作者：24计科（大人院）6班郭庆

TZOJ ID：jasongq

完成题数：30题（至少）

微信号：18702021539

# 1316: Rightmost Digit

## 思路：

快速幂运算，由于只取个位，为了避免运算过程中出现超范围，每次运算都要模10。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. long long fast\_pow(long long n);
8. int main() {
9. int t;
10. scanf("%d", &t);
11. while (t--) {
12. long long n;
13. scanf("%lld", &n);
14. if (n % 10 == 0) printf("0\n");
15. else if (n % 10 == 5)printf("5\n");
16. else {
17. long long ans = fast\_pow(n);
18. printf("%lld\n", ans);
19. }
20. }
21. return 0;
22. }
23. long long fast\_pow(long long n) {
24. long long res = 1;
25. long long temp = n % 10;
26. int temp\_n = n;
27. while (temp\_n) {
28. if (temp\_n % 2 == 1) {
29. res \*= temp;
30. }
31. temp \*= temp;
32. temp\_n /= 2;
33. temp %= 10;
34. res %= 10;
35. }
36. return res;
37. }

# ****1332: 田忌赛马****

## 思路：

感谢俊爷！！！

从最慢的马开始比较，用田忌最慢的马和齐王最快的马比赛。如果田忌最慢的马比齐王最快的马慢，那么这一局齐王胜，指针j（指向田忌的马）向前移动一位，继续下一局比赛。如果田忌最慢的马比齐王最快的马快或者速度相等，那么这一局田忌胜，同时指针i（指向齐王的马）向后移动一位，指针j（指向田忌的马）也向前移动一位，继续下一局比赛。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int get\_coin(vector<int>& minster, vector<int>& lord, int length);
8. int main() {
9. int n;
10. while (1) {
11. scanf("%d", &n);
12. if (n == 0)break;
13. vector<int>tianji(n),king(n);
14. for (int i = 0;i < n;i++)scanf("%d", &tianji[i]);
15. for (int i = 0;i < n;i++)scanf("%d", &king[i]);
16. int ans = get\_coin(tianji, king, n);
17. printf("%d\n", ans);
18. }
19. return 0;
20. }
21. int get\_coin(vector<int>& minster, vector<int>& lord,int length) {
22. int res = 0;
23. sort(minster.begin(), minster.end(), greater<int>());
24. sort(lord.begin(), lord.end(), greater<int>());
25. int min\_left = 0, lord\_left = 0;
26. int min\_right = length - 1, lord\_right = length - 1;
27. while (length--) {
28. if (minster[min\_left] > lord[lord\_left]) {
29. res += 200;
30. min\_left++;
31. lord\_left++;
32. }
33. else if (minster[min\_right] > lord[lord\_right]) {
34. res += 200;
35. min\_right--;
36. lord\_right--;
37. }
38. else if (minster[min\_right] < lord[lord\_left])
39. {
40. res -= 200;
41. min\_right--;
42. lord\_left++;
43. }
44. }
45. return res;
46. }

# 1340: 奇怪的公式

## 思路：

对于一个周长为 n的多边形，要使面积最大，边的长度应该尽量均匀。一个理想的情况是各边的长度尽量接近，因此我们可以选择一种方法来分割周长n 为多条不相等的边。所以这些边的排序最好是一个等差数列。最后由于阶乘的结果会很大，要用大整数进行运算。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. #include<cmath>
5. using namespace std;
6. void print\_arr(vector<long long>arr, long long ind);
7. int main() {
8. int n;
9. while (scanf("%d", &n) != EOF) {
10. if (n == 3) {
11. printf("2 1 2\n2\n");continue;
12. }
13. else if (n == 4) {
14. printf("2 1 3\n3\n");continue;
15. }
16. vector<long long>arr(n);
17. long long ind = 0;
18. for (int i = 2;n >= i;i++) {
19. arr[ind++] = i;
20. n -= i;
21. }
22. long long length = ind;
23. while(n >= 0){
24. arr[length--]++;
25. if (length < 0)length = ind;
26. else n--;
27. }
28. print\_arr(arr,ind);
29. }
30. return 0;
31. }
32. void factorial(vector<long long>& arr,long long length) {
33. vector<int> result(1, 1);
34. for (int i = 0; i < length; ++i) {
35. int carry = 0;
36. for (int j = 0; j < result.size(); ++j) {
37. int temp = result[j] \* arr[i] + carry;
38. result[j] = temp % 10;
39. carry = temp / 10;
40. }
41. while (carry) {
42. result.push\_back(carry % 10);
43. carry /= 10;
44. }
45. }
46. printf("\n");
47. for (int i = result.size() - 1; i >= 0; --i) {
48. printf("%d", result[i]);
49. }
50. printf("\n");
51. }
52. void print\_arr(vector<long long>arr, long long ind) {
53. int length = ind;
54. long long ans = 1;
55. printf("%lld", length);
56. for (int i = 0;i < length;i++) {
57. printf(" %lld", arr[i]);
58. ans \*= arr[i];
59. }
60. factorial(arr, ind);
61. }

# 1352: 仙人球的残影

## 思路：

注意输出格式，特别是n==1

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. void print\_imag(int num);
8. int main() {
9. int n;
10. while (scanf("%d", &n) != EOF) {
11. print\_imag(n);
12. }
13. return 0;
14. }
15. void print\_per\_num(int num) {
16. int count = 0,copy\_num = num;
17. while (copy\_num) {
18. copy\_num /= 10;
19. count++;
20. }
21. for (int i = 0;i < 3 - count;i++) {
22. printf(" ");
23. }
24. printf("%d", num);
25. }
26. void print\_imag(int num) {
27. if (num == 1) {
28. printf("  1\n");
29. return;
30. }
31. int max\_num = (num-1) \* 4, first\_col = (num - 1) \* 4, last\_col = num + 1;
32. for (int i = 1;i <= num;i++) {
33. print\_per\_num(i);
34. }
35. printf("\n");
36. for (int i = 0;i < num - 2;i++) {
37. print\_per\_num(first\_col);
38. for (int j = 0;j < num - 2;j++) {
39. printf("   ");
40. }
41. print\_per\_num(last\_col);
42. first\_col--;
43. last\_col++;
44. printf("\n");
45. }
46. for (int i = first\_col;i >= last\_col;i--) {
47. print\_per\_num(i);
48. }
49. printf("\n");
50. }

