

2015 年初赛真题

2:

CPU 由运算器、控制器，寄存器组成。

7:

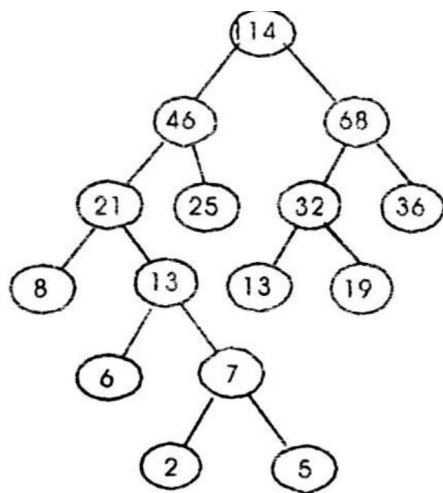
前序遍历根结点在最开始，后序遍历根结点在最后面。故前序遍历和后序遍历相同的二叉树只有根结点。

12:

哈夫曼算法：贪心策略：每次选择权值最小的子树构造新的二叉树。

哈夫曼树：给定 N 个权值作为 N 个叶子结点，构造一棵二叉树，若该树的带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树。哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。

哈夫曼树的叶结点比非叶结点多 1



13:

llink 和 rlink 分别指向前驱和后继,不妨设 p 的前驱为 o ,在未插入前 $p \rightarrow \text{llink}$ 就是 o , $o \rightarrow \text{rlink}$ 就是 p ,插入时,先将 $o \rightarrow \text{rlink}$ 赋为 q ,再将 $q \rightarrow \text{rlink}$ 赋为 p ,然后将 $q \rightarrow \text{llink}$ 赋为 o ,最后将 $p \rightarrow \text{llink}$ 赋为 q 。

答案：

单选

1-5 AAADD

6-10 BBBBD

11-15 DADAA

多选

1、ABCD

2、ABC

3、ACD

4、AB

5、AC

课堂内容

1、P1098

字符串的模拟题

可以大致分为四类：

第一类：不是 ‘-’ 就可以直接输出

第二类： ‘-’ 两边 `ascill` 相差 1，就是两个字符相邻，直接把 ‘-’ 跳过

第三类： ‘-’ 两边都是字母

第四类： ‘-’ 两边都是数字

第三类和第四类都要遵循以下规则：

`p1=1`，填充小写字母；`p1=2`，填充大写字母；`p1=3`，填充 ‘*’

`P2=k`，连续填充 `k` 个字符

`P3=1`，顺序输出；`p3=2`，逆序输出

完整代码：

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  string s,s1;
4  int p1,p2,p3;
5  int len;
6
7  int main() {
8      cin >> p1 >> p2 >> p3;
9      cin>>s;
10     len=s.length();
11     for(int i=0; i<len; i++) {
12         //1. 字符不为-, 直接输出, 跳过
13         if(s[i]!='-') {
14             cout<<s[i];
15             continue;
16         }
17         //2. 字符为-, 但前后相差1, 跳过
18         if(s[i+1]-s[i-1]==1) continue;
19         //3. 字符为-, 前后不相差1, 前后都为数字时
20         if(s[i-1]>='0'&&s[i-1]<='9'&&s[i+1]>='0'&&s[i+1]<='9') {
21             //3.1 后面的数小于等于前面的, 直接输出, 跳过
22             if(s[i+1]<=s[i-1]) {
23                 cout<<s[i];
24                 continue;
25             }
```

```

26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

//3.2 P1等于3, 输出P2个*号
if(p1==3) {
    for(int j=s[i-1]+1; j<s[i+1]; j++) {
        for(int k=1; k<=p2; k++) {
            cout<<"*";
        }
        continue;
    }
}

//3.3 P3等于1, 顺序输出
if(p3==1) {
    for(int j=s[i-1]+1; j<s[i+1]; j++) {
        for(int k=1; k<=p2; k++) {
            cout<<char(j);
        }
        continue;
    }
}

