100 个基本算法

1.数论算法

求两数的最大公约数

```
function gcd(a,b:integer):integer;
begin
if b=0 then gcd:=a
else gcd:=gcd (b,a mod b);
end ;
```

求两数的最小公倍数

```
function lcm(a,b:integer):integer;
begin
if a< b then swap(a,b);
lcm:=a;
while lcm mod b >0 do inc(lcm,a);
end;
```

素数的求法

A.小范围内判断一个数是否为质数：

```
function prime (n: integer): Boolean;
var I: integer;
begin
for I:=2 to trunc(sqrt(n)) do
if n mod I=0 then begin
prime:=false; exit;
end;
prime:=true;
end;
```

B.判断 longint 范围内的数是否为素数（包含求 50000 以内的素数表）：

```
procedure getprime;
var
i,j:longint;
p:array[1..50000] of boolean;
begin
fillchar(p,sizeof(p),true);
p[1]:=false;
i:=2;
while i< 50000 do begin
if p[i] then begin
j:=i*2;
while j< 50000 do begin
p[j]:=false;
inc(j,i);
end;
end;
inc(i);
end;
l:=0;
for i:=1 to 50000 do
if p[i] then begin
inc(l);pr[l]:=i;
```

```
end;
end;{getprime}

function prime(x:longint):integer;
var i:integer;
begin
prime:=false;
for i:=1 to l do
if pr[i] >=x then break
else if x mod pr[i]=0 then exit;
prime:=true;
end;{prime}
```

2.

3.

4.求最小生成树

A.Prim 算法：

```
procedure prim(v0:integer);
var
lowcost,closest:array[1..maxn] of integer;
i,j,k,min:integer;
begin
for i:=1 to n do begin
lowcost[i]:=cost[v0,i];
closest[i]:=v0;
end;
for i:=1 to n-1 do begin
{寻找离生成树最近的未加入顶点 k}
min:=maxlongint;
for j:=1 to n do
if (lowcost[j]< min) and (lowcost[j]< >0) then begin
min:=lowcost[j];
k:=j;
end;
lowcost[k]:=0; {将顶点 k 加入生成树}
{生成树中增加一条新的边 k 到 closest[k]}
{修正各点的 lowcost 和 closest 值}
for j:=1 to n do
if cost[k,j]< lwocost[j] then begin
lowcost[j]:=cost[k,j];
closest[j]:=k;
end;
end;
end;{prim}
```

B.Kruskal 算法：(贪心)

按权值递增顺序删去图中的边，若不形成回路则将此边加入最小生成树。

```
function find(v:integer):integer; {返回顶点 v 所在的集合}
var i:integer;
```

```
    begin
    i:=1;
    while (i< =n) and (not v in vset[i]) do inc(i);
    if i< =n then find:=i else find:=0;
    end;

    procedure kruskal;
    var
    tot,i,j:integer;
    begin
    for i:=1 to n do vset[i]:=[i];{初始化定义n个集合，第I个集合包含一个元素I}
    p:=n-1; q:=1; tot:=0; {p为尚待加入的边数，q为边集指针}
    sort;
    {对所有边按权值递增排序，存于e[I]中，e[I].v1与e[I].v2为边I所连接的两个顶点的序号，
e[I].len为第I条边的长度}
    while p >0 do begin
    i:=find(e[q].v1);j:=find(e[q].v2);
    if i< >j then begin
    inc(tot,e[q].len);
    vset[i]:=vset[i]+vset[j];vset[j]:=[];
    dec(p);
    end;
    inc(q);
    end;
    writeln(tot);
    end;
```

5.最短路径

A.标号法求解单源点最短路径：

```
    var
    a:array[1..maxn,1..maxn] of integer;
    b:array[1..maxn] of integer; {b[i]指顶点i到源点的最短路径}
    mark:array[1..maxn] of boolean;

    procedure bhf;
    var
    best,best_j:integer;
    begin
    fillchar(mark,sizeof(mark),false);
    mark[1]:=true; b[1]:=0;{1为源点}
    repeat
    best:=0;
    for i:=1 to n do
    If mark[i] then {对每一个已计算出最短路径的点}
    for j:=1 to n do
    if (not mark[j]) and (a[i,j] >0) then
    if (best=0) or (b[i]+a[i,j]< best) then begin
    best:=b[i]+a[i,j]; best_j:=j;
    end;
    if best >0 then begin
    b[best_j]:=best; mark[best_j]:=true;
```

```
    end;
    until best=0;
    end;{bhf}
```