# 1356: A == B ?

## 思路：

以字符串输入，去掉前导0和后导0，并对小数点进行特判（感谢阿昊1733807966906）

## 代码：

1. #include <iostream>
2. #include<string>
3. #include <vector>
4. using namespace std;
5. std::string remove\_zero(std::string &s,char pointed);
6. int main()
7. {
8. const char ch = '0';
9. string a,b;
10. while(cin >> a >> b){
11. std::string new\_a = remove\_zero(a,ch);
12. std::string new\_b = remove\_zero(b,ch);
13. *//cout << new\_a << " " << new\_b << endl;*
14. if (new\_a == new\_b)printf("YES\n");
15. else printf("NO\n");
16. }
17. return 0;
18. }
19. std::string remove\_zero(std::string &s,char pointed){
20. std::string new\_str,decimal\_str,ans;
21. int front = 0,end\_ind = s.size()-1;
22. int ptr = 0,have\_point = 0;
23. while(front <= end\_ind){
24. if (s[front] == '.'){
25. have\_point = 1;
26. break;
27. }
28. if (ptr == 0 && s[front] == pointed){
29. front++;
30. }
31. else{
32. ptr = 1;
33. new\_str = new\_str + s[front];
34. front++;
35. }
36. }
37. ptr = 0;
38. if(have\_point){
39. while(front <= end\_ind){
40. if (s[end\_ind] == '.'){
41. have\_point = 1;
42. break;
43. }
44. if (ptr == 0 && s[end\_ind] == pointed){
45. end\_ind--;
46. }
47. else{
48. ptr = 1;
49. decimal\_str = s[end\_ind] + decimal\_str;
50. end\_ind--;
51. }
52. }
53. }
54. if(have\_point and ptr){
55. ans = new\_str +  "." + decimal\_str;
56. }else{
57. ans = new\_str;
58. }
59. return ans;
60. }

# 1357: 不容易系列之一

## 思路：

错排问题，公式为D(n)=(n−1)\*(D(n−1)+D(n−2))

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. const int N = 21;
9. vector<long long>dp(N, 0);
10. dp[0] = 0;
11. dp[1] = 0;
12. dp[2] = 1;
13. for (int i = 3;i < N;i++) {
14. dp[i] = (i - 1) \* (dp[i - 1] + dp[i - 2]);
15. }
16. int n;
17. while (scanf("%d", &n) != EOF) {
18. printf("%lld\n", dp[n]);
19. }
20. return 0;
21. }

# 1367：计算两点间的距离

## 思路：

用欧氏空间计算即可

## 代码：

1. #include <iostream>
2. #include <string>
3. #include <algorithm>
4. #include<cmath>
5. int main() {
6. double x1, y1, x2, y2;
7. while (scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2) != EOF) {
8. double ans;
9. ans = sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));
10. printf("%.2f\n", ans);
11. }
12. return 0;
13. }

# **1368: 计算球体积**

## 思路：

用球体体积计算公式即可

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include<cmath>
5. #define PI 3.1415927
6. int main() {
7. double r;
8. while (scanf("%lf", &r) != EOF) {
9. double ans = 4.0 / 3.0 \* PI \* pow(r, 3);
10. printf("%.3f\n", ans);
11. }
12. return 0;
13. }

# **1370: 数值统计**

## 思路：

输入时同时处理

## 代码：

1. #include <iostream>
2. #include <string>
3. #include<cmath>
4. int main() {
5. int n;
6. float arr[100];
7. while (1) {
8. scanf("%d", &n);
9. if (n == 0) {
10. break;
11. }
12. int nege\_count = 0, pos\_count = 0, zero = 0;
13. for (int i = 0;i < n;i++) {
14. scanf("%f", &arr[i]);
15. if (arr[i] < 0) {
16. nege\_count++;
17. }
18. else if (arr[i] > 0)
19. {
20. pos\_count++;
21. }
22. else
23. {
24. zero ++;
25. }
26. }
27. printf("%d %d %d\n", nege\_count, zero, pos\_count);
28. }
29. return 0;
30. }

# **1372: 水仙花数**

## 思路：

遍历计算（注意输出格式）

## 代码：

1. #include <stdio.h>
2. #include<stdlib.h>
3. #include <math.h>
4. int main(){
5. int m,n;
6. while(scanf("%d %d",&m,&n) != EOF){
7. int amount=0;
8. for(int i=m;i<=n;i++){
9. int bai=i/100;
10. int shi=(i/10)%10;
11. int ge=i%10;
12. if(bai\*bai\*bai+shi\*shi\*shi+ge\*ge\*ge==i){
13. if(amount == 0){
14. printf("%d",i);
15. amount++;
16. }else{
17. printf(" %d",i);
18. amount++;
19. }
20. }
21. }
22. if(amount==0){
23. printf("no\n");
24. }else{
25. printf("\n");
26. }
27. }
28. return 0;
29. }

# **1373: 多项式求和**

## **思路：**



## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. void counting(int num);
8. int main() {
9. int n;
10. scanf("%d", &n);
11. for (int i = 0;i < n;i++) {
12. int num;
13. scanf("%d", &num);
14. counting(num);
15. }
16. return 0;
17. }
18. void counting(int num) {
19. double ans = 0;
20. for (int i = 1;i <= num;i++) {
21. ans += 1.0 / i \* pow(-1, i + 1);
22. }
23. printf("%.2f\n", ans);
24. }

