

8.8 课堂笔记 30

初赛练习

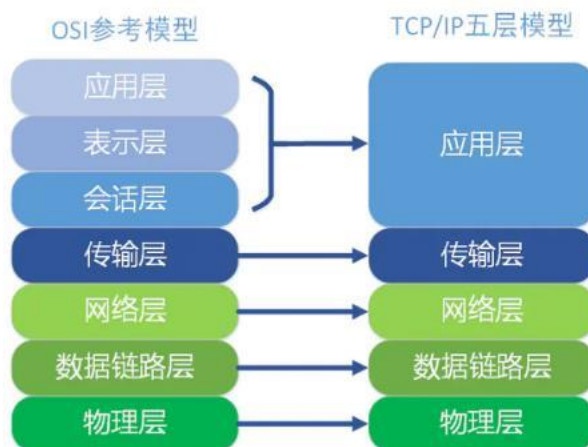
NOIP 提高组 2012 年真题解析

单选：

3：

英特尔公司是美国一家主要以研制 CPU 处理器的公司，是全球最大的个人计算机零件和 CPU 制造商，它成立于 1968 年，具有 46 年产品创新和市场领导的历史。1971 年，英特尔推出了全球第一个微处理器。

4：



5：

快排由于之前一直选取的是第一个元素，所以当遇到特殊输入，比如太大或者太小，就会造成区间划分极度不合理。引入随机化，就是在每一次划分的时候随机选取一个元素作为关键字，用它来进行划分。由于每一次划分都是随机选取，所以每一次都选到不好的元素概率低，可以作为一个优化的方向。

6：

ENIAC 是世界上第一台通用计算机，也是继 ABC（阿塔纳索夫-贝瑞计算机）之后的第二台电子计算机。所以是电子管。

8：

2 的 16 次方 = 65536 = 64KB 的内存地址，而一个 32 位地址总线可以寻址到 4,294,967,296 = 4GB 的地址。

9:

3G 存在四种标准: CDMA2000, WCDMA, TD-SCDMA, WiMAX。

多选:

4:

RGB 三原色: 红、绿、蓝

5:

N 个结点的二叉树叶子结点数量范围是: $1-n/2$ (n 为偶数), $1-(n/2+1)$ (n 为奇数)

完全二叉树的子结点数量范围是: $n/2$ (n 为偶数), $(n/2+1)$ (n 为奇数)

9:

电子邮件协议有三种: SMTP、POP3、IMAP4

电子邮件相关的其他协议有: HTTP/HTML、MIME、LDAP

10:

一个复杂问题如果能在**多项式时间内解决**, 它就被称为 P 问题

一个问题的解可以在**多项式的时间内被验证**, 那么这个问题就是 NP 问题

NOIP 提高组 2012 年真题答案

第十八届全国青少年信息学奥林匹克联赛初赛
提高组参考答案

一、单项选择题（共 10 题，每题 1.5 分，共计 15 分）

1	2	3	4	5	6	7	8	9	10
A	B	B	A	D	A	A	D	A	B

二、不定项选择题（共 10 题，每题 1.5 分，共计 15 分，多选或少选均不得分）

1	2	3	4	5
A	AD	AD	BD	ABC
6	7	8	9	10
CD	AB	A	CD	BD

三、问题求解（共 2 题，每题 5 分，共计 10 分）

- 1. 256
- 2. 5536

四、阅读程序写结果（共 4 题，每题 8 分，其中第 3 题的 2 个小题各 4 分，共计 32 分）

- 1. 41
- 2. 16
- 3. (1) 7 (4 分)
(2) 2004 (4 分)
- 4. 55

五、完善程序（第 1 题第 2 空 3 分，其余每空 2.5 分，共计 28 分）以下各程序填空可能还有一些等价的写法，各省赛区可请本省专家审定和上机验证，可以不上报 CCF NOI 科学委员会检查。

		Pascal 语言	C++语言	C 语言
1	①	false		0
	②	used[data[i]] := false	used[data[i]] = false	used[data[i]] = 0
	③	j		
	④	n		
	⑤	break		
2	①	next := (k mod c) + 1	return (k % c) + 1	
	②	s[n] := q[tail]	s[n] = q[tail]	
	③	q[head]		
	④	q[head]		
	⑤	q[tail]		
	⑥	next(head)		

其中，Pascal 语言和 C++ 语言中的 false 可以用 0 代替；第 2 题第 1 空中的圆括号可以省略。

复赛练习

1. P1004

这题其实是经典的思维 DP, 用数组 $dp[i][j][k][l]$ 表示第一次走到 (i,j) 时和第二次走到 (k,l) 时的最大值能取到多少。

用 $map[i][j]$ 存第 (i,j) 的值是多少。

因为只能往右和下走, 所以转移方程很明显是:

$dp[i][j][k][l] = \max(dp[i-1][j][k-1][l], dp[i-1][j][k][l-1], dp[i][j-1][k-1][l], dp[i][j-1][k][l-1]) + map[i][j] + map[k][l];$

再看题目要求, 当取走一个格子上的数时, 数就没了, DP 也要做到这点。其实只用加一个 $if(i=k \& \& j=l) dp[i][j][k][l] -= map[k][l]$ 就行了, 因为前面取过之后再到的话就不拿, 也就是减去一次。

```
/*
如果只走一次, 那么状态转移方程为: f[i][j] = max(f[i-1][j], f[i][j-1]) + a[i][j]
如果走两次, 同理
*/
#include<bits/stdc++.h>
using namespace std;
int n, ans, f[10][10][10][10], a[10][10];
int main() {
    cin >> n;
    int x, y, z;
    while (cin >> x >> y >> z && x && y && z) {
        a[x][y] = z;
    }
    for (int i = 1; i <= n; i++) { // a[i][j] 表示第一次走到点
        for (int j = 1; j <= n; j++) {
            for (int k = 1; k <= n; k++) { // a[k][l] 表示第二次走到点
                for (int l = 1; l <= n; l++) {
                    f[i][j][k][l] = max(f[i-1][j][k-1][l], max(f[i-1][j][k][l-1], max(f[i][j-1][k-1][l], f[i][j-1][k][l-1]))) + a[i][j] + a[k][l];
                    if (i == k && j == l) {
                        f[i][j][k][l] -= a[i][j]; // 减去重复走的点
                    }
                }
            }
        }
    }
    cout << f[n][n][n][n];
    return 0;
}
```

2. P1019

读题留意:

- 两个单词合并时, 合并部分取的是最小重叠部分
- 相邻的两部分不能存在包含关系就是说如果存在包含关系, 就不能标记为使用过。
- 每个单词最多出现两次。

搜索思路:

先从第一个到最后一个单词看一看哪个单词是指定字母为开头的, 作为深搜的第一个单词, 同时标记使用过一次

```
#include<bits/stdc++.h>
using namespace std;
string str[25], res; // res 表示龙
int n, num[25], sum; // num[] 为每个字符串使用过的次数, sum 为龙的最大长度
char _sstart; // 首字母, 开头

int position(int k) {
    int len1 = res.size();
    int len2 = str[k].size();
    for (int i = 1; i < min(len1, len2); i++) {
        bool flag = true;
        for (int j = 0; j < i; j++) {
            if (res[len1 - i + j] != str[k][j]) { // 龙的末尾跟单词的开头挨个进行比较
                flag = false;
                break;
            }
        }
        if (flag)
            return i;
    }
    return 0;
}
```

```

void dfs() {
    int len = res.size();
    sum = max(sum, len); //如果新找到的第一个单词长度就比之前的龙长，则更新sum
    for (int i = 0; i < n; i++) {
        if (num[i] >= 2) //用过两次的单词，直接跳过
            continue;
        int x = position(i); //找重叠部分的末位，也就是重叠的长度
        if (x > 0) {
            //字符串截取函数的用法：
            //str.substr(x) : 截取从下标x开始往后的所有部分
            //str.substr(x, len) : 截取从下标x开始，往后长度为len的部分
            res += str[i].substr(x);
            num[i]++;
            dfs(); //找下一个
            res.erase(len); //回溯时要将添加的部分删去
            num[i]--;
        }
    }
    return;
}

int main() {
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> str[i]; //字符串数组，保存字符串
    cin >> _sstart;
    for (int i = 0; i < n; i++) {
        if (str[i][0] == _sstart) //找开头的单词
        {
            res = str[i]; //更新龙 res
            num[i]++; //用过的单词计数
            dfs();
            memset(num, 0, sizeof(num)); //每找完一条龙，清零
        }
    }
    cout << sum << endl;
    return 0;
}

```

3. P1025

是排列组合里的题，可以把一个数值为 n 的数当做 n 个小球，划分的份数 k 当做 k 个盒子，那么本题可以转化为“将 n 个小球放到 k 个盒子中，小球之间与盒子之间没有区别，并且最后的结果不允许空盒”

将 n 个小球放到 k 个盒子中的情况总数 =

a.至少有一个盒子只有一个小球的情况数

+b.没有一个盒子只有一个小球的情况数

这样进行划分是因为这种分类可以使 a 和 b 都有能写出来的表达式：

a.因为盒子不加区分，那么 1 的情况数与“将 $n-1$ 个小球放到 $k-1$ 个盒子中”的情况数一样

b.没有一个盒子只有一个小球，那么把每个盒子中拿出来一个小球，对应的是“把 $(n-k)$ 个小球放到 k 个盒子中的情况数”