B.Floyed 算法求解所有顶点对之间的最短路径：

```
    procedure floyed;
    begin
    for I:=1 to n do
    for j:=1 to n do
    if a[I,j] >0 then p[I,j]:=I else p[I,j]:=0; {p[I,j]表示 I 到 j 的最短路径上 j 的前驱结点}
    for k:=1 to n do {枚举中间结点}
    for i:=1 to n do
    for j:=1 to n do
    if a[i,k]+a[j,k]< a[i,j] then begin
    a[i,j]:=a[i,k]+a[k,j];
    p[I,j]:=p[k,j];
    end;
    end;
```

C. Dijkstra 算法：

类似标号法，本质为贪心算法。

```
    var
    a:array[1..maxn,1..maxn] of integer;
    b,pre:array[1..maxn] of integer; {pre[i]指最短路径上 I 的前驱结点}
    mark:array[1..maxn] of boolean;
    procedure dijkstra(v0:integer);
    begin
    fillchar(mark,sizeof(mark),false);
    for i:=1 to n do begin
    d[i]:=a[v0,i];
    if d[i]< >0 then pre[i]:=v0 else pre[i]:=0;
    end;
    mark[v0]:=true;
    repeat {每循环一次加入一个离 1 集合最近的结点并调整其他结点的参数}
    min:=maxint; u:=0; {u 记录离 1 集合最近的结点}
    for i:=1 to n do
    if (not mark[i]) and (d[i]< min) then begin
    u:=i; min:=d[i];
    end;
    if u< >0 then begin
    mark[u]:=true;
    for i:=1 to n do
    if (not mark[i]) and (a[u,i]+d[u]< d[i]) then begin
    d[i]:=a[u,i]+d[u];
    pre[i]:=u;
    end;
    end;
    until u=0;
    end;
```

D.计算图的传递闭包

```
Procedure Longlink;
Var
T:array[1..maxn,1..maxn] of boolean;
Begin
Fillchar(t,sizeof(t),false);
For k:=1 to n do
For I:=1 to n do
For j:=1 to n do T[I,j]:=t[I,j] or (t[I,k] and t[k,j]);
End;
```

6.0-1 背包问题（部分背包问题可有贪心法求解：计算 Pi/Wi)

数据结构：

w[i]:第 i 个背包的重量；

p[i]:第 i 个背包的价值；

(1)0-1 背包： 每个背包只能使用一次或有限次(可转化为一次)：

A.求最多可放入的重量。

NOIP2001 装箱问题

有一个箱子容量为 v(正整数，o≤v≤20000)，同时有 n 个物品(o≤n≤30)，每个物品有一个体积 (正整数)。要求从 n 个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

l 搜索方法

```
procedure search(k,v:integer); {搜索第 k 个物品，剩余空间为 v}
var i,j:integer;
begin
if v< best then best:=v;
if v-(s[n]-s[k-1]) >=best then exit; {s[n]为前 n 个物品的重量和}
if k< =n then begin
if v >w[k] then search(k+1,v-w[k]);
search(k+1,v);
end;
end;
```

l DP

F[I,j]为前 i 个物品中选择若干个放入使其体积正好为 j 的标志，为布尔型。

实现:将最优化问题转化为判定性问题

F[I,j]=f[i-1,j-w[i]] (w[I]< =j< =v) 边界：f[0,0]:=true.

```
For I:=1 to n do
For j:=w[I] to v do F[I,j]:=f[I-1,j-w[I]];
```

优化：当前状态只与前一阶段状态有关，可降至一维。

```
F[0]:=true;
For I:=1 to n do begin
F1:=f;
For j:=w[I] to v do
If f[j-w[I]] then f1[j]:=true;
F:=f1;
End;
```