# **1378: 发工资咯：）**

## 思路：

用数组从大到小储存各面额货币，再用整除和取余来计算

## 代码：

1. #include <iostream>
2. #include <string>
3. #include<cmath>
4. int main() {
5. int n;
6. int maping[6] = { 100,50,10,5,2,1 };
7. while (1) {
8. scanf("%d", &n);
9. if (n == 0) {
10. break;
11. }
12. int ans = 0;
13. int wage;
14. for (int i = 0;i < n;i++) {
15. scanf("%d", &wage);
16. int j = 0;
17. int check = wage;
18. while (check > 0) {
19. ans += check / maping[j];
20. check %= maping[j];
21. j++;
22. }
23. }
24. printf("%d\n",ans);
26. }
27. return 0;
28. }

# **1379: C语言合法标识符**

## 思路：

用cctype库函数isalnum和isalpha判断，并用p作为状态机判断是否符合题意

## 代码：

1. #include <iostream>
2. #include <string>
3. #include<cmath>
4. #include<cctype>
5. int main() {
6. int n;
7. while (scanf("%d", &n) != EOF) {
8. std::cin.ignore();
9. for (int i = 0;i < n;i++) {
10. std::string str;
11. std::getline(std::cin, str);
12. int p = 1;
13. int length = str.size();
14. for (int j = 0;j < length;j++) {
15. if (!isalpha(str[0]) && str[0] != '\_') {
16. p = 0;
17. break;
18. }
19. if (!isalnum(str[j]) && str[j] != '\_') {
20. p = 0;
21. break;
22. }
23. }
24. if (p == 1) {
25. printf("yes\n");
26. }
27. else
28. {
29. printf("no\n");
30. }
31. }
32. }
33. return 0;
34. }

# ****1381: 查找最大元素****

## 思路：

Sort函数从大到小排序，对第一个元素进行标记，

逐位判断输出即可

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. std::string ss;
9. while (std::cin >> ss) {
10. std::string ss\_copy = ss;
11. int length = ss.size();
12. std::sort(ss.begin(), ss.end(), greater<char>());
13. char max\_char = ss[0];
14. for (int i = 0;i < length;i++) {
15. if (ss\_copy[i] == max\_char) {
16. printf("%c(max)", ss\_copy[i]);
17. }
18. else
19. {
20. printf("%c", ss\_copy[i]);
21. }
22. }
23. printf("\n");
24. }
25. return 0;
26. }

# **1386: 十进制转R进制**

## 思路：

用一个数组保存0到F，再用余数作为索引取值，最后反转结果

## 代码：

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. const std::string MAPPING = "0123456789ABCDEF";
5. std::string int\_to\_r(long num, int r) {
6. std::string ans;
7. bool point = num < 0;
8. num = abs(num);
9. while (num > 0) {
10. int yu = num % r;
11. ans += MAPPING[yu];
12. num /= r;
13. }
14. if (ans.empty()) ans = "0";
15. if (point) ans += '-';
16. return ans;
17. }
18. int main() {
19. long n;
20. int r;
21. while (scanf("%lld%d", &n, &r) != EOF) {
22. std::string str = int\_to\_r(n, r);
23. int length = str.size();
24. for (int i = length - 1;i > -1;i--) {
25. printf("%c", str[i]);
26. }
27. printf("\n");
28. }
29. return 0;
30. }

# 1392: 三角形

## **思路：**

依据三角形定义判断（注意题目“实数”1733713834687）

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int n;
9. scanf("%d", &n);
10. for (int i = 0;i < n;i++) {
11. double a, b, c;
12. scanf("%lf%lf%lf", &a, &b, &c);
13. if (b < (a+c) and c < (b+a) and a < (b + c)) {
14. printf("YES\n");
15. }
16. else
17. {
18. printf("NO\n");
19. }
20. }
21. return 0;
22. }

# **1393: 求平均成绩**

## **思路：**

输入遍历处理即可（让人绝望的输出格式判定）

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int n,m;
9. while (scanf("%d%d", &n, &m) != EOF) {
10. std::vector<vector<int>>grade(n, vector<int>(m));
11. std::vector<double> per\_student(n,0.0), per\_lesson(m,0.0);
12. int is = 0, il = 0;
13. for (int i = 0;i < n;i++) {
14. for (int j = 0;j < m;j++) {
15. scanf("%d", &grade[i][j]);
16. per\_student[i] += grade[i][j];
17. per\_lesson[j] += grade[i][j];
18. }
19. per\_student[i] /= m;
20. }
21. int count = 0;
22. for (int i = 0;i < m;i++) {
23. per\_lesson[i] /= (float)n;
24. }
25. for (int i = 0;i < n;i++) {
26. int p = 1;
27. for (int j = 0;j < m;j++) {
28. if (grade[i][j] < per\_lesson[j]) {
29. p = 0;
30. break;
31. }
32. }
33. if (p) {
34. count++;
35. }
36. }
37. for (int i = 0;i < n;i++) {
38. printf("%.2f", per\_student[i]);
39. if (i < n - 1) {
40. printf(" ");
41. }
42. }
43. printf("\n");
44. for (int i = 0;i < m;i++) {
45. printf("%.2f", per\_lesson[i]);
46. if (i < m - 1) {
47. printf(" ");
48. }
49. }
50. printf("\n");
51. printf("%d\n\n",count);
52. }
53. return 0;
54. }

# **1394: 不容易系列之二**

## **思路：**

由题意得，an+1 = an/2 + 1,反推得an =(an+1 - 1)\*2,循环计算即可

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int n;
9. scanf("%d", &n);
10. int count;
11. for (int i = 0;i < n;i++) {
12. scanf("%d", &count);
13. int ans = 3;
14. for (int j = 0;j < count;j++) {
15. ans = (ans - 1) \* 2;
16. }
17. printf("%d\n", ans);
18. }
19. return 0;
20. }

