辽宁大学考研专业课 853(学硕)真题答案

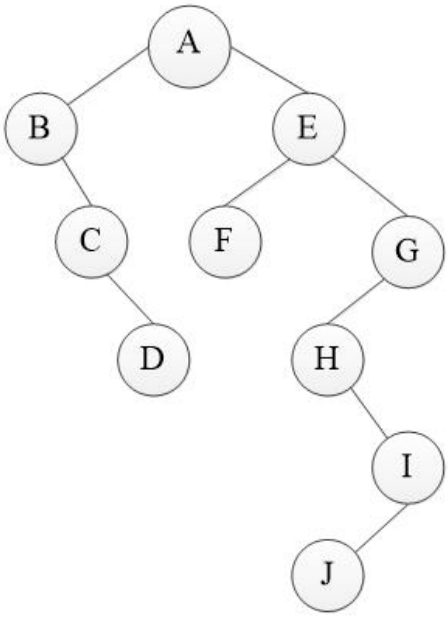
（2013 年—2018 年）

（2013 年）

一、BDCCB　ADDAC　BAADC

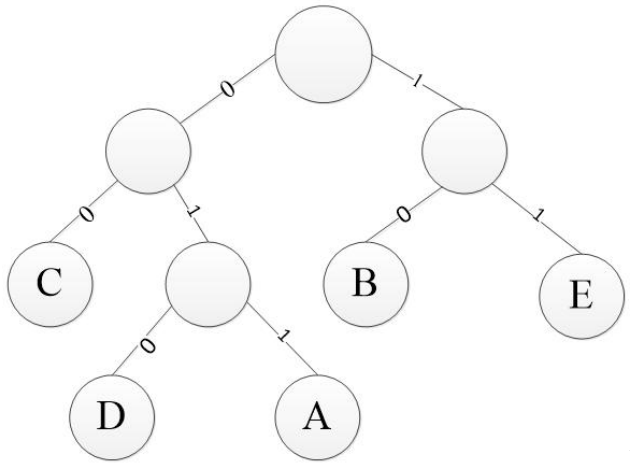
二、

1.



中序遍历：BCDAFEHJIG

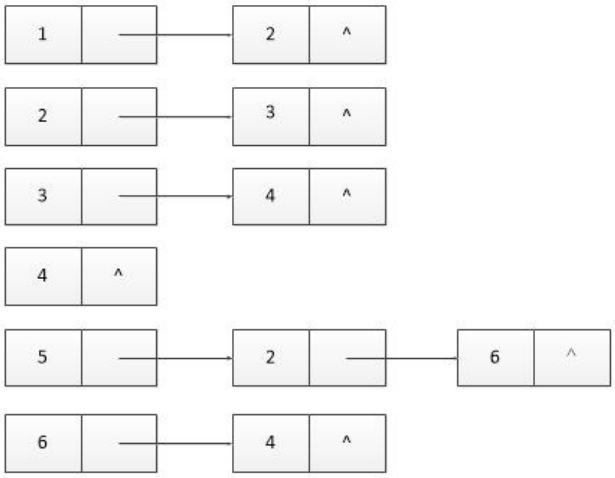
2.



WPL=5*2+1*3+4*3+7*2+9*2=57

A：011　B：10　C:00　D:010　E:11

3.邻接表：



一个拓扑排序：152364

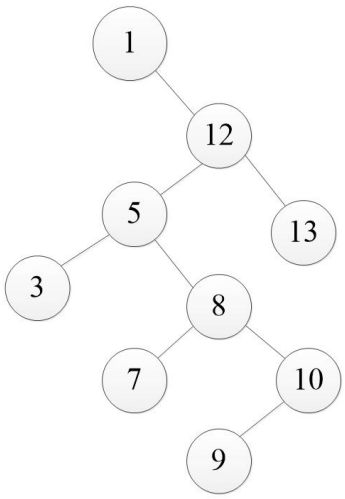
4. 哈希表如下：

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	01	14	55	27	68	19	20	84		23	11	10	77		

$ASL_{成功}=(1+1+1+2+1+1+3+4+3+1+3+2)/12=23/12$

注：$ASL_{失败}=(1+9+8+7+6+5+4+3+2+1+5+4+3)/13=\ldots\ldots$

5.二叉排序树如下：

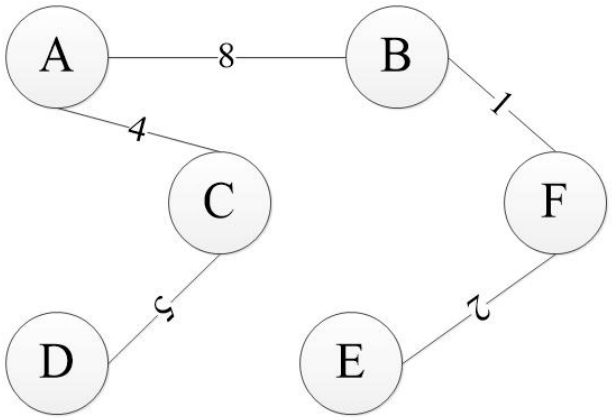


$ASL_{成功}=(1+1*2+2*3+2*4+2*5+1*6)/9=11/3$

6.中序遍历二叉树，输出所有大于等于 100 且小于等于 999 的结点的结点值。

7.输出结果：7；步骤：c=demo(9)=9-demo(7)=9-7+demo(5)=2+5-demo(3)=7-3+3=7

8.①从 F 出发到 ABCDE，FB=1，FE=2，选 FB；
②从 BF 出发到 ACDE，BA=8，BE=3，FE=2，选 FE；
③从 BEF 出发到 ACD，BA=8，EC=9，ED=10，选 BA；
④从 ABEF 出发到 CD，AC=4，EC=9，AD=1，DE=10，选 AC；
⑤从 ABCEF 出发到 D，AD=7，CD=5，ED=10，选 CD。
生成边次序：BF、FE、BA、AC、CD 。最小生成树如下：



9.

顶点	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟	第七趟
2	2 v1→v2						
3	3 v1→v3	3 v1→v3					
4	∞	7 v1→v2→v4	6 v1→v3→v4				
5	∞	∞	13 v1→v3→v5	13 v1→v3→v5	13 v1→v3→v5		
6	∞	∞	∞	10 v1→v3→v4→v6			
7	∞	∞	∞	∞	16 v1→v3→v5→v7	15 v1→v3→v4→v5→v7	
8	∞	∞	∞	∞	16 v1→v3→v4→v6→v8	16 v1→v3→v4→v6→v8	16 v1→v3→v4→v6→v8
集合 S	{v1,v2}	{v1,v2,v3}	{v1,v2,v3,v4}	{v1,v2,v3,v4,v6}	{v1,v2,v3,v4,v6,v5}	{v1,v2,v3,v4,v6,v5,v7}	{v1,v2,v3,v4,v6,v5,v7,v8}

由上表可知，v1 结点到其他各结点的最短距离和最短路径如下表：

起始点	到达点	最短路径	最短距离
v1	v2	v1→v2	2
	v3	v1→v3	3
	v4	v1→v3→v4	6
	v5	v1→v3→v5	13
	v6	v1→v3→v4→v6	10
	v7	v1→v3→v4→v5→v7	15
	v8	v1→v3→v4→v6→v8	16

10.
```
#include<iostream>
#include<malloc.h>
using namespace std;
struct LNode{
  int data;
  struct LNode *next;
 }LNode,*LinkList;
LNode *head1 = NULL;
void createList(LNode *head)
{
   head = (LNode *)malloc(sizeof*LNode);
   head->data = 1;
   head->next = NULL;

   LNode *p2 = (LNode *)malloc(sizeof*LNode);
   p2->data = 2;
   p2->next = NULL;
   head->next = p2;

   LNode *p3 = (LNode *)malloc(sizeof*LNode);
   p3->data = 3;
   p3->next = NULL;
   p2->next = p3;

   LNode *p4 = (LNode *)malloc(sizeof*LNode);
   p4->data = 4;
   p4->next = NULL;
   p3->next = p4;
}

int deleteKNode_2(LNode *head, int k,int e)
{
   if(head == NULL || k <= 0)
       return 0;
  LNode *pre = head;
```

```
    for(int i = 0; i < k - 1; ++i)
    {
        if(pre == NULL)
            return 0;
        pre = pre->next;
    }
    LNode *cur = head;
    while(pre->next)
    {
        pre = pre->next;
        cur = cur->next;
    }
    cout << cur->data;
    e=cur->data;
    cout << "succeed:" << endl;
    return 1;
}
int main()
{

    createList(head1);
    int k = 8;
    int m;
    cout << "Result:" << deleteKNode_2(head1, 2,m) << endl;
    cout<<"被删除节点数据："+m<<endl;
    return 0;
}
```

11. 64 个页面相当于 2^6，也就是 6 位表示页，，1KB 相当于 2^{10}，也就是 10 位表示块号。

（1）0A6C 二进制是 0000 1010 0110 1100，000010 是 2,2 分配的物理块号是 5，即 000101，物理地址：0001 0110 0110 1100，化为十六进制 166C(H);

（2）124B 二进制是 0001 0010 0100 1011，000100 是 4，分配的空闲物理块号是 7，即 000111，物理地址：0001 1110 0100 1011，化为十六进制 1E4B(H).