B.求可以放入的最大价值。

F[I,j]=

C.求恰好装满的情况数。

(2)每个背包可使用任意次：

A.求最多可放入的重量。

状态转移方程为

f[I,j]=max{f[i-w[j]

B.求可以放入的最大价值。

USACO 1.2 Score Inflation

进行一次竞赛，总时间T固定，有若干种可选择的题目，每种题目可选入的数量不限，每种题目有一个ti（解答此题所需的时间）和一个si（解答此题所得的分数），现要选择若干题目，使解这些题的总时间在T以内的前提下，所得的总分最大，求最大的得分。

*易想到：

f[i,j] = max { f [i- k*w[j], j-1] + k*v[j] } (0< =k< = i div w[j])

其中f[i,j]表示容量为i时取前j种背包所能达到的最大值。

*优化：

```
Begin
FillChar(problem,SizeOf(problem),0);
Assign(Input,'inflate.in');
Reset(Input);
Readln(M,N);
For i:=1 To N Do
With problem[i] Do
Readln(point,time);
Close(Input);

FillChar(f,SizeOf(f),0);
For i:=1 To M Do
For j:=1 To N Do
If i-problem[j].time >=0 Then
Begin
t:=problem[j].point+f[i-problem[j].time];
If t >f[i] Then f[i]:=t;
End;

Assign(Output,'inflate.out');
Rewrite(Output);
Writeln(f[M]);
Close(Output);
End.
```

C.求恰好装满的情况数。

AhOI2001 Problem2

求自然数n本质不同的质数和的表达式的数目。

思路一，生成每个质数的系数的排列，在一一测试，这是通法。

```
procedure try(dep:integer);
var i,j:integer;
begin
cal; {此过程计算当前系数的计算结果，now为结果}
if now >n then exit; {剪枝}
```

```
if dep=l+1 then begin {生成所有系数}
cal;
if now=n then inc(tot);
exit;
end;
for i:=0 to n div pr[dep] do begin
xs[dep]:=i;
try(dep+1);
xs[dep]:=0;
end;
end;
```

思路二，递归搜索效率较高

```
procedure try(dep,rest:integer);
var i,j,x:integer;
begin
if (rest< =0) or (dep=l+1) then begin
if rest=0 then inc(tot);
exit;
end;
for i:=0 to rest div pr[dep] do
try(dep+1,rest-pr[dep]*i);
end;
```

思路三：可使用动态规划求解

USACO1.2 money system

V 个物品，背包容量为 n，求放法总数。

转移方程：

```
Procedure update;
var j,k:integer;
begin
c:=a;
for j:=0 to n do
if a[j] >0 then
for k:=1 to n div now do
if j+now*k< =n then inc(c[j+now*k],a[j]);
a:=c;
end;
{main}
begin
read(now); {读入第一个物品的重量}
i:=0; {a[i]为背包容量为 i 时的放法总数}
while i< =n do begin
a[i]:=1; inc(i,now); end; {定义第一个物品重的整数倍的重量 a 值为 1，作为初值}
for i:=2 to v do
begin
read(now);
update; {动态更新}
end;
writeln(a[n]);
```

7.排序算法
A.快速排序：

```
procedure sort(l,r:integer);
var i,j,mid:integer;
begin
i:=l;j:=r; mid:=a[(l+r) div 2]; {将当前序列在中间位置的数定义为中间数}
repeat
while a[i]< mid do inc(i); {在左半部分寻找比中间数大的数}
while mid< a[j] do dec(j);{在右半部分寻找比中间数小的数}
if i< =j then begin {若找到一组与排序目标不一致的数对则交换它们}
swap(a[i],a[j]);
inc(i);dec(j); {继续找}
end;
until i >j;
if l< j then sort(l,j); {若未到两个数的边界，则递归搜索左右区间}
if i< r then sort(i,r);
end;{sort}
```

