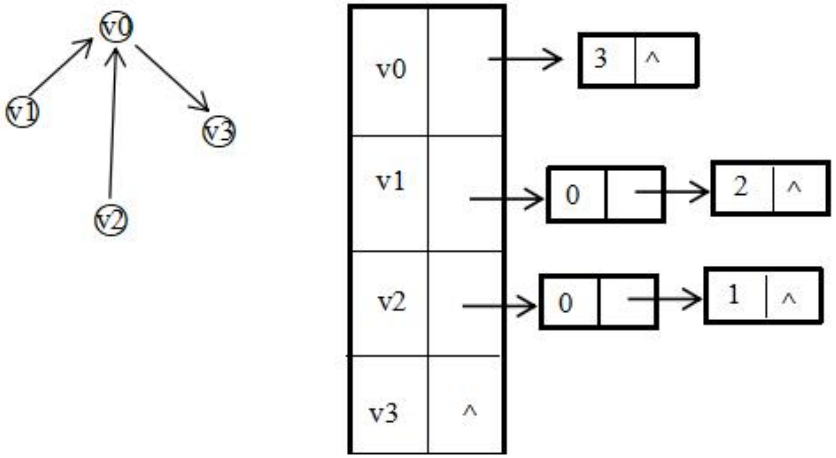
09 邻接表存储结构

邻接表是一种将数组与链表相结合的存储方法，其具体实现为：将图中顶点用一个一维数组存储，每个顶点 Vi 的所有邻接点用一个单链表来存储，链表中存放与当前节点相邻的节点在数组中的下标。它适用于稀疏图。



无向图第 i 个链表的边节点数目正好是第 i 个顶点的度。

对于具有 n 个节点、e 条边的无向图，邻接表中数组有 n 个顶点节点、链表有 2e 个边节点。



有向图第 i 个链表的边节点数目正好是第 i 个顶点的出度。

对于具有 n 个节点、e 条边的有向图，邻接表中数组有 n 个顶点节点、链表有 e 个边节点。

09 深度优先与广度优先遍历

深度优先遍历与深搜 dfs 相似，从一个点 A 出发，将这个点标为已访问 visited[i]=true，然后再访问所有与之相连，且未被访问过的点。当 A 的所有邻接点都被访问过后，再退回到 A 的上一个点（假设是 B），再从 B 的另一个未被访问的邻接点出发，继续遍历。

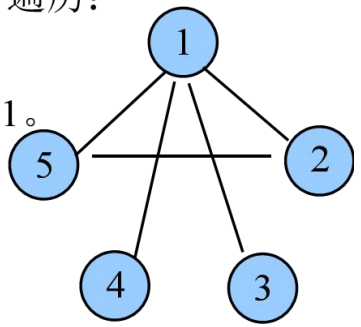
例如对右边的这个无向图深度优先遍历，假定先从 1 出发。程序以如下顺序遍历：

a．1→2→5，然后退回到 2，退回到 1。

b．从 1 开始再访问未被访问过的点 3 ，3 没有未访问的邻接点，退回 1。

c．再从 1 开始访问未被访问过的点 4，再退回 1 。

d．起点 1 的所有邻接点都已访问，遍历结束。



广度优先遍历并不常用，从编程复杂度的角度考虑，通常采用的是深度优先遍历。

广度优先遍历和广搜 bfs 相似，因此使用广度优先遍历一张图并不需要掌握什么新的知识，在原有的广度优先搜索的基础上，做一点小小的修改，就成了广度优先遍历算法。

一笔画问题

如果一个图存在一笔画，则一笔画的路径叫做**欧拉路**，如果最后又回到起点，那这个路径叫做**欧拉回路**。

定义奇点是指跟这个点相连的边数目有奇数个的点。对于能够一笔画的图，于是有以下两个定理。

定理 1：存在欧拉路的条件：图是连通的，有且只有 2 个奇点。

定理 2：存在欧拉回路的条件：图是连通的，有 0 个奇点。

两个定理的正确性是显而易见的，既然每条边都要经过一次，那么对于欧拉路，除了起点和终点外，每个点如果进入了一次，显然一定要出去一次，显然是偶点。对于欧拉回路，每个点进入和出去次数一定都是相等的，显然没有奇点。