12.（1）144,148,151,176,178,131,103,95,92,87；

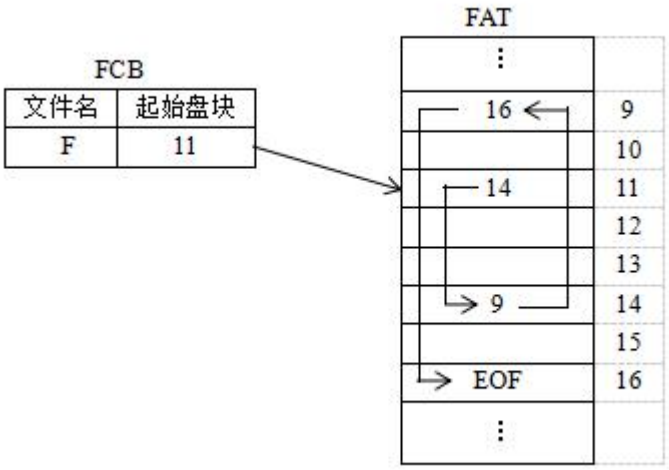
（2） $\mathrm{ASL}=(4+3+25+2+47+28+8+3+5)/9=125/9$

13.不能，系统会进入不安全状态。

	Allocation	Need	Available
P0	0 0 3	0 0 1	1 4 0
P1	1 0 0	1 6 5	
P2	1 5 8	2 1 2	
P3	0 0 3	0 6 5	
P4	0 0 1	0 6 5	

14.（1）500MB/1KB=500K，至少需要 2^{19} 字节即需要 19 位，扩展为半个字节整数倍即 4 的倍数，则需要 20 位，20/8=2.5 个字节，2.5*500K=1250KB；

（2）



15.

```
int A;
int B;
count A=0,count B = 0;
semphore MA,MB,mutex;
Process A{
P(MA);
if(countA)=0;
P(mutex);
countA=countA+1;
V(MA);
A 方向过桥；
P(MA);
countA=countA-1;
if countA=0;
V(mutex);
V(MA);
}
```

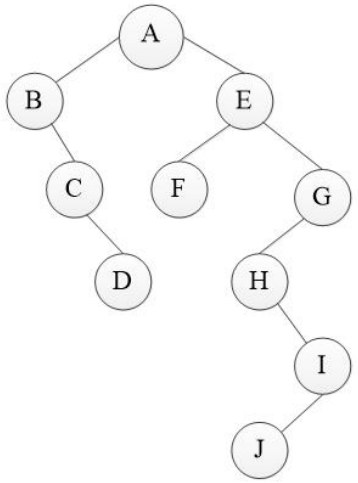
```
Process B{
P(MB);
if(countB)=0;
P(mutex);
countB=countB+1;
V(MB);
B 方向过桥；
P(MB);
countB=countB-1;
if countB=0;
V(mutex);
V(MB);
}
```

（2014 年）

一、BBCCB　　CDDAB　　BBBBA

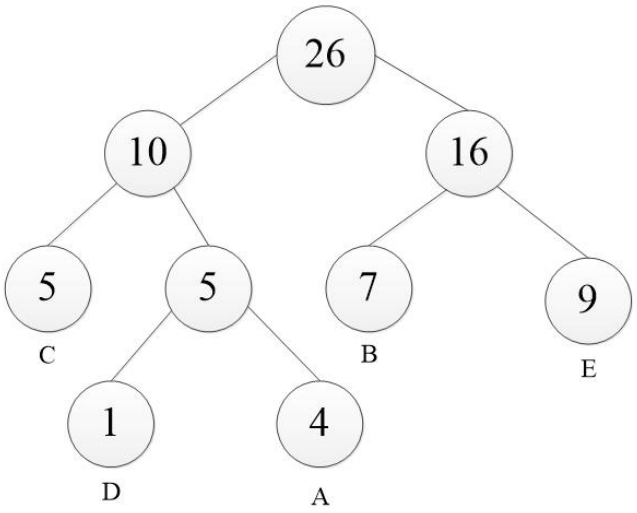
二、

1.



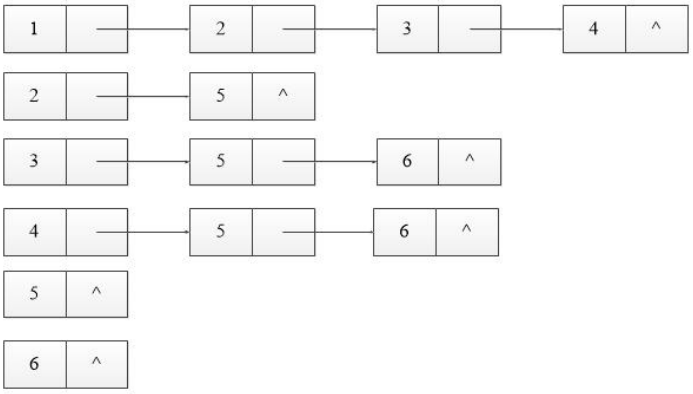
先序遍历序列：ABCDEFGHIJ

2. A=4，B=7，C=5,D=1,E=9



WPL=5*2+1*3+4*3+7*2+9*2=57

3.



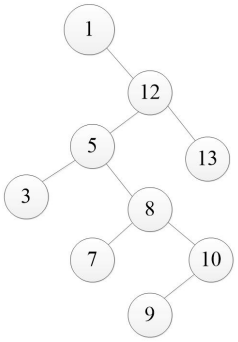
其中的一个拓扑序列：1,2,3,4,5,6

4.哈希表如下：

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	01	14	55	27	68	19	20	84		23	11	10	77		

ASL 成功=$(1+1+1+2+1+1+3+4+3+1+3+2)/12=23/12$

5.二叉排序树如下：



ASL 成功=$(1+1*2+2*3+2*4+2*5+1*6)/9=11/3$

6.将进入的序列逆序输出并删除元素 e.

7.demo(9)=9-demo(7)=7;

demo(7)=7-demo(5)=2;

demo(5)=5-demo(3)=5;

demo(3)=3-demo(1)=9;

demo(1)=3.

8.①从 F 出发到 ABCDE，FB=1，FE=2，选 FB；

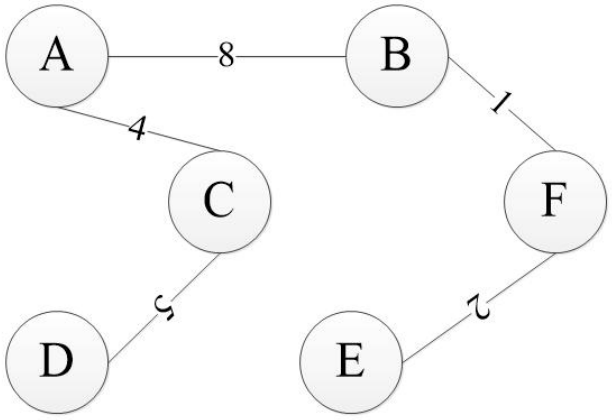
②从 BF 出发到 ACDE，BA=8，BC=10，BE=3，FE=2，选 FE；

③从 BEF 出发到 ACD，BA=8，BC=10，EC=9，ED=10，选 BA；

④从 ABEF 出发到 CD，AD=7，AC=4，BC=10，EC=9，ED=10，选 AC；

⑤从 ABCEF 出发到 D，AD=7，CD=5，ED=10，选 CD。

生成边次序：BF、FE、BA、AC、CD 。最小生成树如下：



9.