然后将上面的思路化为动态转移方程：

设 $f[n,k]$ 代表将 n 个小球放到 k 个盒子中且没有空盒的情况，那么 $f[n,k] = f[n-1,k-1] + f[n-k,k]$

而当 $k=1$ 时只有 1 种方法(小球全部放进 1 个盒子)

```

#include<iostream>
using namespace std;
long long dp[300][10];
int n, k;

int main() {
    cin >> n >> k;
    dp[0][0] = 1;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= min(k, i); j++) {
            dp[i][j] = dp[i-j][j] + dp[i-1][j-1];
        }
    }
    cout << dp[n][k] << endl;
    return 0;
}

```

4. P1032

求解的个数使用 dfs，求最优解使用 bfs（考虑到寻找顺序一般找到的第一个解就是最优解）

开两个数组记录串的转换关系，然后以 a 串（原串）为起点开始搜索，搜索目标是 b 串。

需要一个 map 记录某个串是不是被搜到过，如果已经搜过了就不再继续搜。

枚举当前队列中队头那个串的每一个位置，对每一个位置枚举所有可能的转换手段，然后去尝试拼接。

对于一个试图要改变的串 str，在它的第 i 位用第 j 种手段改变，首先判断是否可行，然后再逐位拼接。并且如果拼接出的串是合法的，那么就把这个串继续压入队列，再次搜索，中间记录一下步数 step 和 ans。

最后输出 ans 时判断，如果 ans 超过了步数限制直接输出无解，否则输出步数。

```

#include<bits/stdc++.h>
using namespace std;
string a[10], b[10];
int n;
map<string, int> vis; // 类似与 vis 标记数组，如果这个字符串已经搜过了，就不用再搜了
struct e {
    string str; // 当前的字符串
    int step; // 所用步数
};
queue<e> q;

int bfs() {
    q.push(e{a[0], 0}); // 初始化，存原字符串，step 为 0
    while (!q.empty()) {
        e x = q.front();
        q.pop();
        if (x.step == 10) // 不能超过 10 步
            continue;
        for (int i = 1; i <= n; i++) // 遍历替换规则
            // 遍历字符串，看有没有能替换的
            for (int j = x.str.find(a[i], 0); j != -1; j = x.str.find(a[i], j+1)) {
                /* a.find(str, i) 表示在 a 这个字符串中从第 i 个字符寻找 str 这个字符串
                如果没找到，返回 -1 */
                string strTemp = x.str;
                strTemp.replace(j, a[i].length(), b[i]);
                /* 这个 a.replace(i, len, str) 表示在字符串 a 的第 i 个字符后
                的 len 个字符（包括第 i 个字符）替换成 str 这个字符串 */
                if (!vis[strTemp]) // 出现重复的字符串，跳过
                    if (strTemp == b[0]) // 找到
                        return x.step + 1;
                vis[strTemp] = 1; // 标记
                q.push(e{strTemp, x.step + 1});
            }
    }
    return -1; // 找不到
}

```

```

int main() {
    while(cin>>a[n]>>b[n])
        n++;
    n--; //规则为n-1个
    int ans = bfs();
    if(ans!=-1) {
        cout << ans;
    }
    else {
        cout << "NO ANSWER!";
    }
    return 0;
}

```

5. P1033

位移 (X) 公式: $X = V_0 t + 0.5 a t^2$ $X = V_0 t + 0.5 a t^2$, 其中 V_0 是初速度, t 为时间, a 为加速度。

而自由落体 (就是小球下落) 的初速度为 0, 加速度等于重力加速度 g

由此得知, 自由落体运动位移 (Y) 公式: $Y = 0.5 g t^2$ $Y = 0.5 g t^2$

先读入 double 型变量 H, S, V, L, K, n 。

车高为 k , 故小球下落的最短运动距离为 $\sqrt{(h-k)/5}$ $\sqrt{(h-k)/5}$, 那么此时接住小球编号为 $s - v * (\sqrt{(h-k)/5} + 1)$ $s - v * (\sqrt{(h-k)/5} + 1)$;

最长运动距离为 $\sqrt{h/5}$ $\sqrt{h/5}$, 接住的小球编号为 $s - v * \sqrt{h/5}$ $s - v * \sqrt{h/5}$ 。

由于接住的小球编号不会超过 $n-1$ $n-1$ 也不会小于 0, 所以最大的接住数量为 $n - 1 - 0 + 1$ $n - 1 - 0 + 1$ 即 n 个。

```

#include<bits/stdc++.h>
using namespace std;
double h, s, v, l, k;
int n;
double t1, t2;
int s1, s2;

int main() {
    cin>>h>>s>>v>>l>>k>>n;
    t1=sqrt((h-k)/5);
    t2=sqrt(h/5);
    s1=s-t1*v+1;
    s2=s-t2*v;
    s1=min(s1, n);
    s2=max(s2, 0);
    cout << s1-s2;
    return 0;
}

```