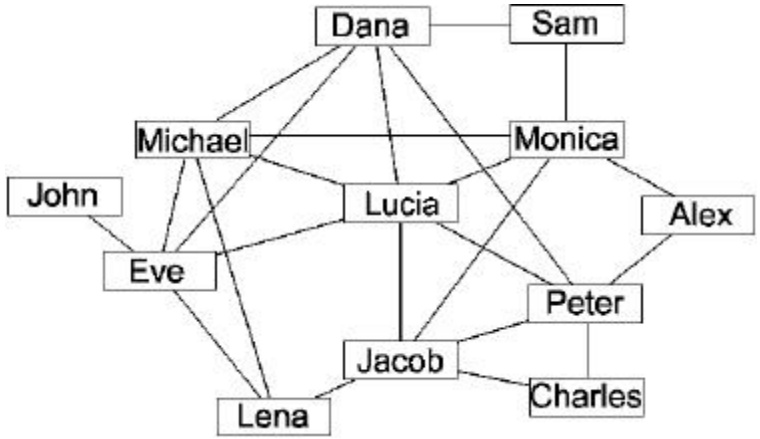
求欧拉路的算法很简单，使用深度优先遍历即可。

根据一笔画的两个定理，如果寻找欧拉回路，对任意一个点执行深度优先遍历；找欧拉路，则对一个奇点执行dfs，时间复杂度为O(m+n)，m为边数，n是点数。

【课堂练习】

09 课堂练习

1.【NOIP2016】Lucia 和她的朋友以及朋友的朋友都在某社交网站上注册了账号。下图是他们之间的关系图，两个人之间有边相连代表这两个人是朋友，没有边相连代表不是朋友。这个社交网站的规则是：如果某人 A 向他（她）的朋友 B 分享了某张照片，那么 B 就可以对该照片进行评论；如果 B 评论了该照片，那么他（她）的所有朋友都可以看见这个评论以及被评论的照片，但是不能对该照片进行评论（除非 A 也向他（她）分享了该照片）。现在 Lucia 已经上传了一张照片，但是她不想让 Jacob 看见这张照片，那么她可以向以下朋友(　　)分享该照片。



A. Dana, Michael, Eve　　B. Dana, Eve, Monica

C. Michael, Eve, Jacob　　D. Micheal, Peter, Monica

课堂练习

2.【NOIP2014】有向图中每个顶点的度等于该顶点的(　　)。

A.入度　　B.出度　　C.入度与出度之和　　D.入度与出度之差

3.【NOIP2014】在无向图中，所有顶点的度数之和是边数的(　　)倍。

A.0.5　　B.1　　C.2　　D.4

4.【NOIP2016】设简单无向图 G 有 16 条边且每个顶点的度数都是 2，则图 G 有(　　)个顶点。

A. 10　　B. 12　　C. 8　　D. 16

5.【NOIP2010】关于拓扑排序，下面说法正确的是(　)。

A.所有连通的有向图都可以实现拓扑排序

B.对一个图而言，拓扑排序的结果是唯一的

C.拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点的前面

D.拓扑排序结果序列中的第一个结点一定是入度为 0 的点

课堂练习

6.【NOIP2008】设 T 是一棵有 n 个顶点的树，下列说法不正确的是（）。

A．T 有 n 条边　　B．T 是连通的

C．T 是无环的　　D．T 有 n-1 条边

7.【NOIP2017】设 G 是有 n 个结点、m 条边（n ≤ m）的连通图，必须删去 G 的（）条边，才能使得 G 变成一棵树。

A. m - n + 1　　B. m - n　　C. m + n + 1　　D. n - m + 1

8.【NOIP2015】6 个顶点的连通图的最小生成树，其边数为(　　)。

A.6　　B.5　　C.7　　D.4

9.【NOIP2014】设 G 是有 6 个结点的完全图，要得到一棵生成树，需要从 G 中删去(　　)条边。

A.6　　B.9　　C.10　　D.15

课堂练习

10.【NOIP2009】已知 n 个顶点的有向图，若该图是强连通的（从所有顶点都存在路径到达其他顶点），则该图中最少有多少条有向边？

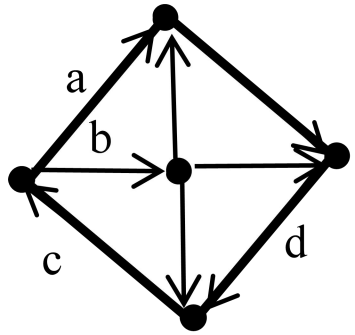
A.n　　B.n+1　　C.n-1　　D. n*(n-1)