顶点	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟
B	∞	∞	17 A→D→C →B	16 A→G→B	16 A→G→B	16 A→G→B
C	∞	6 A→D→C				
D	1 A→D					
E	∞	11 A→D→E	11 A→D→E	11 A→D→E		
F	∞	∞	∞	∞	14 A→D→E →F	
G	7 A→G	7 A→G	7 A→G			
集合	{A,D}	{A,D,C}	{A,D,C,G}	{A,D,C,G, E}	{A,D,C,G, E,F}	{A,D,C,G, E,F,B}

由上表可知，A 结点到其他各结点的最短距离和最短路径如下表：

起始点	到达点	最短路径	最短距离
A	B	A→G→B	16
	C	A→D→C	6
	D	A→D	1
	E	A→D→E	11
	F	A→D→E→F	14
	G	A→G	7

10.#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef MAXLISTSIZE 100

```
typedef struct seqlist
{
    ElemType *elem; //存储空间基址
    int listlength ;          //当前长度
    int listsize;             //允许的最大存储容量(以 sizeof(ElemType)为单位)
}seqlist;                     //俗称顺序表

void InitList(seqlist *L);
void InitList(seqlist *L) //
{
    printf("--------Start InitList()--------\n");
    L->elem = (int*)malloc(MAXLISTSIZE * sizeof(int) );
    if( L->elem==NULL )
    {
        exit(1);                    //存储分配失败,程序退出
    }
    L->listsize = MAXLISTSIZE;      //该顺序表可以存储元素的最大容量
    L->listlength = 0;              //顺序表的初始长度为 0
    printf("--------End InitList()--------\n");
}

int deleteKNode_2(seqlist *L, int k,int e)
{
    if(L->Length==0)      //线性表为空
    return 0;


        for(int i=n-k+1;i<L->Length;i++)
        {
            L->data[i-1]=L->data[i];
        }
        L->Length--;
    return 1;
}
int main()
{

    createList(head1);
    int k = 8;
    int m;
    cout << "Result:" << deleteKNode_2(head1, 2,m) << endl;
    cout<<"被删除节点数据："+m<<endl;
    return 0;
}
```

11.（1）85,80,70,46,38,10,98,100,125,135,160;

（2）$\mathrm{ASL}=(10+24+8+28+88+2+25+10+25)/10=22$

12.（1）FIFO 置换算法选择最优先进入内存的页面进行替换，由表可知物理块 3 的第 3 页最先进入，则选它；

（2）LRU 算法最近最长时间未使用的进行替换，即物理块 0 中第 2 页；

（3）改进 Clock 算法从上一位置开始扫描，首先寻找未被访问和修改的，即第 2 块中第 0 页。

13.

走向	6	5	4	3	6	5	7	6	5	4	3	7
块 1	6	6	6	6			7	7	7	7	3	3
块 2		5	5	5			5	6	6	6	6	7
块 3			4	4			4	4	5	5	5	5
块 4				3			3	3	3	4	4	4
缺页数	√	√	√	√			√	√	√	√	√	√

缺页次数 10，缺页率 10/12=5/6.

14.（1）$(16*2^{10}*2^{10})\mathrm{KB}/4\mathrm{KB}=4*2^{20}$，$\left|4*2^{20}/32\right|=2^{17}$ 字节。

（2）字节的整偶数倍，即 8 的偶数倍，取 $32,32/8*4*2^{20}=16\mathrm{MB}$

15.

```
semaphore mutex=1;
semaphore countA=0;
semaphore countB=0;
A()
{
while(1){
if(countB==0)
      wait(mutex);
wait(mutex);
countA++;
if(countA==0)
访问文件；
countA--;
if(countA==0)
signal(mutex);
}
}
```

```
B(){
while(1){
if(countA==0)
      wait(mutex);
wait(mutex);
countB++;
if(countA==0)
访问文件；
countB--;
if(countB==0)
signal(mutex);
}
}
```

16.（1）$\lceil 500/32\rceil=16$ （2）$32\mathrm{i}+j$

（3）申请时自上而下，自左而右扫描位示图，跳过为 1 的位找到第一个迁到 0

位，根据 i,j 算出对应块号，并分配出去。

17.64 个页面相当于 2^6，也就是 6 位表示页，，1KB 相当于 2^{10}，也就是 10 位表示块号。

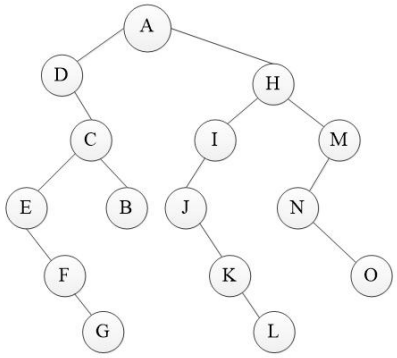
（1）0A5C 二进制是 0000 1010 0101 1100，000010 是 2,2 分配的物理块号是 4，即 000100，物理地址：0001 0010 0101 1100，化为十六进制 125C(H);
（2）103C 二进制是 0001 0000 0011 1100，000100 是 4，分配的空闲物理块号是 6，即 000110，物理地址：0001 1000 0011 1100，化为十六进制 183C(H).

（2015 年）

一、BBCAB　CACCD　CBCAB

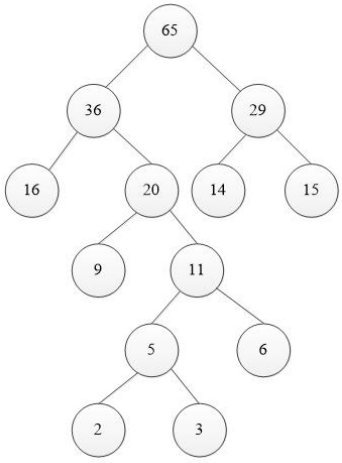
二、

1.



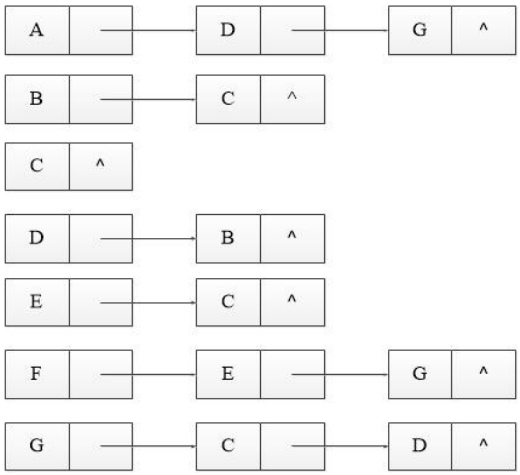
先序遍历序列：ADCEFGBHIJKLMNO

2.



$$WPL = 2*5+3*5+14*3+15*3+9*3+6*4+16*1=179$$

3.



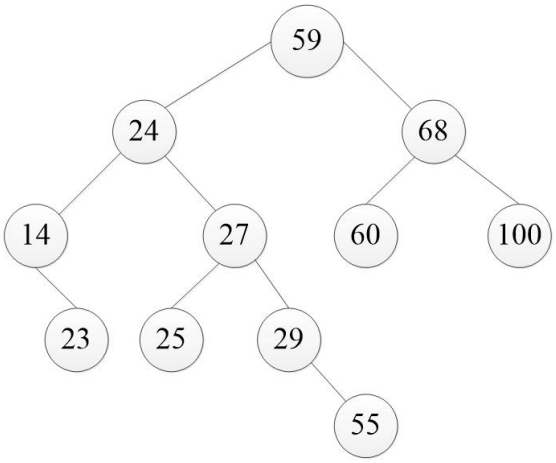
一个拓扑序列：AFEGDBC

4.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	01	14	55	27		84	19	20		23	11	10	77		

$$ASL_{成功}=(1+1+1+2+1+2+1+4+1+3+2)/11=19/11$$

5.



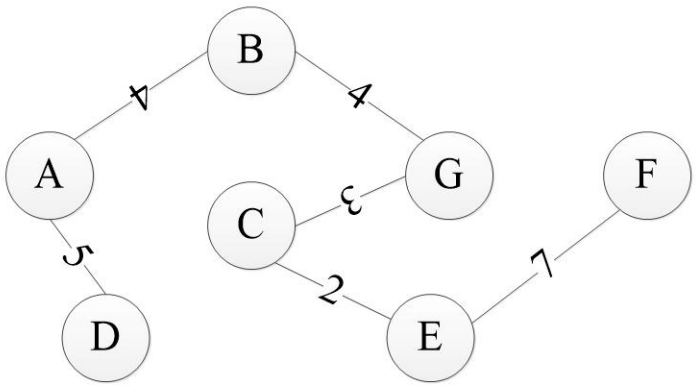
$$ASL=(1*1+2*2+4*3+3*4+1*5)/11=34/11$$

6.删除栈中元素 e，将其余元素逆序输出。

7.输出结果：7；步骤：c=demo(9)=9-demo(7)=9-7+demo(5)=2+5-demo(3)=7-3+3=7

8.①从 F 出发到 ABCDEFG，FB=8,FE=7,选 FE；②从 EF 出发到 ABCDG，FB=8，EC=2，ED=10，选 EC；③从 CEF 出发到 ABDG，FB=8,CB=9,CG=3,ED=10,选 CG；④从 CEFG 出发到 ABD，FB=8，GB=4，CB=9，ED=10,选 GB；⑤从 BCEFG 出发到 AD，BA=4，ED=10，选 BA；⑥从 ABCEFG 出发到 D，AD=5，ED=10，选 AD

生成边次序：FE、EC、CG、GB、BA、AD



9.