//3.4 P3等于2, 逆序输出
if(p3==2) {
    for(int j=s[i+1]-1; j>s[i-1]; j--) {
        for(int k=1; k<=p2; k++) {
            cout<<char(j);
        }
        continue;
    }
}

// 4. 字符为-, 前后不相差1, 前后都为字母时
if(s[i-1]>='a'&& s[i-1]<='z'&& s[i+1]>='a'&& s[i+1]<='z') {
    //4.1 后面的字母小于等于前面的, 直接输出, 跳过
    if(s[i+1]<=s[i-1]) {
        cout<<s[i];
        continue;
    }
    //4.2 P1等于3, 输出P2个*号
    if(p1==3) {
        for(int j=s[i-1]+1; j<s[i+1]; j++) {
            for(int k=1; k<=p2; k++) {
                cout<<"*";
            }
            continue;
        }
    }
    //4.3 P3等于1, 顺序输出
    if(p3==1) {
        //小写字母
        if(p1==1) {
            for(int j=s[i-1]+1; j<s[i+1]; j++) {
                for(int k=1; k<=p2; k++) {
                    cout<<char(j);
                }
            }
            continue;
        }
    }
}

```

```

81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
}

//大写字母
if(p1==2) {
    for(int j=s[i-1]+1; j<s[i+1]; j++) {
        for(int k=1; k<=p2; k++) {
            cout<<char(j-32);
        }
    }
    continue;
}

//4.4 P3等于2, 逆序输出
if(p3==2) {
    //小写字母
    if(p1==1) {
        for(int j=s[i+1]-1; j>s[i-1]; j--) {
            for(int k=1; k<=p2; k++) {
                cout<<char(j);
            }
        }
        continue;
    }
    //大写字母
    if(p1==2) {
        for(int j=s[i+1]-1; j>s[i-1]; j--) {
            for(int k=1; k<=p2; k++) {
                cout<<char(j-32);
            }
        }
        continue;
    }
}

cout<<s[i];
return 0;
}

```

2、P1006

题解：

首先，题目意思是从左上角传到右下角，再从右下角传到左上角，中间不能有重复的点。相当于从左上角到右下角传两次，不能有重复的点，这样就跟 P1004 思路是一样的。

可以用四维数组 $f[i][j][k][l]$, i 和 j 表示第一次遍历走的点的横纵坐标, k 和 l 表示第二次走的点的横纵坐标, 状态转移方程为:

① $f[i][j][k][l] = \max(f[i-1][j][k-1][l], \max(f[i][j-1][k-1][l], \max(f[i-1][j][k][l-1], f[i][j-1][k][l-1]))) + a[i][j] + a[k][l];$ ($i \neq k, j \neq l$)

② $f[i][j][k][l] = \max(f[i-1][j][k-1][l], \max(f[i][j-1][k-1][l], \max(f[i-1][j][k][l-1], f[i][j-1][k][l-1]))) + a[i][j];$ ($i=k, j=l$) 同样的点只选一次

完整代码:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int n,m,f[60][60][60][60],a[60][60];
4  int main(){
5      cin >> n ;
6      int p,q,num;
7      while(cin >> p >> q >> num){
8          if(p==0&&q==0&&num==0)
9              break;
10         a[p][q]=num;
11     }
12     for(int i=1;i<=n;i++){
13         for(int j=1;j<=n;j++){
14             for(int k=1;k<=n;k++){
15                 for(int l=1;l<=n;l++){
16                     f[i][j][k][l]=max(f[i-1][j][k-1][l],max(f[i][j-1][k-1][l],
17                     max(f[i-1][j][k][l-1],f[i][j-1][k][l-1]))) + a[i][j] + a[k][l];
18                     if(i==k&&j==l){
19                         f[i][j][k][l] -= a[i][j];
20                     }
21                 }
22             }
23         }
24     }
25     cout << f[n][n][n][n];
26     return 0;
27 }
```