# ****1395: 密码****

## 思路：

用内置函数进行判断即可

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int m;
9. scanf("%d", &m);
10. std::string ss;
11. while (m--) {
12. std::cin >> ss;
13. vector<int>cnt(4, 0);
14. size\_t length = ss.size();
15. if (8 > length or length > 16) {
16. printf("NO\n");
17. continue;
18. }
19. for (int i = 0;i < length;i++) {
20. if (isalpha(ss[i])) {
21. if (isupper(ss[i]))cnt[0]++;
22. else cnt[1]++;
23. }
24. else if (isdigit(ss[i]))cnt[2]++;
25. else cnt[3]++;
26. }
27. int count = 0;
28. for (int num : cnt) {
29. if (num > 0) count++;
30. }
31. if (count >= 3)printf("YES\n");
32. else printf("NO\n");
33. }
34. return 0;
35. }

# 1396: 一只小蜜蜂...

## 思路：

动态规划，同题1398，从n-1到n有1条路，n-2到n有2条路

## 代码：

1. #include <iostream>
2. #include<string>
3. #include <vector>
4. using namespace std;
5. int main()
6. {
7. const int N = 51;
8. vector<long long>dp(N,0);
9. dp[1] = 1;
10. dp[2] = 2;
11. for(int i = 3;i < N;i++){
12. dp[i] = dp[i-1] + dp[i-2];
13. }
14. int n;
15. scanf("%d",&n);
16. while(n--){
17. int a,b;
18. scanf("%d%d",&a,&b);
19. int num = b-a;
20. printf("%lld\n",dp[num]);
21. }
22. return 0;
23. }

# ****1397: 不容易系列之(3)―― LELE的RPG难题****

## 思路：

动态规划，设dp[n][4],dp[n][0]~dp[n][2]为’R’’P’’G’,递推公式为 dp[i][0]=dp[i−1][1]+dp[i−1][2],dp[i][1]=dp[i−1][0]+dp[i−1][2],dp[i][2]=dp[i−1][0]+dp[i−1][1],dp[i][3]=dp[i][0]+dp[i][1]+dp[i][2]−dp[i−1][3]。首先计算所有长度为 iii 的合法方案数（dp[i][0] + dp[i][1] + dp[i][2]），这些方案满足相邻格子不同色。然后减去首尾颜色相同的方案数 dp[i−1][3]，因为 dp[i−1][3] 表示首尾颜色相同的情况下，剩余中间部分满足条件的方案数。

（这题其实是使用python递归枚举算出各项结果再判断规律的1733923684056）

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. vector <vector< long long >> dp(51, vector<long long>(4, 0));
9. dp[1][0] = dp[1][1] = dp[1][2] = 1;
10. dp[0][0] = dp[0][1] = dp[0][2] = 0;
11. dp[1][3] = 3, dp[2][3] = 6;
12. for (int i = 2;i < 51;i++) {
13. dp[i][0] = dp[i - 1][1] + dp[i - 1][2];
14. dp[i][1] = dp[i - 1][0] + dp[i - 1][2];
15. dp[i][2] = dp[i - 1][1] + dp[i - 1][0];
16. if (i > 2)dp[i][3] = dp[i][0] + dp[i][1] + dp[i][2] - dp[i - 1][3];
17. }
18. int n;
19. while (scanf("%d", &n) != EOF) {
20. if (n == 1) {
21. printf("%d\n", 3);
22. continue;
23. }
24. long long ans = dp[n][3];
25. printf("%lld\n", ans);
26. }
27. return 0;
28. }

# 1398: 骨牌铺方格

## 思路：

动态规划，从n-1到n时，只有竖排一种，从n-2到n时，用双竖排和双横排两种，故dp[n] = dp[n-1] + dp[n-2]

## 代码：

1. #include <iostream>
2. #include<string>
3. #include <vector>
4. using namespace std;
5. int main()
6. {
7. const int N = 51;
8. vector<long long>dp(N,0);
9. dp[1] = 1;
10. dp[2] = 2;
11. for(int i = 3;i < N;i++){
12. dp[i] = dp[i-1] + dp[i-2];
13. }
14. int num;
15. while(scanf("%d",&num) != EOF){
16. printf("%lld\n",dp[num]);
17. }
18. return 0;
19. }

# **1399: 阿牛的EOF牛肉串**

## 思路：

动态规划，定义状态dp[i][0] 表示长度为 i的合法字符串中，以 "E" 结尾的字符串数量，dp[i][1] 表示以 "O" 结尾，dp[i][2] 表示以 "F" 结尾。由于不能有“OO”，所以dp[i][1]=dp[i-1][0]+dp[i-1][2]，其他则是sum（dp[i-1]）。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. const int N = 41;
9. vector<vector<long long>>dp(N, vector<long long>(3, 0));
10. dp[1][0] = 1;
11. dp[1][1] = 1;
12. dp[1][2] = 1;
13. for (int i = 2;i < N;++i) {
14. dp[i][0] = dp[i - 1][0] + dp[i - 1][1] + dp[i - 1][2];
15. dp[i][1] = dp[i - 1][0] + dp[i - 1][2];
16. dp[i][2] = dp[i - 1][0] + dp[i - 1][1] + dp[i - 1][2];
17. }
18. int n;
19. while (scanf("%d", &n) != EOF) {
20. long long ans = dp[n][0] + dp[n][1] + dp[n][2];
21. printf("%lld\n", ans);
22. }
23. return 0;
24. }