顶点	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟
B	∞	∞	16 A→G→B	16 A→G→B	16 A→G→B	16 A→G→B
C	∞	6 A→D→C				
D	1 A→D					
E	∞	11 A→D→E	11 A→D→E	11 A→D→E		
F	∞	∞	∞	∞	14 A→D→E →F	
G	7 A→G	7 A→G	7 A→G			
集合	{A,D}	{A,D,C}	{A,D,C,G}	{A,D,C,G, E}	{A,D,C,G, E,F}	{A,D,C,G, E,F,B}

由上表可知，A 结点到其他各结点的最短距离和最短路径如下表：

起始点	到达点	最短路径	最短距离
A	B	A→G→B	16
	C	A→D→C	6
	D	A→D	1
	E	A→D→E	11
	F	A→D→E→F	14
	G	A→G	7

10.#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef MAXLISTSIZE 100

```
typedef struct seqlist
{
    ElemType *elem; //存储空间基址
    int listlength ;          //当前长度
    int listsize;             //允许的最大存储容量(以 sizeof(ElemType)为单位)
}seqlist;                     //俗称顺序表

void InitList(seqlist *L);

void InitList(seqlist *L) //
{
    printf("--------Start InitList()--------\n");
    L->elem = (int*)malloc(MAXLISTSIZE * sizeof(int) );
    if( L->elem==NULL )
```

```
    {
        exit(1);                        //存储分配失败,程序退出
    }
    L->listsize = MAXLISTSIZE;      //该顺序表可以存储元素的最大容量
    L->listlength = 0;                  //顺序表的初始长度为 0
    printf("--------End InitList()--------\n");
}

int deleteKNode_2(seqlist *L, int k,int e)
{
    if(L->Length==0)       //线性表为空
    return 0;


        for(int i=n-k+1;i<L->Length;i++)
         {
              L->data[i-1]=L->data[i];
         }
         L->Length--;
    return 1;
}
int main()
{

    createList(head1);
    int k = 8;
    int m;
    cout << "Result:" << deleteKNode_2(head1, 2,m) << endl;
    cout<<"被删除节点数据："+m<<endl;
    return 0;
}
```

11.

走向	4	3	2	1	4	3	5	4	3	2	1	5
块 1	4	4	4	4			4			4	4	5
块 2		3	3	3			3			3	3	3
块 3			2	2			5			5	1	1
块 4				1			1			2	2	2
缺页	√	√	√	√			√			√	√	√

缺页次数 8 次，缺页率为 8/12=2/3.

12.

（1）$2^8 = 256$ 段；（2）$2^{16} = 64\text{KB}$；

（2）①逻辑地址[1,50]的段内地址超过段长且无法进行地址变换，将产生越界中断；

②逻辑地址[2,30]所在的第 2 段没有驻存在内存中，无法进行地址转换，将产生缺段中断；

③逻辑地址[3,70]的主存地址：4000+70=4070.

13.64 个页面相当于 2^6，也就是 6 位表示页，，1KB 相当于 2^{10}，也就是 10 位表示块号。

（1）①0B7B 二进制是 0000 1011 0111 1011，000010 是 2,2 分配的物理块号是 7，即 000111，物理地址：0001 1111 0111 1011，化为十六进制 1F7B(H);

②046F 二进制是 0000 0100 0110 1111，000001 是 1，1<8 是合理页面，分配的空闲物理块号是 8，即 001000，物理地址：0010000001101111，化为十六进制 206F(H).

（2）2ns*2=4ns,2ns+40+2*2ns=46ns

14.（1）85,98,100,125,135,160,80,70,46,38,10;

（2）$\mathrm{ASL}=(13+2+25+10+25+80+10+24+8+28)/10=45/2$

15.可以分配，P3 请求（0,4,1,0）后，

Process	Allocation A B C D	Need A B C D	Available A B C D
P0	0 0 3 2	0 0 1 2	
P1	1 0 0 0	1 6 5 0	
P2	1 3 5 4	2 3 5 6	1 2 1 2
P3	0 4 4 2	0 2 4 2	
P4	0 0 1 4	0 6 5 6	

16. 四个信号量 S1,S2,S3,S4 相协调工作

```
semaphore
S1=S2=S3=S4=0
Process S1(){
有客人;
P(S1);
接受点菜;
V(S2);
util false;
}
```

```
Process S2(){
P(S2);
做菜;
V(S3);
util false;
}
```

```
Process S3(){
P(S3);
打包;
V(S4);
util false;
}
```

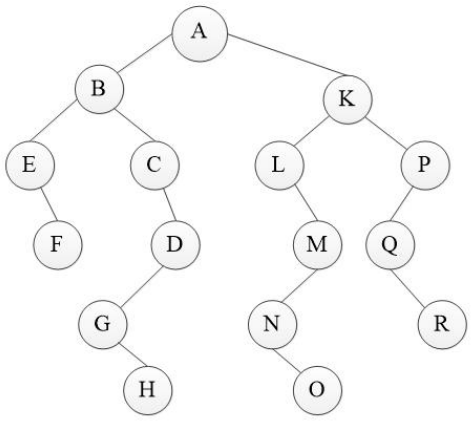
```
Process S4(){
P(S4);
收款;
V(S1);
util false;
}
```

（2016 年）

一、BDAAD　CDAAB　DBABC

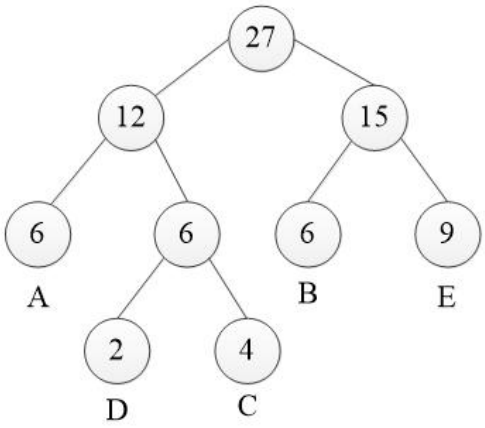
二、

1.



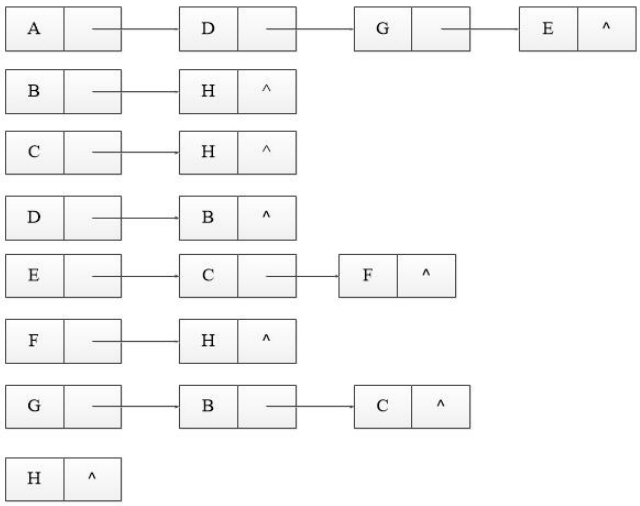
中序遍历：EFBCGHDALNOMKQRP

E：9，A:6，B：6；C:4；D:2



$$WPL = 6*2 + 2*3 + 4*3 + 6*2 + 9*2 = 60$$

3.