B.插入排序：

```
procedure insert_sort(k,m:word); {k为当前要插入的数，m为插入位置的指针}
var i:word; p:0..1;
begin
p:=0;
for i:=m downto 1 do
if k=a[i] then exit;
repeat
If k >a[m] then begin
a[m+1]:=k; p:=1;
end
else begin
a[m+1]:=a[m]; dec(m);
end;
until p=1;
end;{insert_sort}
```

l 主程序中为：

```
a[0]:=0;
for I:=1 to n do insert_sort(b[I],I-1);
```

C.选择排序：

```
procedure sort;
var i,j,k:integer;
begin
for i:=1 to n-1 do begin
k:=i;
for j:=i+1 to n do
if a[j]< a[k] then k:=j; {找出a[I]..a[n]中最小的数与a[I]作交换}
if k< >i then begin
a[0]:=a[k];a[k]:=a[i];a[i]:=a[0];
end;
end;
end;
```

D. 冒泡排序

```
procedure sort;
var i,j,k:integer;
begin
for i:=n downto 1 do
for j:=1 to i-1 do
if a[j] >a[i] then begin
a[0]:=a[i];a[i]:=a[j];a[j]:=a[0];
end;
end;
```

E.堆排序:

```
procedure sift(i,m:integer);{调整以 i 为根的子树成为堆,m 为结点总数}
var k:integer;
begin
a[0]:=a[i]; k:=2*i;{在完全二叉树中结点 i 的左孩子为 2*i,右孩子为 2*i+1}
while k< =m do begin
if (k< m) and (a[k]< a[k+1]) then inc(k);{找出 a[k]与 a[k+1]中较大值}
if a[0]< a[k] then begin a[i]:=a[k];i:=k;k:=2*i; end
else k:=m+1;
end;
a[i]:=a[0]; {将根放在合适的位置}
end;

procedure heapsort;
var
j:integer;
begin
for j:=n div 2 downto 1 do sift(j,n);
for j:=n downto 2 do begin
swap(a[1],a[j]);
sift(1,j-1);
end;
end;
```

F. 归并排序

```
{a 为序列表，tmp 为辅助数组}
procedure merge(var a:listtype; p,q,r:integer);
{将已排序好的子序列 a[p..q]与 a[q+1..r]合并为有序的 tmp[p..r]}
var I,j,t:integer;
tmp:listtype;
begin
t:=p;i:=p;j:=q+1;{t 为 tmp 指针，I,j 分别为左右子序列的指针}
while (t< =r) do begin
if (i< =q){左序列有剩余} and ((j >r) or (a[i]< =a[j])) {满足取左边序列当前元素的要
求}
then begin
tmp[t]:=a[i]; inc(i);
end
else begin
tmp[t]:=a[j];inc(j);
end;
```

```
inc(t);
end;
for i:=p to r do a[i]:=tmp[i];
end;{merge}

procedure merge_sort(var a:listtype; p,r: integer); {合并排序a[p..r]}
var q:integer;
begin
if p< >r then begin
q:=(p+r-1) div 2;
merge_sort (a,p,q);
merge_sort (a,q+1,r);
merge (a,p,q,r);
end;
end;
{main}
begin
merge_sort(a,1,n);
end.
```

G.基数排序
思想：对每个元素按从低位到高位对每一位进行一次排序

8.高精度计算
A.
B.
C.
D.

9.树的遍历顺序转换
A. 已知前序中序求后序

```
procedure Solve(pre,mid:string);
var i:integer;
begin
if (pre='') or (mid='') then exit;
i:=pos(pre[1],mid);
solve(copy(pre,2,i),copy(mid,1,i-1));
solve(copy(pre,i+1,length(pre)-i),copy(mid,i+1,length(mid)-i));
post:=post+pre[1]; {加上根，递归结束后post即为后序遍历}
end;
```

B.已知中序后序求前序

```
procedure Solve(mid,post:string);
var i:integer;
begin
if (mid='') or (post='') then exit;
i:=pos(post[length(post)],mid);
pre:=pre+post[length(post)]; {加上根，递归结束后pre即为前序遍历}
solve(copy(mid,1,I-1),copy(post,1,I-1));
solve(copy(mid,I+1,length(mid)-I),copy(post,I,length(post)-i));
```

```
end;
```