# **1400: 神、上帝以及老天爷**

## 思路：

错排问题，公式计算：D(n)=(n−1)×(D(n−1)+D(n−2))，分母为n!，概率为：P(n)=D(n) / n!​

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. long long factorial\_num(int num);
8. long long derangement(int num);
9. int main() {
10. int c;
11. scanf("%d", &c);
12. for (int i = 0;i < c;i++) {
13. int n;
14. scanf("%d", &n);
15. long long total = factorial\_num(n);
16. long long res = derangement(n);
17. double ans = (double)res / total \* 100.0;
18. printf("%.2f%%\n", ans);
19. }
20. return 0;
21. }
22. long long factorial\_num(int num) {
23. long long res = 1;
24. for (int i = 2;i <= num;i++) {
25. res \*= i;
26. }
27. return res;
28. }
29. long long derangement(int num) {
30. if (num == 1)return 0;
31. if (num == 2)return 1;
32. long long res1 = 0, res2 = 1, ans = 0;
33. for (int i = 3;i <= num;++i) {
34. ans = (i-1) \* (res1 + res2);
35. res2 = res1;
36. res1 = ans;
37. }
38. return ans;
39. }

# **1403: Picture**

## 思路：

水题（凑数），无

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. void print\_imagination(int row, int col);
8. int main() {
9. int n, m;
10. while (scanf("%d%d", &n, &m) != EOF) {
11. print\_imagination(n, m);
12. printf("\n");
13. }
14. return 0;
15. }
16. void print\_boun(int col) {
17. printf("+");
18. for (int i = 0;i < col;i++) {
19. printf("-");
20. }
21. printf("+\n");
22. }
23. void print\_mid(int row, int col) {
24. printf("|");
25. for (int i = 0;i < col;i++) {
26. printf(" ");
27. }
28. printf("|\n");
29. }
30. void print\_imagination(int col, int row) {
31. print\_boun(col);
32. for (int i = 0;i < row;i++) {
33. print\_mid(row, col);
34. }
35. print\_boun(col);
36. }

# ****1404: Switch Game****

## 思路：

由于只用输出第n个灯的状态，故循环时只需要对该灯进行判断即可，以true和false作为灯的亮与灭

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int n;
9. while (scanf("%d", &n) != EOF) {
10. bool n\_state = false;
11. for (int i = 1;i < n + 1;i++) {
12. if (n % i == 0) {
13. n\_state = !n\_state;
14. }
15. }
16. if (n\_state)printf("1\n");
17. else printf("0\n");
18. }
19. return 0;
20. }

# 1406: A + B Again

## 思路：

转转乐，对正负号进行预处理再进行计算

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include <sstream>
6. #include <iomanip>
7. #include<cmath>
8. using namespace std;
9. std::string digit\_prosess(std::string& ss, int\* p);
10. long long hex\_to\_int(std::string& hex\_num);
11. int main() {
12. std::string a, b;
13. while (cin >> a >> b) {
14. int pa = 1, pb = 1, pans = 1;
15. std::string new\_a = digit\_prosess(a, &pa);
16. std::string new\_b = digit\_prosess(b, &pb);
17. long long num\_a = hex\_to\_int(new\_a);
18. long long num\_b = hex\_to\_int(new\_b);
19. if (pa == 0)  num\_a = -num\_a;
20. if (pb == 0) num\_b = -num\_b;
21. long long ans = num\_a + num\_b;
22. if (ans < 0) {
23. ans = abs(ans);
24. pans = 0;
25. }
26. if (pans)printf("%llX\n", ans);
27. else printf("-%llX\n", ans);
28. }
29. return 0;
30. }
31. std::string digit\_prosess(std::string& ss,int\*p) {
32. std::string new\_ss;
33. size\_t length = ss.size();
34. if (ss[0] == '-') {
35. \*p = 0;
36. for (int i = 1;i < length;i++) {
37. new\_ss += ss[i];
38. }
39. }
40. else if (ss[0] == '+') {
41. for (int i = 1;i < length;i++) {
42. new\_ss += ss[i];
43. }
44. }
45. else
46. {
47. new\_ss = ss;
48. }
49. return new\_ss;
50. }
51. long long hex\_to\_int(std::string& hex\_num) {
52. long long res;
53. std::stringstream ss;
54. ss << std::hex << hex\_num;
55. ss >> res;
56. return res;
57. }

# ****1407: The sum problem****

## 思路：

要找到总和为m的子序列，相当于要找到一段的严格单调递增序列，由等差数列和公式(a1+an)\*n/2 = M得，要使a1大于0，a1 > ((1+8\*M)^0.5-1)/2，且a1要为整数。以此为数学关系，可以求出结果。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. void counting(long long n, long long m);
8. int main() {
9. long long n, m;
10. while (1) {
11. scanf("%lld%lld", &n, &m);
12. if (n == 0 and m == 0) {
13. break;
14. }
15. counting(n, m);
16. printf("\n");
17. }
18. return 0;
19. }
20. void counting(long long n, long long m) {
21. long long right = (sqrt(1 + 8 \* m)-1 )/ 2.0 + 1;
22. for (long long num = right;num > -1 ;num--) {
23. double count = (2.0 \* m / (num + 1) - num) / 2.0;
24. if ((long long)count == count / 1 and count > 0) {
25. long long s = count;
26. long long e = count + num;
27. if (e <= n) {
28. printf("[%lld,%lld]\n", s, e);
29. }
30. }
31. }
32. }

# 1416: 夹角有多大

## 思路：

由计算得，一小时30度，一分钟6度，每过一秒分针加0.1度，每过一分钟时针加0.5度，以此为数量关系进行计算

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int get\_angle(int h, int m, int s);
8. int main() {
9. int t;
10. scanf("%d", &t);
11. for (int i = 0;i < t;i++) {
12. int h, m, s;
13. scanf("%d%d%d", &h, &m, &s);
14. int ans = get\_angle(h, m, s);
15. printf("%d\n", ans);
16. }
17. return 0;
18. }
19. int get\_angle(int h, int m, int s) {
20. if (h > 11)h -= 12;
21. float angle\_min = m \* 6 + s \* 0.1;
22. float angle\_hour = (m + s / 60.0) \* 0.5 + h \* 30;
23. float angle\_hour\_min = fabs(angle\_hour - angle\_min);
24. if (angle\_hour\_min > 180)angle\_hour\_min = 360 - angle\_hour\_min;
25. return angle\_hour\_min;
26. }