11.【NOIP2011】无向完全图是图中每对顶点之间都恰有一条边的简单图。已知无向完全图 G 有 7 个顶点，则它共有(　　)条边。

A.7　　B.21　　C.42　　D.49

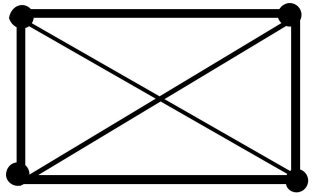
12.【NOIP2011】对一个有向图而言，如果每个节点都存在到达其他任何节点的路径，那么就称它是强连通的。例如，右图就是一个强连通图。事实上，在删掉边(　　)后，它依然是强连通的。

A.a　　B.b　　C.c



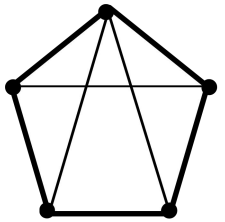
09 课堂练习

13.【NOIP2013】在一个无向图中，如果任意两点之间都存在路径相连，则称其为连通图。下图是一个有 4 个顶点、6 条边的连通图。若要使它不再是连通图，至少要删去其中的(　　)条边。



A.1　　B.2　　C.3　　D.4

14.【NOIP2013】在一个无向图中，如果任意两点之间都存在路径相连，则称其为连通图。右图是一个有 5 个顶点、8 条边的连通图。若要使它不再是连通图，至少要删去其中的(　　)条边。



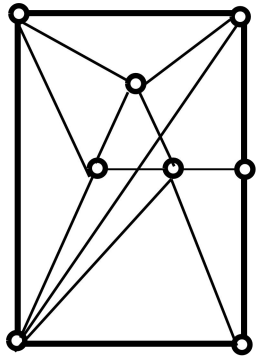
A.2　　B.3　　C.4　　D.5

15.【NOIP2013】二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。那么，12 个顶点的二分图至多有(　　)条边。

A.18　　B.24　　C.36　　D.66

09 课堂练习

16.【NOIP2015】对图 G 中各个结点分别指定一种颜色，使相邻结点颜色不同，则称为图 G 的一个正常着色。正常着色图 G 所必需的最少颜色数，称为 G 的色数。那么下图的色数是(　　)。



A.3　　B.4　　C.5　　D.6

17.【NOIP2016】G 是一个非连通简单无向图，共有 28 条边，则该图至少有（）个顶点。

A. 10　　B. 9　　C. 8　　D. 7

18.【NOIP2017】由四个不同的点构成的简单无向连通图的个数是（）。

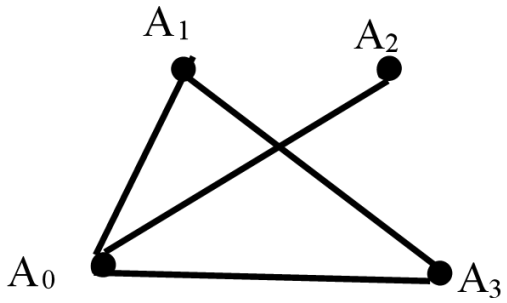
A. 32　　B. 35　　C. 38　　D. 41

19.【NOIP2018】由四个没有区别的点构成的简单无向连通图的个数是（　　）。

A.6　　B.7　　C.8　　D.9

09 课堂练习

20.【NOIP2013】以 A_0 作为起点，对下面的无向图进行深度优先遍历时，遍历顺序不可能是(　　)。



A.A_0, A_1, A_2, A_3　B.A_0, A_1, A_3, A_2　C.A_0, A_2, A_1, A_3　D.A_0, A_3, A_1, A_2

21.【NOIP2015】具有 n 个顶点，e 条边的图采用邻接表存储结构，进行深度优先遍历和广度优先遍历运算的时间复杂度均为(　　)。

A.$O(n^2)$　　B.$O(e^2)$　　C.O(ne)　　D.O(n + e)