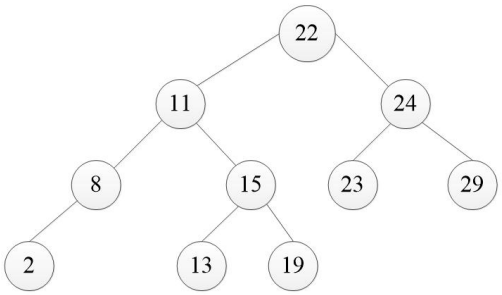
一个拓扑序列：ADGEBCFH

4.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	01	66	29		18	57	33	21	8	19	24	12	77		

$$ASL=(1+1+1+1+2+1+2+1+2+5+1+2)/12=5/3$$

5.



$$\mathrm{ASL}=(1*1+2*2+4*3+3*4)/10=29/10$$

6.先序遍历二叉链表，并统计结点在 A~Z 之间的结点个数；时间复杂度 O(n)。

7.un(4)=un(3)+un(2)；

un(3)=un(2)+un(1);

un(2)=un(1)+un(0)=1+1=2;

un(3)=3,un(4)=3+2=5

8.①从 A 出发到 BCDEFG，AB=4,AC=8,AD=5，选 AB；

②从 AB 出发到 CDEFG，AD=5，AC=8，BC=9，BG=4，BF=8，选 BG；

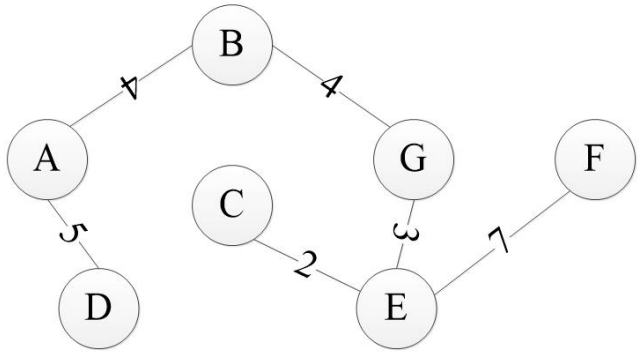
③从 ABG 出发到 CDEF，AD=5，AC=8，BC=9，BF=8，GE=3，选 GE；

④从 ABEG 出发到 DCF，AD=5，AC=8，BC=9，BF=8，EC=2,ED=10,EF=7,选 EC；

⑤从 ABCEG 出发到 DF，AD=5,ED=10,BF=8,EF=7,选 AD；

⑥从 ABCDEG 出发到 F，BF=8,EF=7,选 EF。

生成边次序：AB BG GE EC AD EF



9.

顶点	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟
B	∞	∞	16 A→D→C →B	12 A→G→B		
C	9 A→C	6 A→D→C				
D	1 A→D					
E	∞	12 A→D→E	12 A→D→E	12 A→D→E	12 A→D→E	
F	∞	∞	18 A→D→C →F	18 A→D→C →F	16 A→G→B →F	15 A→D→E →F
G	7 A→G	7 A→G	7 A→G			
集合	{A,D}	{A,D,C}	{A,D,C,G}	{A,D,C,G, B}	{A,D,C,G, B,E}	{A,D,C,G, B,E,F}

由上表可知，A 结点到其他各结点的最短距离和最短路径如下表：

起始点	到达点	最短路径	最短距离
A	B	A→G→B	12
	C	A→D→C	6
	D	A→D	1
	E	A→D→E	12
	F	A→D→E→F	15
	G	A→G	7

10.ListInsert(SqList*L,int e){

pNode i,j;

for(i=0;i<L.length;i++){

if(L.[i]==e){

return 1;

else if(L.[i]>e)

j=i;}}

for(i=L.length;j<i;i--){

L.[i-1]=L.[i];

L.[j]=e;

L.length++;

return 1;}

}

11.多道程序设计技术允许多个程序同时进入内存并运行。即同时把多个程序放入内存，并允许他们交替在 CPU 中运行，他们共享系统中的各种硬、软件系统。优点是资源利用率高、多道程序共享计算机资源从而使各种资源得到充分利用，系统吞吐量大，CPU 和其他资源保持忙碌状态。

12.①0B62 二进制为 0000 1011 0110 0010，000010 为 2，分配的物理块号为 9，即 001001，

物理地址为 0010 0111 0110 0010，化为十六进制为 2762（H）；
②1743 二进制为 0001 0111 0100 0011,000101 为 5，没有可以分配的物理块号；
③1A64 二进制为 0001 1010 0110 0100,000110 为 6，没有可以分配的物理块号。
13.（1）

$Need$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 6 & 4 & 2 \end{bmatrix}$$

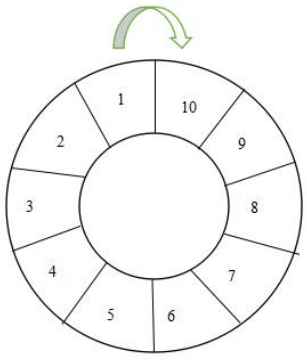
（2）安全序列：$P_0 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$

14.（1）4 组空闲盘块。第一组 2 块，第 2 组、第 3 组分别有 100 块，第 4 组虽记为 100 块，但除去用于结束标记后的实际只有 99 块，总计 301 块；
（2）盘块号 299,300,301,301

空闲盘块栈

s.nfree	98
s.nfree(0)	400
s.nfree(1)	399
	⋮
s.nfree(97)	303
s.nfree(98)	302
s.nfree(99)	301

15.



（1）读一个逻辑记录的时间 20ms/10=2ms，读出后还需要 4ms 处理时间，故当磁头处于某记录起点时，处理它需要 6ms 时间，又逻辑记录按逆时针方向安排，因此处理完一个逻辑记录后将磁头转到下一个逻辑记录需要 16ms 时间（6ms 后磁盘已经到达 4 的位置），故处理完这 10 个记录需要花费 $6+9*(16+6)=204ms$

（2）可使处理程序处理完一个记录后，磁头刚好转到下一个记录的始点，顺序为：记录 1，记录 4，记录 7，记录 10，记录 3，记录 6，记录 9，记录 2，记录 5，记录 8，共需要时间 6*10=60ms.

16.semaphore well=1;//水井互斥访问
semaphore vat=1;//水缸互斥访问
semaphore empty=12;//水缸中剩余空间容纳水桶数
semaphore pail=4;//有多少水桶可用
semaphore full=0;//表示水缸中水的桶数

```
//老和尚
while(1){
  P(full);
  P(pail);
  P(vat);
  从水缸中打一桶水；
  V(vat);
  V(empty);
  喝水；
  V(pail);
}
```

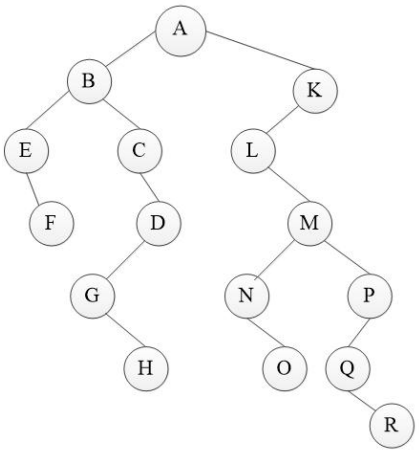
```
//小和尚
while(1){
  P(empty);
  P(pail);
  P(well);
  从井中打一桶水；
  V(well);
  V(vat);
  将水倒入桶中；
  V(vat);
  V(full);
  V(pail);
  }
```

（2017 年）

一、BAACA　CCBAD　ADCBD

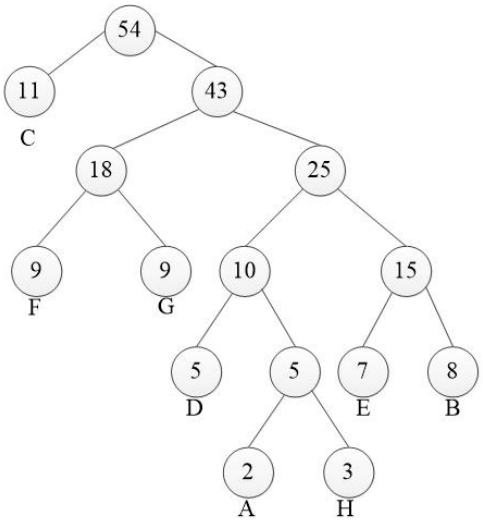
二、

1.

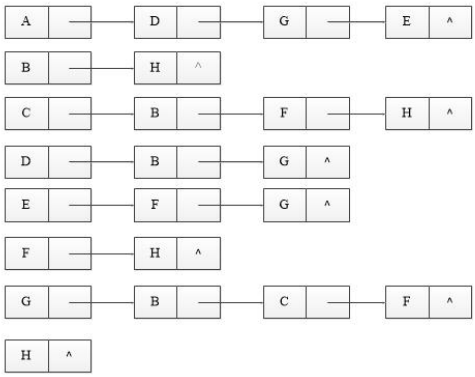


先序遍历：ABEFCDGHKLMNOPQR;
中序遍历：EFBCGHDALNOMQRPK;
后序遍历：FEHGDCBONRQPMLKA.

2. $WPL = 11*1+9*3+5*4+7*4+8*4+2*5+3*5=170$



3.