# ****1418: 复习时间****

## 思路：

没用的m1733877104279

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int t;
9. scanf("%d", &t);
10. while (t--) {
11. int n, m;
12. scanf("%d%d", &n, &m);
13. long long min\_num = 110, a;
14. while (n--) {
15. scanf("%lld", &a);
16. if (a < min\_num) min\_num = a;
17. }
18. long long ans = pow(100 - min\_num, 2);
19. printf("%lld\n", ans);
20. }
21. return 0;
22. }

# 1424: 核反应堆

## 思路：

动态规划，1个高能质点分成3个高能质点和1个低能质点，

1个低能质点分成2个高能质点和1个低能质点。所以初始化数组dp[i][2]，dp[i][0]为高能质点，dp[i][1]为低能质点，因此有dp[i][0] = dp[i-1][0] \* 3 + dp[i-1][1] \* 2， dp[i][1] = dp[i-1][0] + dp[i-1][1]

## 代码:

1. #include <iostream>
2. #include<string>
3. #include <vector>
4. using namespace std;
5. int main()
6. {
7. const int N = 34;
8. vector<vector<long long>>dp(N,vector<long long>(2,0));
9. dp[0][0] = 1;
10. dp[0][1] = 0;
11. for(int i = 1;i < N;i++){
12. dp[i][0] = dp[i-1][0] \* 3 + dp[i-1][1] \* 2;
13. dp[i][1] = dp[i-1][0] + dp[i-1][1];
14. }
15. int n;
16. while(1){
17. scanf("%d",&n);
18. if (n == -1){
19. return 0;
20. }
21. printf("%lld, %lld\n",dp[n][0],dp[n][1]);
22. }
24. return 0;
25. }

# 1427: 不要62

## 思路：

暴力即可

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. int main() {
8. int n, m;
9. while (1) {
10. scanf("%d%d", &n, &m);
11. if (!n && !m) {
12. break;
13. }
14. int ans = 0;
15. for (int i = n;i <= m;i++) {
16. string ss = to\_string(i);
17. if (ss.find("4") == string::npos && ss.find("62") == string::npos  ) {
18. ans++;
19. }
20. }
21. printf("%d\n", ans);
22. }
23. return 0;
24. }

# 1429: 小明A+B

## 思路：

……（凑数题）

## 代码：

1. #include <iostream>
2. #include <string>
3. #include <algorithm>
4. #include<cmath>
5. using namespace std;
6. int main() {
7. int t;
8. scanf("%d", &t);
9. for (int i = 0;i < t;i++) {
10. int a, b;
11. scanf("%d%d", &a,&b);
12. a %= 100;
13. b %= 100;
14. int ans = (a + b) % 100;
15. printf("%d\n", ans);
16. }
17. return 0;
18. }

# ****1707: Catch That Cow****

## 思路：

简单的bfs,但要注意k小于和等于n的情况

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. #include <queue>
5. #include<cmath>
6. #define MAX\_INT 100100
7. using namespace std;
8. const vector<int>MOVING = { 1,-1,2 };
9. vector<bool>visited(MAX\_INT, false);
10. int bfs();
11. int farmer, cow;
12. int main() {
13. scanf("%d %d", &farmer, &cow);
14. int ans;
15. if (cow < farmer) ans = farmer - cow;
16. else if (cow == farmer) ans = 0;
17. else ans = bfs();
18. printf("%d\n", ans);
19. return 0;
20. }
21. int bfs() {
22. queue<pair<int, int>>q;
23. q.push({ 0,farmer });
24. visited[farmer] = true;
25. while (!q.empty()) {
26. pair<int, int>top = q.front();
27. int step = top.first, coor = top.second;
28. q.pop();
29. for (const auto& move : MOVING) {
30. int new\_coor;
31. if (move == 2)new\_coor = coor \* 2;
32. else new\_coor = coor + move;
33. if (new\_coor >= 0 and new\_coor < MAX\_INT and !visited[new\_coor]) {
34. if (new\_coor == cow) {
35. return step + 1;
36. }
37. q.push({ step + 1,new\_coor });
38. visited[new\_coor] = true;
39. }
40. }
41. }
42. }

# ****2984: Image Transformation****

## ****思路：****



## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. using namespace std;
6. void input\_arr(vector <vector< int >>& arr, int n, int m);
7. void print\_arr(vector <vector< int >>& arr, int n, int m,int i);
8. int main() {
9. int n, m,i = 1;
10. while (1) {
11. scanf("%d%d", &n, &m);
12. if (n == 0 and m == 0)break;
13. vector <vector< int >>image\_color(n, vector<int>(m));
14. input\_arr(image\_color, n, m);
15. print\_arr(image\_color, n, m,i);
16. i++;
17. }
18. return 0;
19. }
20. void input\_arr(vector <vector< int >>& arr, int n, int m) {
21. for (int i = 0;i < 3;i++) {
22. for (int j = 0;j < n;j++) {
23. for (int k = 0;k < m;k++) {
24. int num;
25. scanf("%d", &num);
26. arr[j][k] += num;
27. if(i == 2)arr[j][k] /= 3;
28. }
29. }
30. }
31. }
32. void print\_arr(vector <vector< int >>& arr, int n, int m,int i) {
33. printf("Case %d:\n", i);
34. for (int i = 0;i < n;i++) {
35. for (int j = 0;j < m;j++) {
36. if (j > 0)printf(",");
37. printf("%d",arr[i][j]);
38. }
39. printf("\n");
40. }
41. }

