2082第一道生成函数题；

#include <stdio.h>

#include <string.h>

int main()

{

    int n, m, i, j, k, pre[60], cur[60], num[30];

    long long int S;

    while (scanf("%d", &n) != EOF)

    {

        for (m = 1; m <= n; m++)

        {

            for (i = 1; i <= 26; i++)

                scanf("%d", &num[i]);

            memset(pre, 0, sizeof(int) \* 60);

            memset(cur, 0, sizeof(int) \* 60);

            for (i = 0; i <= num[1]; i++)

                pre[i] = 1;

            for (i = 2; i <= 26; i++)

            {

                if (num[i] == 0)

                    continue;

                for (j = 0; j <= 50; j++)

                {

                    for (k = 0; k <= num[i] \* i && k + j <= 50; k = k + i)

                    {

                        cur[k + j] += pre[j];

                    }

                }

                for (j = 0; j <= 50; j++)

                {

                    pre[j] = cur[j];

                    cur[j] = 0;

                }

            }

            S = 0;

            for (i = 1; i <= 50; i++)

                S += pre[i];

            printf("%lld\n", S);

        }

    }

    return 0;

}

2110 带求余的生成函数；

#include <stdio.h>

#include <string.h>

int main()

{

    int n, m, i, j, k, sum, value[100], num[100], pre[10010], cur[10010];

    while (scanf("%d", &n) && n != 0)

    {

        sum = 0;

        for (m = 1; m <= n; m++)

        {

            scanf("%d %d", &value[m], &num[m]);

            sum += value[m] \* num[m];

        }

        if (sum % 3 != 0)

        {

            printf("sorry\n");

            continue;

        }

        sum /= 3;

        memset(pre, 0, sizeof(int) \* 10010);

        memset(cur, 0, sizeof(int) \* 10010);

        for (i = 0; i <= num[1] \* value[1]; i += value[1])

            pre[i] = 1;

        for (i = 2; i <= n; i++)

        {

            for (j = 0; j <= (sum + 10); j++)

            {

                for (k = 0; k <= num[i] \* value[i] && k + j <= (sum + 10); k += value[i])

                    cur[k + j] += pre[j] % 10000;

            }

            for (j = 0; j <= (sum + 10); j++)

            {

                pre[j] = cur[j] % 10000;

                cur[j] = 0;

            }

        }

        if (pre[sum] != 0)

            printf("%d\n", pre[sum]);

        else

            printf("sorry\n");

    }

    return 0;

}

2546 第一个背包；

#include <stdio.h>

#include <string.h>

int main()

{

    int n, m, i, j, bag[1005], value[1005], money, max, temp, sum;

    while (scanf("%d", &n) != EOF && n != 0)

    {

        for (m = 1; m <= n; m++)

        {

            scanf("%d", &value[m]);

        }

        scanf("%d", &money);

        if (money < 5)

        {

            printf("%d\n", money);

            continue;

        }

        for (i = 1; i <= n; i++)

        {

            if (value[n] < value[i])

            {

                temp = value[n];

                value[n] = value[i];

                value[i] = temp;

            }

        }

        memset(bag, 0, sizeof(int) \* 1005);

        sum = 0;

        for (i = 1; i < n; i++)

        {

            for (j = money - 5; j >= value[i]; j--)

            {

                temp = bag[j - value[i]] + value[i];

                bag[j] = (bag[j] > temp) ? bag[j] : temp;

            }

        }

        printf("%d\n", money - bag[money - 5] - value[n]);

    }

    return 0;

}

背包

#include <stdio.h>

#include <string.h>

struct bone

{

    long long int volume, value;

};

int main()

{

    long long int n, m, i, j, k, bag[1010][1010], c, v;

    struct bone Teddy[1010];

    while (scanf("%lld", &n) != EOF)

    {

        for (m = 1; m <= n; m++)

        {

            scanf("%lld %lld", &c, &v);

            for (i = 1; i <= c; i++)

                scanf("%lld", &Teddy[i].value);

            for (i = 1; i <= c; i++)

                scanf("%lld", &Teddy[i].volume);

            for (i = 0; i <= c; i++)

                for (j = 0; j <= v; j++)

                    bag[i][j] = 0;

            for (i = 1; i <= c; i++)

            {

                for (j = v; j >= Teddy[i].volume; j--)