22.【NOIP2013】对一个 n 个顶点、m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路时，如果<u>不使用</u>堆或其它优先队列进行优化，则其时间复杂度为(　　)。

A.$O(mn + n^3)$　　B.$O(n^2)$

C.$O((m + n) \log n)$　　D.$O((m + n^2) \log n)$

课堂练习

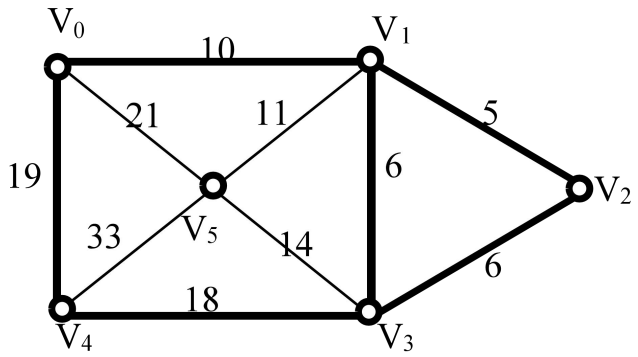
23.【NOIP2009】右图给出了一个加权无向图，从顶点 V_0 开始用 prim 算法求最小生成树。则依次加入最小生成树的顶点集合的顶点序列为：

A.V0, V1, V2, V3, V5, V4

B.V0, V1, V5, V4, V3, V3

C.V1, V2, V3, V0, V5, V4

D.V1, V2, V3, V0, V4, V5



【不定项选择题】

课堂练习

1.【NOIP2014】以下哪些结构可以用来存储图(　　)。

A.邻接矩阵　　B.栈　　C.邻接表　　D.二叉树

2.【NOIP2009】若 3 个顶点的无权图 G 的邻接矩阵用数组存储为{{0，1，1}，{1，0，1}，{0，1，0}}，假定在具体存储中顶点依次为：v_1，v_2，v_3 关于该图，下面的说法哪些是正确的：

A.该图是有向图。　　B.该图是强连通的。

C.该图所有顶点的入度之和减所有顶点的出度之和等于 1。

D.从 v_1 开始的深度优先遍历所经过的顶点序列与广度优先的顶点序列是相同的。

3.【NOIP2010】关于拓扑排序，下列说法正确的是(　　)。

A.所有连通的有向图都可以实现拓扑排序

B.对同一个图而言，拓扑排序的结构是唯一的

C.拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点的前面

D.拓扑排序结果序列中的第一个结点一定是入度为 0 的点

课堂练习

4.【NOIP2012】已知带权有向图 G 上的所有权值均为正整数，记顶点 u 到顶点 v 的最短路径的权值为 d(u, v)。若 v1, v2, v3, v4, v5 是图 G 上的顶点，且它们之间两两都存在路径可达，则以下说法正确的有(　　)。

A.v1 到 v2 的最短路径可能包含一个环

B.d(v1, v2) = d(v2, v1)

C.d(v1, v3) ≤ d(v1, v2) + d(v2, v3)

D.如果 v1→v2→v3→v4→v5 是 v1 到 v5 的一条最短路径，那么 v2→v3→v4 是 v2 到 v4 的一条最短路径

5.【NOIP2015】以下图中一定可以进行黑白染色的有(　　)。(黑白染色：为各个结点分别指定黑白两种颜色之一，使相邻结点颜色不同。)

A.二分图　　B.完全图　　C.树　　D.连通图

6.【NOIP2018】下列关于最短路算法的说法正确的有(　　)。

A. 当图中不存在负权回路但是存在负权边时，Dijkstra 算法不一定能求出源点到所有点的最短路。

B. 当图中不存在负权边时，调用多次 Dijkstra 算法能求出每对顶点间最短路径。

C. 图中存在负权回路时，调用一次 Dijkstra 算法也一定能求出源点到所有点的最短路。

D. 当图中不存在负权边时，调用一次 Dijkstra 算法不能用于每对顶点间最短路计算

09 课堂练习

7.【NOIP2011】对右图使用 Dijkstra 算法计算 S 点到其余各点的最短路径长度时，到 B 点的距离 d[B]初始时赋为 8，在算法的执行过程中还会出现的值有(　　)。

A．3　　B．7　　C．6　　D．5