# ****6738: 切蛋糕****

## 思路：

由样例得思路，最多3刀。当有两者的需求相同时，若两者需求为0，则1刀，否则2刀，当其中两者的需求之和等于第三者，则2刀，若都不是，为3刀。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include <map>
6. #include <set>
7. #include<climits>
8. #include <queue>
9. #include<cmath>
10. using namespace std;
11. int cnt\_cut(vector<int>& arr);
12. int main() {
13. int t;
14. scanf("%d", &t);
15. while (t--) {
16. vector<int> arr(3);
17. for (int i = 0;i < 3;i++)scanf("%d", &arr[i]);
18. int ans = cnt\_cut(arr);
19. printf("%d\n", ans);
20. }
21. return 0;
22. }
23. int cnt\_cut(vector<int>& arr) {
24. int cnt = 0;
25. if (arr[0] == 0) { cnt++; }
26. if (arr[1] == 0) { cnt++; }
27. if (arr[2] == 0) { cnt++; }
28. if (cnt == 2) {
29. return 0;
30. }
31. else if (cnt == 1)
32. {
33. if (arr[0] == arr[1] || arr[0] == arr[2] || arr[2] == arr[1])return 1;
34. else return 2;
35. }
36. if (arr[0] == arr[1] || arr[0] == arr[2] || arr[1] == arr[2])return 2;
37. sort(arr.begin(), arr.end());
38. if (arr[0] + arr[1] == arr[2])return 2;
39. return 3;
40. }

# ****2939: 解救小Q****

## 思路:

bfs，队列模拟，用map储存传送门位置，出队入队遍历即可，注意：“同一个传送门可以多次使用”（感谢俊爷！！！）

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<climits>
6. #include<unordered\_map>
7. #include <queue>
8. #include<cmath>
9. using namespace std;
10. const vector<vector<int>>MOVING = { {1,0},{-1,0},{0,-1},{0,1} };
11. vector<vector<int>>cnt(51, vector<int>(51, 2500));
12. void input\_point(vector <vector< char >>& point, unordered\_map<char, vector<pair<int, int>>>& portals, int start\_coor[], int end\_coor[], int n, int m);
13. int min\_dis(vector <vector< char >>& point, unordered\_map<char, vector<pair<int, int>>>& portals, int start\_coor[], int end\_coor[], int n, int m);
14. int main() {
15. int t;
16. scanf("%d", &t);
17. while (t--) {
18. int n, m;
19. scanf("%d%d", &n, &m);
20. vector<vector<char>>point(n + 1, vector<char>(m + 1));
21. unordered\_map<char, vector<pair<int, int>>> portals;
22. int start\_coor[2], end\_coor[2];
23. input\_point(point, portals, start\_coor, end\_coor, n, m);
24. int ans = min\_dis(point, portals, start\_coor, end\_coor, n, m);
25. printf("%d\n", ans);
26. }
27. return 0;
28. }
29. void input\_point(vector <vector< char >>& point, unordered\_map<char, vector<pair<int, int>>>& portals, int start\_coor[], int end\_coor[], int n, int m) {
30. for (int i = 1;i <= n;i++) {
31. for (int j = 1;j <= m;j++) {
32. char ss;
33. cin >> ss;
34. point[i][j] = ss;
35. if (isalpha(ss)) {
36. if (ss == 'L') {
37. start\_coor[0] = i;start\_coor[1] = j;
38. }
39. else if (ss == 'Q') {
40. end\_coor[0] = i;end\_coor[1] = j;
41. }
42. else {
43. portals[ss].emplace\_back(i, j);
44. }
45. }
46. }
47. }
48. }
49. int min\_dis(vector <vector< char >>& point, unordered\_map<char, vector<pair<int, int>>>& portals, int start\_coor[], int end\_coor[], int n, int m) {
50. queue<pair<int, pair<int, int>>>q;
51. vector<vector<bool>>states(n + 1, vector<bool>(m + 1, false));
52. q.push({ 0,{start\_coor[0],start\_coor[1]} });
53. states[start\_coor[0]][start\_coor[1]] = true;
54. while (!q.empty()) {
55. pair<int, pair<int, int>>top = q.front();
56. int step = top.first;
57. int x = top.second.first, y = top.second.second;
58. q.pop();
59. step++;
60. for (const auto& move : MOVING) {
61. int newx = x + move[0], newy = y + move[1];
62. if (newx > 0 && newy > 0 && newx <= n && newy <= m && point[newx][newy] != '#' && point[newx][newy] != 'L' && !states[newx][newy]) {
63. if (point[newx][newy] == 'Q') {
64. return step;
65. }
66. if (islower(point[newx][newy])) {
67. cnt[newx][newy]--;
68. if (cnt[newx][newy] <= 0) {
69. states[newx][newy] = true;
70. }
71. char ind = point[newx][newy];
72. if (portals[ind][0].first == newx && portals[ind][0].second == newy) {
73. newx = portals[ind][1].first;
74. newy = portals[ind][1].second;
75. }
76. else {
77. newx = portals[ind][0].first;
78. newy = portals[ind][0].second;
79. }
80. cnt[newx][newy]--;
81. if (cnt[newx][newy] <= 0) {
82. states[newx][newy] = true;
83. }
84. q.push({ step,{newx,newy} });
85. }
86. else
87. {
88. cnt[newx][newy]--;
89. if (cnt[newx][newy] <= 0) {
90. states[newx][newy] = true;
91. }
92. q.push({ step,{newx,newy} });
93. }
94. }
95. }
96. }
97. return -1;
98. }