3、P1149

题解: 先确定 n ($n < 2000$) 需要的火柴数, 遍历 a, b , 如果 a, b 和 $a+b$ 需要的火柴数+4 (等号和加号共需要四个火柴) = 总共的火柴数, 计数加一。

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int n;
4  int cnt[2001];
5  int a[10]={6,2,5,5,4,5,6,3,7,6};
6  int sum;
7
8  int main(){
9      cnt[0] = 6;
10     cin >> n;
11     for(int i=1;i<=2000;i++){
12         int temp = i;
13         //先计算每个数所用的火柴棒
14         while(temp > 0){
15             cnt[i] += a[temp%10];
16             temp = temp/10;
17         }
18     }
19     for(int i=0;i<=1000;i++){
20         for(int j=0;j<=1000;j++){
21             if(cnt[i]+cnt[j]+cnt[i+j]+4==n){
22                 sum++;
23             }
24         }
25     }
26     cout << sum;
27     return 0;
28 }

```

4、P1167

对于有限时间内尽可能多的刷题，只需要从用时短的题开始做，用贪心算法即可。所以麻烦的是求起始时间和终止时间的时间差，多少分钟。额外判断 2 月和闰年的情况即可。


```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int n, year1, year2, mon1, mon2, day1, day2, hour1, hour2, min1, min2;
5  int mon[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
6  long long ans1, ans2, ans;
7  char c1;
8  int _time[5010];
9  bool isRun(int year){
10     if(year % 400 == 0 || year % 4 == 0 && year % 100 != 0)
11         return true;
12     else return false;
13 }
14 int main(){
15     cin >> n;
16     for(int i = 1; i <= n; i++){
17         cin >> _time[i];
18     }
19     sort(_time + 1, _time + 1 + n);
20     cin >> year1 >> c1 >> mon1 >> c1 >> day1 >> c1 >> hour1 >> c1 >> min1;
21     cin >> year2 >> c1 >> mon2 >> c1 >> day2 >> c1 >> hour2 >> c1 >> min2;
22     ans1 += min1 + hour1 * 60 + (day1 - 1) * 1440;
23     ans2 += min2 + hour2 * 60 + (day2 - 1) * 1440;
24     for(int i = 1; i < mon1; i++){
25         if(i == 2){
26             if(isRun(year1)){
27                 ans1 += 29 * 1440;
28             }
29             else ans1 += 28 * 1440;
30         }
31         else ans1 += mon[i] * 1440;
32     }
33     for(int i = 1; i < mon2; i++){
34         if(i == 2){
35             if(isRun(year2)){
36                 ans2 += 29 * 1440;
37             }
38             else ans2 += 28 * 1440;
39         }
40         else ans2 += mon[i] * 1440;
41     }
42     for(int i = year1; i < year2; i++){
43         if(isRun(i)){
44             ans2 += 366 * 1440;
45         }
46         else ans2 += 365 * 1440;
47     }
48     ans = ans2 - ans1;
49     int t = 0;
50     for(int i = 1; i <= n; i++){
51         if(ans >= _time[i]){
52             ans -= _time[i];
53             t++;
54         }
55         else break;
56     }
57     cout << t;
58     return 0;
59 }
60

```


5、P2058

思路：用队列存每个人的“国籍和到达时间”即可，用一个标记数组记录每种国籍的人数。然后每到达一艘船，取队头元素出来判断即可，如果队头的人到达的时间跟这艘船的到达时间差没超过 24 小时，说明队列里的人不需要减少，接着再统计此船新出现的国籍数即可。如果时间差超过了 24 小时，那么就从队头开始减少队列里的人，直到减少到第一个 24 小时内的人即可。

同时注意每次输出的国籍数 ans 是可以累加的，不需要清零，当某个国籍的人数刚好为 1 时，ans++，当某个国籍的人数降到 0 时，ans--。