C.已知前序后序求中序

```
function ok(s1,s2:string):boolean;
var i,l:integer; p:boolean;
begin
ok:=true;
l:=length(s1);
for i:=1 to l do begin
p:=false;
for j:=1 to l do
if s1[i]=s2[j] then p:=true;
if not p then begin ok:=false;exit;end;
end;
end;

procedure solve(pre,post:string);
var i:integer;
begin
if (pre='') or (post='') then exit;
i:=0;
repeat
inc(i);
until ok(copy(pre,2,i),copy(post,1,i));
solve(copy(pre,2,i),copy(post,1,i));
midstr:=midstr+pre[1];
solve(copy(pre,i+2,length(pre)-i-1),copy(post,i+1,length(post)-i-1));
end;
```

10.求图的弱连通子图(DFS)

```
procedure dfs ( now,color: integer);
begin
for i:=1 to n do
if a[now,i] and c[i]=0 then begin
c[i]:=color;
dfs(I,color);
end;
end;
```

11.拓扑排序

寻找一数列，其中任意连续 p 项之和为正，任意 q 项之和为负，若不存在则输出 NO.

12.进制转换

A.整数任意正整数进制间的互化

NOIP1996 数制转换

设字符串 A$的结构为: A$='mp'

其中 m 为数字串(长度< =20),而 n,p 均为 1 或 2 位的数字串(其中所表达的内容在 2-10 之间)

程序要求:从键盘上读入A$后(不用正确性检查),将A$中的数字串m(n进制)以p进制的形式输出.
例如:A$='48< 10 >8'
其意义为:将10进制数48,转换为8进制数输出.
输出结果:48< 10 >=60< 8 >

B.实数任意正整数进制间的互化
C.负数进制:
NOIP2000
设计一个程序，读入一个十进制数的基数和一个负进制数的基数，并将此十进制数转换为此负 进制下的数：-R∈{-2，-3，-4,....-20}

13.全排列与组合的生成
排列的生成：（1..n）

```
procedure solve(dep:integer);
var
i:integer;
begin
if dep=n+1 then begin writeln(s);exit; end;
for i:=1 to n do
if not used[i] then begin
s:=s+chr(i+ord('0'));used[i]:=true;
solve(dep+1);
s:=copy(s,1,length(s)-1); used[i]:=false;
end;
end;
```

组合的生成(1..n中选取k个数的所有方案)

```
procedure solve(dep,pre:integer);
var
i:integer;
begin
if dep=k+1 then begin writeln(s);exit; end;
for i:=1 to n do
if (not used[i]) and (i >pre) then begin
s:=s+chr(i+ord('0'));used[i]:=true;
solve(dep+1,i);
s:=copy(s,1,length(s)-1); used[i]:=false;
end;
end;
```

14 递推关系
计算字串序号模型
USACO1.2.5 StringSobits
长度为N (N< =31)的01串中1的个数小于等于L的串组成的集合中找出按大小排序后的第I个01串。

数字划分模型
*noiP2001数的划分

将整数 n 分成 k 份，且每份不能为空，任意两种分法不能相同(不考虑顺序)。

```
d[0,0]:=1;
for p:=1 to n do
for i:=p to n do
for j:=k downto 1 do inc(d[i,j],d[i-p,j-1]);
writeln(d[n,k]);
```

*变形 1：考虑顺序

d[i, j] : = d [i-k, j-1] (k=1..i)

*变形 2：若分解出来的每个数均有一个上限 m

d[i, j] : = d [i-k, j-1] (k=1..m)