                {

                    bag[i][j] = (bag[i - 1][j] >= bag[i - 1][j - Teddy[i].volume] + Teddy[i].value) ? bag[i - 1][j] : bag[i - 1][j - Teddy[i].volume] + Teddy[i].value;

                }

            }

            printf("%lld\n", bag[c][v]);

        }

    }

    return 0;

}

1203 一维背包, 不用恰好把背包装满。概率题，一定用乘法；

#include <stdio.h>

struct offer

{

    int expense;

    double percent;

};

int main()

{

    int n, m, i, j, k, c;

    double bag[41000];

    struct offer sk[10010];

    while (scanf("%d %d", &n, &m) != EOF && (n != 0 || m != 0))

    {

        for (c = 1; c <= m; c++)

        {

            scanf("%d %lf", &sk[c].expense, &sk[c].percent);

        }

        for (i = 0; i < 41000; i++)

            bag[i] = 1;

        for (i = 1; i <= m; i++)

        {

            for (j = n; j >= sk[i].expense; j--)

            {

                bag[j] = (bag[j] <= bag[j - sk[i].expense] \* (1.0 - sk[i].percent)) ? bag[j] : (bag[j - sk[i].expense] \* (1.0 - sk[i].percent));

            }

        }

        printf("%.1f%%\n", (1.0 - bag[n])\*100.0);

    }

    return 0;

}

2159  完全背包 + 二维背包，二维数组加顺序 即可；

#include <stdio.h>

#include <string.h>

struct game

{

    int experience, patience;

};

int main()

{

    int n, m, i, j, k, bag[200][200], mexperience, mpatience, mkind, mkill, min, doudou, c;

    struct game xhd[200];

    while (scanf("%d %d %d %d", &mexperience, &mpatience, &mkind, &mkill) != EOF)

    {

        for (n = 1; n <= mkind; n++)

        {

            scanf("%d %d", &xhd[n].experience, &xhd[n].patience);

        }

        memset(bag, 0, sizeof(int) \* 200 \* 200);

        for (i = 1; i <= mkind; i++)

        {

            for (j = 1; j <= mkill; j++)

            {

                for (k = xhd[i].patience; k <= mpatience; k++)

                {

                    bag[j][k] = (bag[j][k] >= bag[j - 1][k - xhd[i].patience] + xhd[i].experience) ? bag[j][k] : (bag[j - 1][k - xhd[i].patience] + xhd[i].experience);

                }

            }

        }

        doudou = 0;

        c = 9999999;

        for (i = 0; i <= mpatience; i++)

        {

            for (j = 1; j <= mkill; j++)

            {

                if (bag[j][i] >= mexperience)

                {

                    doudou = 1;

                    c = i;

                    break;

                }

            }

            if (doudou)

                break;

        }

        if (c != 9999999)

            printf("%d\n", mpatience - c);

        else

            printf("-1\n");

    }

    return 0;

}

2955 带概率的基本都是用乘法，这个题为什么不用Roy[i].c直接乘，

因为这样算出来的是既第一次被抓又第二次被抓概率，还要加上第一次被抓第二次不被抓和第一次不被抓第二次被抓。

这样很麻烦，所以直接算1-Roy[i]相成，就是不会被抓的概率，最后算1-  就行了；

初始化也很重要，这个是要恰好，因为我们要算出每一个准确金额的概率，必须装满；

#include <stdio.h>

#include <string.h>

struct bank

{

    int money;

    double c;

};

int main()

{

    int n, m, i, j, k, num, sum, u;

    double bag[11000], mcatch;

    struct bank Roy[110];

    while (scanf("%d", &n) != EOF)

    {

        for (m = 1; m <= n; m++)

        {

            sum = 0;

            scanf("%lf %d", &mcatch, &num);

                for (u = 1; u <= num; u++)

                {

                    scanf("%d %lf", &Roy[u].money, &Roy[u].c);

                        sum += Roy[u].money;

                }

                bag[0] = 1;

                for (i = 0; i < 11000; i++)

                    bag[i] = 0;

            for (i = 1; i <= num; i++)

            {

                for (j = sum; j >= Roy[i].money; j--)

                {

                    bag[j] = (bag[j] > bag[j - Roy[i].money] \* (1.0-Roy[i].c)) ? bag[j] : (bag[j - Roy[i].money] \* (1.0-Roy[i].c));

                }

            }

            for (j = sum; j >= 0; j--)