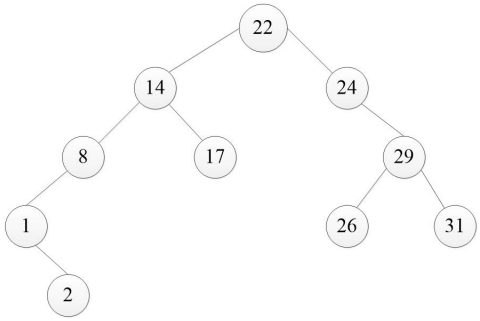
2 个拓扑序列：ADEGCBFH、AEDGCFBH

4.

0	1	2	3	4	5	6	7	8	9	10	11	12
	01	12			27	61	18	8	31	21	19	

$$ASL_{成功}=(1+1+1+1+1+1+1+4+2)/11=13/11$$

5.



$$ASL=(1*1+2*2+3*3+4*3+5*1)/10=31/10$$

6.先序遍历二叉树，统计结点值大于等于 0 小于等于 9 的个数；时间复杂度 O(n).

7.un(6)=un(5)+un(4)+un(3)=13+7+4=24.

un(5)=un(4)+un(3)+un(2)=7+4+2=13.

un(4)=un(3)+un(2)+un(1)=4+2++1=7.

un(3)=un(2)+un(1)+un(0)=2+1+1=4.

8.①从 B 出发到 ACDEFGH，BA=1,BC=7,BG=4,BF=8,选 BA；

②从 AB 出发到 CDEFGH，AD=5,AC=8,BC=7,BG=4,BF=8,选 BG；

③从 ABG 出发到 CDEFH，AD=5,AC=8,BC=7,BF=8,GC=5,GE=3,选 GE；

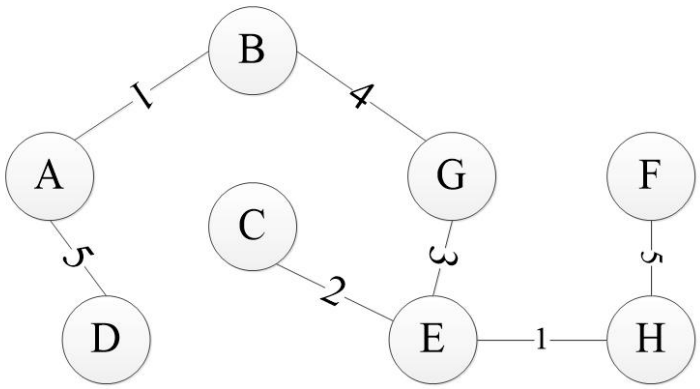
④从 ABEG 出发到 CDFH，AD=5,AC=8,BC=7,BF=8,GC=5,ED=9,EC=2,EF=7,EH=1,选 EH；

⑤从 ABEGH 出发到 CDF，AD=5,AC=8,BC=7,BF=8,GC=5,EC=2,AD=5,FH=5,选 EC；

⑥从 ABCEGH 出发到 DF，AD=5,ED=9,FB=8,FE=7,FH=5,选 AD；（或者选 HF 也可以）

⑦从 ABCDEGH 出发到 D，ED=9，HF=5 选 HF。

生成边次序：AB BG GE EH EC AD HF



9.

顶点	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟	第七趟	第八趟
B	∞	∞	14 A→C→B	11 A→G→B	11 A→G→B			
C	2 A→C	2 A→C						
D	1 A→D							
E	∞	9 A→D→E	9 A→D→E	9 A→D→E				
G	7 A→G	7 A→G	7 A→G					
H	∞	∞	∞	∞	15 A→G→B→H	15 A→G→B→H	15 A→G→B→H	15 A→G→B→H
I	∞	∞	∞	∞	12 A→D→E→I	12 A→D→E→I		
J	∞	∞	14 A→C→J	14 A→C→J	14 A→C→J	14 A→C→J	14 A→C→J	
集合	{A,D}	{A,D,C}	{A,D,C,G}	{A,D,C,G,E}	{A,D,C,G,E,B}	{A,D,C,G,E,B,I}	{A,D,C,G,E,B,I,J}	{A,D,C,G,E,B,I,J,H}

由上表可知，A 结点到其他各结点的最短距离和最短路径如下表：

起始点	到达点	最短路径	最短距离
A	B	A→G→B	11
	C	A→C	2
	D	A→D	1
	E	A→D→E	9
	G	A→G	7
	H	A→G→B→H	15
	I	A→D→E→I	12
	J	A→C→J	14

10. LinkList ListUnion(LinkList a, LinkList b, LinkList c)

{ // 单链表 a 和 b 元素不减，原地合并为递增有序链表 c

Node *pa, *pb, *pc;

pa = a->next;

pb = b->next;

pc = c->next;

while (pa && pb)

```
{
if (pa->data <= pb->data)
{
pc->next = pa;
pc = pa;
pa = pa->next;
}
else
{
pc->next = pb;
pc = pb;
pb = pb->next;
}
}
pc->next = pa ? pa : pb; //  追加剩余结点
free(a);
free(b);
}
```

11.死锁是指多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程将无法向前推进；

原因：①系统资源的竞争；②进程推进顺序非法；

必要条件：①互斥条件；②不可剥夺条件；③请求和保持条件；④循环等待条件。

12.设页号为 P，页内位移为 D

1011： P=INT(1011/1024)=0,D=1011MOD1024=1011, 查表第 0 页在第 2 块，物理地址 2*1024+1011=3059；

2148： P=INT(2148/1024)=2,D=2148MOD1024=100, 查表第 2 页在第 1 块，物理地址 1*1024+100=1124；

4000： P=INT(4000/1024)=3,D=4000OMD1024=928, 查表第 3 页在第 6 块，物理地址 6*1024+928=7072；

5012：P=INT(5012/1024)=4,D=5012MOD1024=916,因页号超过页表长度，该逻辑地址非法。

13.（1）

$$Need$$

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 0 & 2 \\ 1 & 0 & 3 \\ 4 & 2 & 0 \end{bmatrix}$$

（2）P3 请求资源，剩余资源数（4,2,3）；P2 请求资源，剩余资源数（8,3,4）；

P1 请求资源，剩余资源数（9,3,4）；P4 请求资源，剩余资源数（9,3,6）；

安全序列：$P_3 \rightarrow P_2 \rightarrow P_1 \rightarrow P_4$

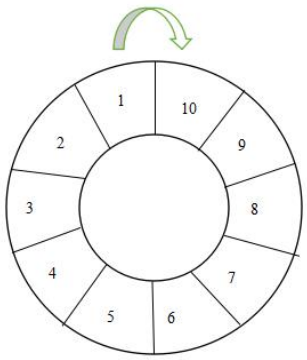
12.（1）4 组空闲盘块。第一组 2 块，第 2 组、第 3 组分别有 100 块，第 4 组虽记为 100 块，但除去用于结束标记后的实际只有 99 块，总计 301 块；

（2）盘块号 299,300,301,301

空闲盘块栈

s.nfree	98
s.nfree(0)	400
s.nfree(1)	399
	⋮
s.nfree(97)	303
s.nfree(98)	302
s.nfree(99)	301

15.



（1）读一个逻辑记录的时间 20ms/10=2ms，读出后还需要 4ms 处理时间，故当磁头处于某记录起点时，处理它需要 6ms 时间，又逻辑记录按逆时针方向安排，因此处理完一个逻辑记录后将磁头转到下一个逻辑记录需要 16ms 时间（6ms 后磁盘已经到达 4 的位置），故处理完这 10 个记录需要花费 $6+9*(16+6)=204ms$

（2）可使处理程序处理完一个记录后，磁头刚好转到下一个记录的始点，顺序为：记录 1，记录 4，记录 7，记录 10，记录 3，记录 6，记录 9，记录 2，记录 5，记录 8，共需要时间 6*10=60ms.