15.算符优先法求解表达式求值问题

```
const maxn=50;
var
s1:array[1..maxn] of integer; {s1 为数字栈}
s2:array[1..maxn] of char; {s2 为算符栈}
t1,t2:integer; {栈顶指针}

procedure calcu;
var
x1,x2,x:integer;
p:char;
begin
p:=s2[t2]; dec(t2);
x2:=s1[t1]; dec(t1);
x1:=s1[t1]; dec(t1);
case p of
'+':x:=x1+x2;
'-':x:=x1-x2;
'*':x:=x1*x2;
'/':x:=x1 div 2;
end;
inc(t1);s1[t1]:=x;
end;

procedure work;
var c:char;v:integer;
begin
t1:=0;t2:=0;
read(c);
while c< >';' do
case c of
'+','-': begin
while (t2 >0) and (s2[t2]< >'(') do calcu;
inc(t2);s2[t2]:=c;
read(c);
end ;
'*','/':begin
if (t2 >0) and ((s2[t2]='*') or (s2[t2]='/')) then calcu;
inc(t2);s2[t2]:=c;
```

```
read(c);
end;
'(':begin inc(t2); s2[t2]:=c; read(c); end;
')':begin
while s2[t2]< >'(' do calcu;
dec(t2); read(c);
end;
'0'..'9':begin
v:=0;
repeat
v:=10*v+ord(c)-ord('0');
read(c);
until (c< '0') or (c >'9');
inc(t1); s1[t1]:=v;
end;
end;
while t2 >0 do calcu;
writeln(s1[t1]);
end;
```

16.查找算法

折半查找

```
function binsearch(k:keytype):integer;
var low,hig,mid:integer;
begin
low:=1;hig:=n;
mid:=(low+hig) div 2;
while (a[mid].key< >k) and (low< =hig) do begin
if a[mid].key >k then hig:=mid-1
else low:=mid+1;
mid:=(low+hig) div 2;
end;
if low >hig then mid:=0;
binsearch:=mid;
end;
```

树形查找

二叉排序树：每个结点的值都大于其左子树任一结点的值而小于其右子树任一结点的值。

查找

```
function treesrh(k:keytype):pointer;
var q:pointer;
begin
q:=root;
while (q< >nil) and (q^.key< >k) do
if k< q^.key then q:=q^.left
else q:=q^.right;
treesrh:=q;
end;
```

17. KMP 算法

18．贪心

*会议问题

（1） n 个活动每个活动有一个开始时间和一个结束时间，任一时刻仅一项活动进行，求满足活动数最多的情况。

解：按每项活动的结束时间进行排序，排在前面的优先满足。

（2）会议室空闲时间最少。

（3）每个客户有一个愿付的租金，求最大利润。

（4）共 R 间会议室，第 i 个客户需使用 i 间会议室，费用相同，求最大利润。

附录 1 常用技巧

1.带权中位数

我国蒙古大草原上有 N（N 是不大于 100 的自然数）个牧民定居点 P1（X1，Y1）、P2（X2，Y2）、…Pn（Xn，Yn），相应地有关权重为 Wi，现在要求你在大草原上找一点 P（Xp，Yp），使 P 点到任 一点 Pi 的距离 Di 与 Wi 之积之和为最小。

即求 D=W1*D1+W2*D2+…+Wi*Di+…+Wn*Dn 有最小值

结论：对 x 与 y 两个方向分别求解带权中位数，转化为一维。

设最佳点 p 为点 k，则点 k 满足：

令 W 为点 k 到其余各点的带权距离之和，则

sigema(i=1 to k-1) Wi*Di < = W/2

sigema(i=k+1 to n) Wi*Di < = W/2

同时满足上述两式的点 k 即为带权中位数。

2．求一序列中连续子序列的最大和

```
begin
maxsum:=-maxlongint;
sum:=0;
for i:=1 to n do begin
inc(sum,data[i]);
if sum >maxsum then maxsum:=sum;
if sum< 0 then sum:=0;
end;
writeln(maxsum);
end;
```

3.