# ****6948: 走迷宫****

## 思路：

bfs

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include <queue>
6. #include<cmath>
7. using namespace std;
8. const vector<vector<int>>MOVING = { {1,0},{0,1},{-1,0},{0,-1} };
9. vector<vector<bool>>visited(10, vector<bool>(10, false));
10. vector<vector<char>>maze(10, vector<char>(10));
11. void input\_arr(vector<vector<char>>& matrix);
12. int bfs();
13. int main() {
14. input\_arr(maze);
15. int ans = bfs();
16. if (ans == 1)printf("successful!\n");
17. else printf("fail!\n");
18. return 0;
19. }
20. void input\_arr(vector<vector<char>>& matrix) {
21. for (int i = 0;i < 10;i++) {
22. for (int j = 0;j < 10;j++) {
23. cin >> matrix[i][j];
24. }
25. }
26. }
27. int bfs() {
28. int sx = 1, sy = 1, ex = 8, ey = 8;
29. queue<pair<int, int>>q;
30. q.push({ sx,sy });
31. visited[sx][sy] = true;
32. while (!q.empty()) {
33. pair<int, int>top = q.front();
34. int x = top.first, y = top.second;
35. q.pop();
36. for (const auto& move : MOVING) {
37. int newx = x + move[0], newy = y + move[1];
38. if (newx > 0 and newx < 9 and newy > 0 and newy < 9 and maze[newx][newy] != '0' and !visited[newx][newy]) {
39. if (newx == ex and newy == ey) {
40. return 1;
41. }
42. visited[newx][newy] = true;
43. q.push({ newx,newy });
44. }
45. }
46. }
47. return 0;
48. }

# **洛谷题：**

# P1115 最大子段和

## 思路：

经典的前缀和题目，每次更新前几位数的和并与结果作对比，然后如果几位数的和小于0，则重置为0。

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. using namespace std;
5. int max\_sum(vector<int>& arr, int n);
6. int main() {
7. int n;
8. scanf("%d", &n);
9. vector<int>arr(n);
10. for (int i = 0;i < n;i++)scanf("%d", &arr[i]);
11. int ans = max\_sum(arr, n);
12. printf("%d\n", ans);
13. return 0;
14. }
15. int max\_sum(vector<int>& arr, int n) {
16. int res = -100000;
17. int temp\_sum = 0;
18. for (int i = 0;i < n;i++) {
19. temp\_sum += arr[i];
20. res = max(res, temp\_sum);
21. if (temp\_sum < 0) {
22. temp\_sum = 0;
23. }
24. }
25. return res;
26. }

# P1706 全排列问题：

## 思路：

递归排序

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. #include<cmath>
5. using namespace std;
6. void perment(int i, int n, vector<int>& path);
7. int main() {
8. int n;
9. scanf("%d", &n);
10. vector<int>path;
11. perment(0, n, path);
12. return 0;
13. }
14. void perment(int i,int n, vector<int>& path) {
15. if (i == n) {
16. for (int a = 0;a < n;a++) {
17. printf("    %d", path[a]);
18. }
19. printf("\n");
20. return;
21. }
22. for (int j = 1;j <= n;j++) {
23. if (find(path.begin(), path.end(), j) == path.end()) {
24. path.push\_back(j);
25. i++;
26. perment(i, n, path);
27. i--;
28. path.pop\_back();
29. }
30. }
31. }

# P1093 [NOIP2007 普及组] 奖学金：

## 思路：

依题意，先对比总成绩，再对比语文，最后对比编号

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<cmath>
6. using namespace std;
7. bool cmp(const vector<int>& a, const vector<int>& b) {
8. if (a[4] != b[4]) return a[4] > b[4];
9. else
10. {
11. if (a[1] != b[1])return a[1] > b[1];
12. else
13. {
14. return a[0] < b[0];
15. }
16. }
17. }
18. int main() {
19. int n;
20. scanf("%d", &n);
21. vector <vector< int >> grade(n, vector<int>(5));
22. for (int i = 0;i < n;i++) {
23. scanf("%d%d%d", &grade[i][1], &grade[i][2], &grade[i][3]);
24. grade[i][0] = i + 1;grade[i][4] = grade[i][1] + grade[i][2] + grade[i][3];
25. }
26. sort(grade.begin(), grade.end(), cmp);
27. for (int i = 0;i < 5;i++) {
28. printf("%d %d\n", grade[i][0], grade[i][4]);
29. }
30. return 0;
31. }

# P1443 马的遍历：

## 思路：

bfs，用队列模拟，遍历每一个方向并更新，如果用dfs会超时，并且一个点会遍历多次

## 代码：

1. #include <iostream>
2. #include <vector>
3. #include <string>
4. #include <algorithm>
5. #include<climits>
6. #include <queue>
7. #include<cmath>
8. using namespace std;
9. const vector<vector<int>>moving = { {1,2},{2,1},{-1,-2},{-2,-1},{1,-2},{-1,2},{2,-1},{-2,1} };
10. int main() {
11. int n, m, x, y;
12. scanf("%d%d%d%d", &n, &m, &x, &y);
13. vector<vector<int>>coordinated(n + 1, vector<int>(m + 1, INT\_MAX));
14. queue<pair<int, int>> q;
15. q.push({ x, y });
16. int cx, cy;
17. coordinated[x][y] = 0;
18. while (!q.empty()) {
19. pair<int, int> top = q.front();
20. int cx = top.first;
21. int cy = top.second;
22. q.pop();
23. for (const auto& move : moving) {
24. int nx = cx + move[0];
25. int ny = cy + move[1];
26. if (nx > 0 && nx <= n && ny > 0 && ny <= m && coordinated[nx][ny] == INT\_MAX) {
27. coordinated[nx][ny] = coordinated[cx][cy] + 1;
28. q.push({ nx, ny });
29. }
30. }
31. }
32. if (coordinated[x][y] == INT\_MAX)coordinated[x][y] = 0;
33. for (int i = 1;i <= n;i++) {
34. for (int j = 1;j <= m;j++) {
35. if(coordinated[i][j] == INT\_MAX)
36. printf("-1  ");
37. else {
38. printf("%d  ", coordinated[i][j]);
39. }
40. }
41. printf("\n");
42. }
43. return 0;
44. }

# 总结和感受：

都在思路里