16.semaphore empty=1;//表示进程 R 可向缓冲区 B 中读入整数个数
semaphore SW1=0，SW2=0；//表示开始时 B 中没有奇数（偶数）可供 W1（W2）读取
cobegin

```
Process R(){
  int X;
  while(1){
  从输入设备上读一个整数 x;
  P(empty);
  B=x;
  if(x%2==1)
    V(SW1)
  else
    V(SW2)
  }
}
```

```
Process SW1(){
 int y;
 while(1){
 P(SW1);//收到 R 发来的信号，已产生一个奇数
 y=B;//取出缓冲区 B 中的奇数到变量 y 中
 打印 y 中的数；
 V(empty);//向 R 发出信号，进程 R 可以读入另一个数到 B
 }
}
```

```
Process SW2(){
 int z;
 while(1){
 P(SW2);//收到 R 发来的信号，已产生一个偶数
 z=B;//取出缓冲区 B 中的偶数到变量 z 中
 打印 z 中的数；
 V(empty);
}
```

（2018 年）

1.线程是程序执行流的最小单元。一个标准的线程由线程 ID，当前指令指针(PC)，寄存器集合和堆栈组成。

进程和线程都是由操作系统程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性。线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。在有进程和线程的系统中，进程是系统资源分配的独立单位，而线程是可调度运行的独立单位。

2.死锁是指多个进程因竞争资源而造成的一种僵局,若无外力的作用,这些进程都将永远不能再向前推进。所以,死锁是由于系统中多个进程所共享的资源不足以同时满足需要时,引起对资源的竞争而产生的。但竞争资源不一定都会产生死锁,因为只要进程推进顺序合法,就不会产生死锁。

3.主存分为 256 块，说明一共存放 256 个页面，每块大小为 4KB，所以每页大小 4*1024=4096,9016=4096*2+824，所以页号为 2，根据表中页号 2 所对应块号为 32，所以物理地址为 32*4KB+824=952

同理，12300=4096*3+12，页号为 3，状态为 1，表示并未被主存装入，由此将产生缺页中断。

4、

160
135
125
100
98
80
70
46
38
10

120
85

（1）80->70->46->38->10->98->100->125->135->160

(2)(5+10+24+8+28+88+2+25+10+25)/10=22.5

5、(1)A 和 B 两个进程的相互制约关系是既有互斥又有同步：对缓冲区的访问必须互斥，并且当缓冲区满时，A 进程不可以写，必须等待；当缓冲区空时，B 进程不可以读，必须等待。

(2)用 P、V 操作表示 A、B 进程的同步算法如下：

```
BEGIN
Buffer: ARRAY [0..N-A] of integer;
m, out: Integer;
S0，SA，SB: Semaphore;
S0:=A; SA"=0; SB:=N;
in:=out:=0;
Cobegin
Process PROCEDURE A;
BEGIN
   LA:  生产数据 m;
```

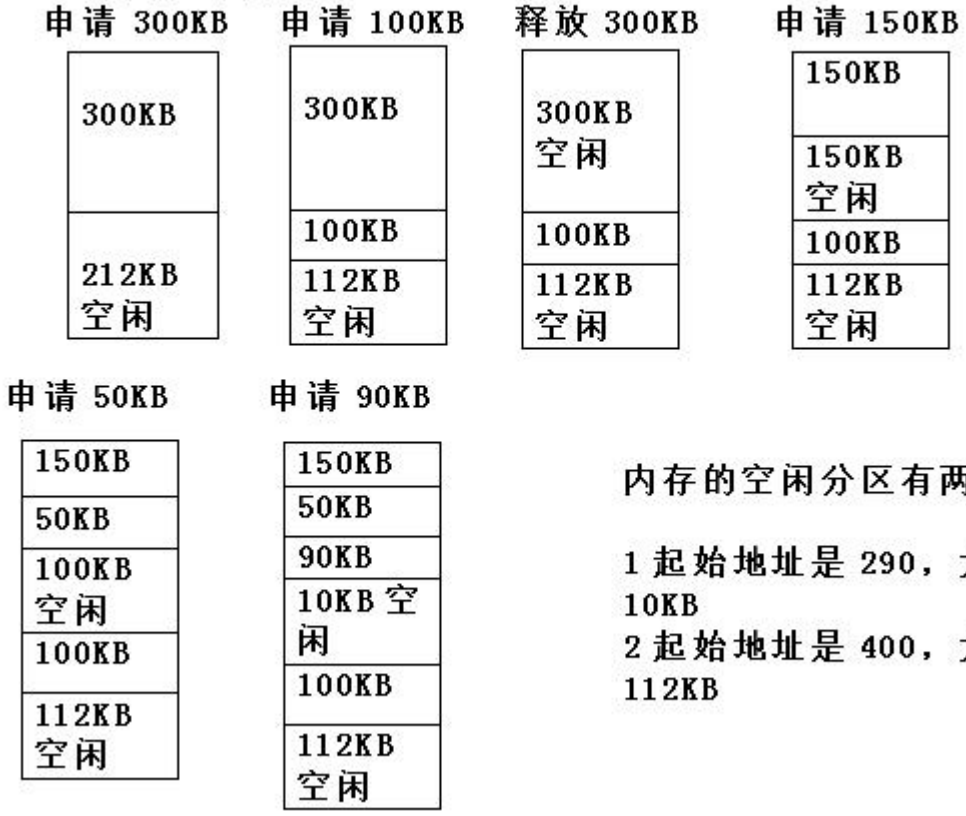
```
Process PROCEDURE B;
BEGIN
   LB: P(SA); //等待新数据
```

```
    P(SB);  //等待可用的缓冲区空间
    P(S0); //保护对 in、out 的操作
    Buffer(in):=m;
    in:=(in+A) MOD N;
    V(SA); //有新数据，唤醒消费者进程
    V(S0);
    Goto LA
END
Coend
END
```

```
    P(S0); //保护对 in、out 的操作
    m"=burfer(out);
    out:=(out+A) MOD N;
    V(SB); //增加可用空间
    V(S0);
    消费 m;
    goto LB
END
```

6、

（1）首次适用算法



内存的空闲分区有两块

1 起始地址是 290，大小是 10KB

2 起始地址是 400，大小是 112KB

（2）最佳适用算法

申请 300KB

申请 100KB

释放 300KB

申请 150KB

申请 50KB

申请 90KB

内存的空闲分区有两块

1 起始地址是 240，大小是 60KB

2 起始地址是 450，大小是 62KB

（3）如果再申请 80KB，首次适用算法可以分配出 80KB 而最佳适用算法则没有 80KB 的连续空闲分区让其申请

首次适用算法

7、FCFS:

作业执行次序	执行时间	优先级	等待时间	周转时间	带权周转时间
1	10	3	0	10	1
2	1	1	10	11	11
3	2	3	11	13	6.5
4	1	4	13	14	14
5	5	2	14	19	3.8

系统中作业的平均周转时间为 T=(10+11+13+14+19)/5=13.4

系统中作业的平均带权周转时间为 W=(1+11+6.5+14+3.8)/5=7.26

RR：

作业	周转时间	带权周转时间
1	19	1.9
2	2	2
3	7	3.5
4	4	4
5	14	2.8

各作业在系统中的执行情况为：

（1,2,3,4,5），（1,3,5），（1,5,1,5,1,5），（1,1,1,1,1）

作业 1 的周转时间 T1=19；T2=2；T3=7;T4=44;T5=14

系统中作业的平均周转时间为 T=(19+2+7+4+14)/5=9.2

系统中作业的平均带权周转时间为 W=(1.9+2+3.5+4+2.8)/5=2.84

SJF：

作业执行次序	执行时间	优先级	等待时间	周转时间	带权周转时间
2	1	1	0	1	1
4	1	4	1	2	2
3	2	3	2	4	2
5	5	2	4	9	1.8
1	10	3	9	19	1.9

系统中作业的平均周转时间为 T=(1+2+4+9+19)/5=7

系统中作业的平均带权周转时间为 W=(1+2+2+1.8+1.9)/5=1.74

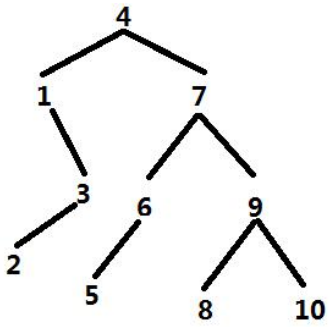
非剥夺式优先级调度算法：

作业执行次序	执行时间	优先级	等待时间	周转时间	带权周转时间
2	1	1	0	1	1
5	5	2	1	6	1.2
1	10	3	6	16	1.6
3	2	3	16	18	9
4	1	4	18	19	19