附录 2 数据结构相关操作

1．链表的定位函数 loc(I:integer):pointer; {寻找链表中的第 I 个结点的指针}

```
procedure loc(L:linklist; I:integer):pointer;
var p:pointer;
j:integer;
begin
p:=L.head; j:=0;
if (I >=1) and (I< =L.len) then
while j< I do begin p:=p^.next; inc(j); end;
loc:=p;
end;
```

2．单链表的插入操作

```
procedure insert(L:linklist; I:integer; x:datatype);
var p,q:pOInter;
begin
p:=loc(L,I);
new(q);
q^.data:=x;
q^.next:=p^.next;
p^.next:=q;
inc(L.len);
end;
```

3．单链表的删除操作

```
procedure delete(L:linklist; I:integer);
var p,q:pOInter;
begin
p:=loc(L,I-1);
q:=p^.next;
p^.next:=q^.next;
dispose(q);
dec(L.len);
end;
```

4．双链表的插入操作（插入新结点 q）

```
p:=loc(L,I);
new(q);
q^.data:=x;
q^.pre:=p;
q^.next:=p^.next;
p^.next:=q;
q^.next^.pre:=q;
```

5．双链表的删除操作

```
p:=loc(L,I); {p 为要删除的结点}
p^.pre^.next:=p^.next;
p^.next^.pre:=p^.pre;
dispose(p);
```

初赛的内容增加

在初赛的内容个增加以下内容：
数据结构 1．指针类型 2．多维数组 3．单链表及循环链表 4．二叉树 5．文件操作（从文本文件中读入数据，并输出到文本文件中）
程序设计 1．算法的实现能力 2．程序调试基本能力 3．设计测试数据的基本能力 4．程序的时间复杂度和空间复杂度的估计
算法处理 1．离散数学知识的应用（如排列组合、简单图论、数理逻辑）2．分治思想 3．模拟法 4．贪心法 5．简单搜索算法（深度优先 广度优先）搜索中的剪枝 6．动态规划的思想及基本算法

（一） 初赛内容与要求：
计 基算 本机 常的 识 1．计算机和信息社会（信息社会的主要特征、计算机的主要特征、数字通信网络的主要特征、数字化）2．信息输入输出基本原理（信息交换环境、文字图形多媒体信息的输入输出方式）3．信息的表示与处理（信息编码、微处理部件 MPU、内存储结构、指令、程序，和存储程序原理、程序的三种基本控制结构）4．信息的存储、组织与管理（存储介质、存储器结构、文件管理、数据库管理）5．信息系统组成及互连网的基本知识（计算机构成原理、槽和端口的部件间可扩展互连方式、层次式的互连结构、互联网络、TCP/IP 协议、HTTP 协议、WEB 应用的主要方式和特点）6．人机交互界面的基本概念（窗口系统、人和计算机交流信息的途径（文件及交互操作））7．信息技术的新发展、新特点、新应用等。
计 基算 本机 操的 作 1．WINDOWS 和 LINUX 的基本操作知识 2．互联网的基本使用常识（网上的浏览、搜索和查询等）3．常用的工具软件使用（文字编辑、电子邮件收发等）
程序设计的基本知识 数据结构 1．程序语言中基本数据类型（字符、整数、长整数、浮点）2．浮点运算中的精度和数值比较 3．一维数组（串）与线性表 4．记录类型（PASCAL）/结构类型（C）
程序设计 1．结构化程序设计的基本概念 2．阅读理解程序的基本能力 3．具有将简单问题抽象成适合计算机解决的模型的基本能力 4．具有针对模型设计简单算法的基本能力 5．程序流程描述（自然语言/伪码/NS 图/其他）6．程序设计语言（pascal/C/C++，2003 仍允许 BASIc）
基本算法处 理 1．初等算法（计数、统计、数学运算等）2．排序算法（冒泡法、插入排序、合并排序、快速排序）3．查找（顺序查找、二分法）4．回溯算法
（二）复赛内容与要求：

在初赛的内容个增加以下内容：
数据结构 1．指针类型 2．多维数组 3．单链表及循环链表 4．二叉树 5．文件操作（从文本文件中读入数据，并输出到文本文件中）
程序设计 1．算法的实现能力 2．程序调试基本能力 3．设计测试数据的基本能力 4．程序的时间复杂度和空间复杂度的估计
算法处理 1．离散数学知识的应用（如排列组合、简单图论、数理逻辑）2．分治思想 3．模拟法 4．贪心法 5．简单搜索算法（深度优先 广度优先）搜索中的剪枝 6．动态规划的思想及基本算法