```
4   int n, t, m, x;
5   int tong[100005];
6   int ans;
7   struct node
8   {
9       int s, t;
10  };
11  int main()
12  {
13      queue<node> ship;
14      node h;
15      cin >> n;
16      for(int i=1;i<=n;i++)
17      {
18          cin >> t >> m;
19          while(!ship.empty())
20          {
21              h = ship.front();
22              if(h.t + 86400 <= t)
23              {
24                  tong[h.s]--;
25                  if(tong[h.s]==0){
26                      ans--;
27                  }
28                  ship.pop();
29                  continue;
30              }
31              break;
32          }
33          for(int j=1;j<=m;j++)
34          {
35              cin >> x;
36              h.s = x;
37              h.t = t;
38              ship.push(h);
39              tong[x]++; // 桶排
40              if(tong[x]==1){
41                  ans++;
42              }
43          }
44          cout << ans << endl;
45      }
46      return 0;
```

6、P1154

题解：如果 a 、 b 对 k 的求余结果相同，那么 $a-b$ 一定是 k 的整数倍。

所以可以先把 $a-b$ 的差值全部标记出来，然后枚举模 k ，如果 k 没有被标记，并且 k 的倍数也没有被标记，那么这个 k 就是我们要求的结果

```
6  #include<bits/stdc++.h>
7  using namespace std;
8  int a[5010];
9  bool vis[1000010]; //标记a-b
10 int n;
11
12 int main(){
13     cin >> n;
14     for (int i = 0; i < n; i++){
15         cin >> a[i];
16     }
17     //标记所有数之间的差值a-b
18     for (int i = 0; i < n - 1; i++){
19         for (int j = i + 1; j < n; j++){
20             vis[abs(a[i] - a[j])] = true;
21         }
22     }
23     //从小到大枚举模k
24     for (int i = n; i < 1000010; i++){
25         //i没有被标记
26         if (!vis[i]){
27             bool flag = true;
28             //判断i的整数倍是否被标记
29             for (int j = i + i; j < 1000010; j += i){
30                 if (vis[j]){
31                     flag = false;
32                     break;
33                 }
34             }
35             //如果i和i的整数倍都没有被标记，那么输出
36             if (flag){
37                 cout << i << endl;
38                 break;
39             }
40         }
41     }
42     return 0;
}
```

7、P1158

题解：我们尽可能的让系统 2 去拦截离系统 1 远的，然后系统 1 去拦截剩余导弹的最远距离，这样一直循环，不断判断，取两个系统工作半径和的最小值。

首先计算出所有导弹到两个系统的距离，然后根据到系统 1 的距离从远到近排序，遍历导弹，temp 来更新 0...i-1 号导弹到系统 2 的最远距离，s[i].s1 就是 i...n 号导弹中离系统 1 最远的，我们所求的工作半径平方和的最小值就是 temp+s[i].s1 的最小值。

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4  int x1,y1,x2,y2,n,minn=1e9;
5  int a,b,temp;
6  struct node{
7      int s1;
8      int s2;
9  }s[100010];
10
11 bool cmp(node x ,node y){
12     return x.s1 > y.s1;
13 }
14
15 int main(){
16     cin >> x1 >> y1 >> x2 >> y2 >> n;
17     for(int i=1;i<=n;i++){
18         cin >> a >> b;
19         s[i].s1=(a-x1)*(a-x1)+(b-y1)*(b-y1); //此导弹到系统1的距离平方
20         s[i].s2=(a-x2)*(a-x2)+(b-y2)*(b-y2); //此导弹到系统2的距离平方
21     }
22     sort(s+1,s+n+1,cmp);
23     for(int i=1;i<=n;i++){
24         temp = max(temp,s[i-1].s2); //系统2拦之前遍历过的导弹
25         minn = min(minn,s[i].s1+temp); //系统1拦剩下的
26     }
27     cout<<minn<<endl;
28     return 0;
29 }
```