系统中作业的平均周转时间为 T=(1+6+16+18+19)/5=12

系统中作业的平均带权周转时间为 W=(1+1.2+1.6+9+19)/5=6.36

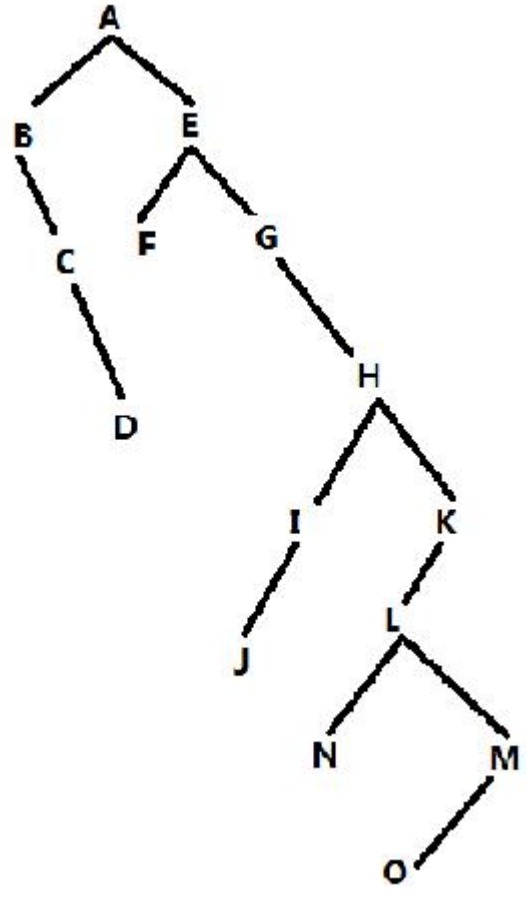
8、



知道先序后序不能重构二叉树

只有知道先序或者后序中的其中一个和中序一起才能重构二叉树

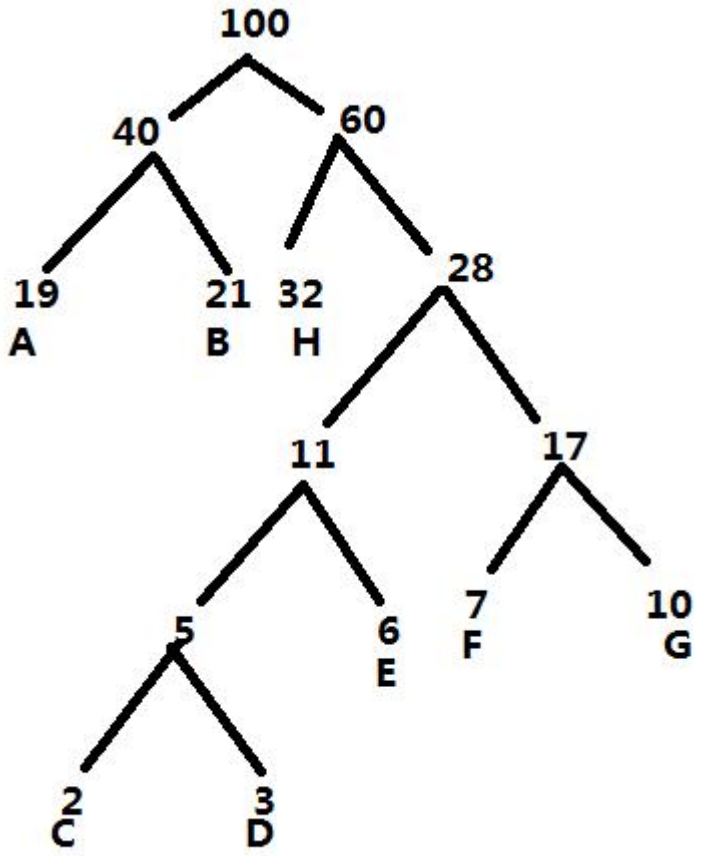
9、



先序：ABCDEFGHIJKLNMO

后序：DCBFJINOMLKHGEA

10、



A:00
B:01
C:11000
D:11001
E:1101
F:1110
G:1111
H:10

11、

又称普里姆算法，以图上的顶点为出发点，逐次选择到最小生成树顶点集距离最短的顶点为最小生成树的顶点，并加入到该顶点集，直到包含所有的顶点。

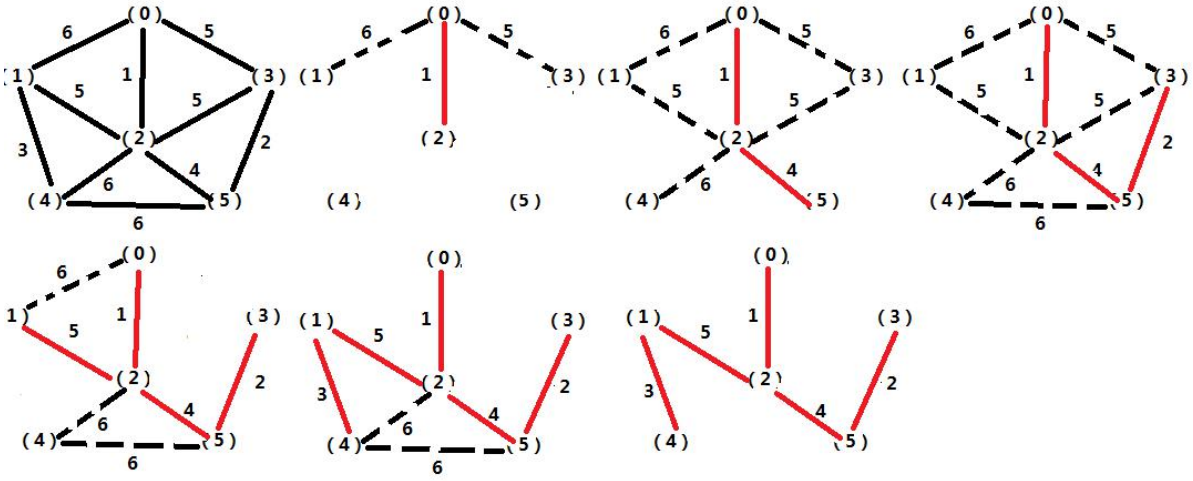
步骤：

1.选择一出发点，加入集合 A。

2.遍历与集合 A 中的点相邻的边，找到最短的边，并且不构成回路。

3.将步骤 2 得到的边的目标点加入集合 A。

4.重复 2，3 直到所有结点都加入到集合 A 中。



12、

（1）

1、选择一个入度为 0 的顶点并输出

2、从网中删除此顶点及所有出边

3、循环 1、2 直到所有顶点被去除或者无法找到入

度为 0 的顶点为止，结束后，若输出的顶点数小于网中的顶点数，则输出“有回路”信息，否则输出的顶点顺序就是拓扑序列。

（2）AEBCFD/AEFBCD/EABCFD/EAFBCD/EFABCD

13、Low=0 high=10

mid=[(1+10)/2]=5 r[5]=31 k>31 low=6

mid=[(6+10)/2]=8 r[8]=48 k>48 low=9

mid=[(9+10)/2]=9 r[9]=50 k>50 low=10

mid=[(10+10)/2]=10 r[10]=55 k>55 low=11

Low>high return 0

14、

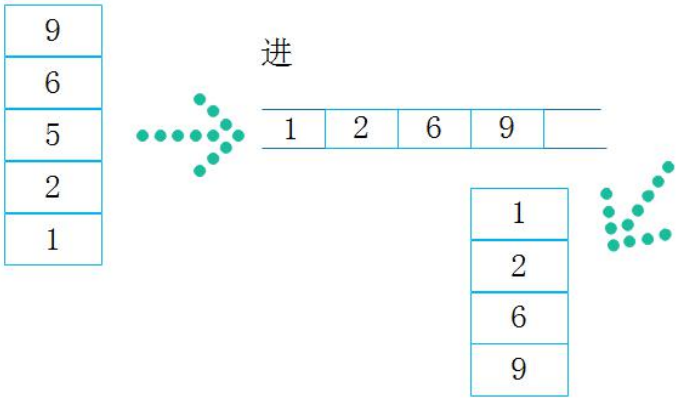
散列地址	0	1	2	3	4	5	6	7	8	9	10	11	12	13
关键字	55	01	14		27		19	20	84		23	11	10	77
比较次数	1	1	2		1		1	1	3		1	1	3	1

ASL=(1*8+2*1+2*3)/11=18/11

15、

```
DelElem(LinkList list){
pNode p,q,r;
For(p=list->Link;q!=NULL;q=q->Link){
r->Link;q->Mink;
free(q);
q=r;
}else{
r=r->Link;
}
}
```

16、（1）结果是：1269“1”为栈顶,过程如下图：



（2）demo 的功能：实现删除 e（5）元素，并实现栈 S 数据逆置

17、1.查询链表的尾结点
2.将第一个结点链接到链表尾部作为新的尾结点
3.返回线性表为(a2,a3,...,an,a1)

18、(10,18,4,3,6,12,1,9,18,8)
(8,18,4,3,6,12,1,9,18,10)
(8,10,4,3,6,12,1,9,18,18)
(8,9,4,3,6,12,1,10,18,18)
(8,9,4,3,6,10,1,12,18,18)
(8,9,4,3,6,1,10,12,18,18)
(1,9,4,3,6,8)10(12,18,18)
(1,8,4,3,6,9)10(12,18,18)
(1,4,3,8,6,9)10(12,18,18)
(1,4,3,6,8,9)10(12,18,18)
(1,4,3,6,8,9)10(12,18,18)
(1),(3),(4),(6),(8),(9),(10),(12),(18),(18)
19、DFS:13452
BFS:13245