            {

                if (bag[j] > (1.0-mcatch))

                    break;

            }

            printf("%d\n", j);

        }

    }

    return 0;

}

2141  Ai+Bj+Ck==S  二分查找；

#include <stdio.h>

#include <stdlib.h>

int comp(const void \* p1, const void \* p2)

{

    const int \* a1 = (const int \*)p1;

    const int \* a2 = (const int \*)p2;

    if (\*a1 < \*a2)

        return -1;

    else if (\*a1 == \*a2)

        return 0;

    else

        return 1;

}

int main()

{

    int a[510], b[510], c[510], s, d = 1, temp, i, j, n, m, k, x, y, u, A, B, C, S, D, doudou, set[250010];

    while (scanf("%d %d %d", &A, &B, &C) != EOF)

    {

        for (i = 1; i <= A; i++)

            scanf("%d", &a[i]);

        for (i = 1; i <= B; i++)

            scanf("%d", &b[i]);

        for (i = 1; i <= C; i++)

            scanf("%d", &c[i]);

        u = 0;

        for (i = 1; i <= C; i++)

        {

            for (j = 1; j <= A; j++)

            {

                set[u++] = a[j] + c[i];

            }

        }

        qsort(set, A\*C, sizeof(int), comp);

        scanf("%d", &S);

        printf("Case %d:\n", d);

        d++;

        for (n = 1; n <= S; n++)

        {

            scanf("%d", &s);

            doudou = 0;

            for (i = 1; i <= B; i++)

            {

                x = 0;

                y = A\*C-1;

                while (y >= x)

                {

                    if (s - b[i] < set[(x + y) / 2])

                        y = (x + y) / 2 - 1;

                    else if (s - b[i] > set[(x + y) / 2])

                        x = (x + y) / 2 + 1;

                    else

                    {

                        doudou = 1;

                        break;

                    }

                }

                if (doudou)

                    break;

            }

            if (doudou)

                printf("YES\n");

            else

                printf("NO\n");

        }

    }

    return 0;

}

2844   多重背包   终于过啦。。哈哈哈  人生得到了满足！！

#include <stdio.h>

struct coin

{

    int value;

    int num;

};

int main()

{

    int n, m, i, j, k, bag[1000010], mvalue, c, count;

    struct coin Hibix[2000];

    while (scanf("%d %d", &c, &mvalue) != EOF && (c != 0 || mvalue != 0))

    {

        for (m = 1; m <= c; m++)

            scanf("%d", &Hibix[m].value);

        for (m = 1; m <= c; m++)

            scanf("%d", &Hibix[m].num);

        for (i = 1; i <= mvalue; i++)

            bag[i] = -10000000;

        bag[0] = 0;

        for (i = 1; i <= c; i++)

        {

            if (Hibix[i].value \* Hibix[i].num >= mvalue)

            {

                for (j = Hibix[i].value; j <= mvalue; j++)

                    bag[j] = (bag[j] > bag[j - Hibix[i].value] + Hibix[i].value) ? bag[j] : (bag[j - Hibix[i].value] + Hibix[i].value);

            }

            else

            {

                k = 1;

                while (k <= Hibix[i].num)

                {

                    for (j = mvalue; j >= k\*Hibix[i].value; j--)

                        bag[j] = (bag[j] > bag[j - k\*Hibix[i].value] + k\*Hibix[i].value) ? bag[j] : (bag[j - k\*Hibix[i].value] + k\*Hibix[i].value);

                    Hibix[i].num -= k;

                    k \*= 2;

                }

                if(Hibix[i].num)

                    for (j = mvalue; j >= Hibix[i].num\*Hibix[i].value; j--)

                        bag[j] = (bag[j] > bag[j - Hibix[i].num\*Hibix[i].value] + Hibix[i].num\*Hibix[i].value) ? bag[j] : (bag[j - Hibix[i].num\*Hibix[i].value] + Hibix[i].num\*Hibix[i].value);

            }

        }

        count = 0;

        for (j = 1; j <= mvalue; j++)

        {

            if (bag[j] > 0)

                count++;

        }

        printf("%d\n", count);

    }

    return 0;

}

2159  二维背包，耐性背包恰好装满，不能一个怪都不杀；有负数的要赋值为负无穷

#include <stdio.h>

#include <string.h>

struct monster

{

    int exp;

    int pati;

};

int main()

{

    int n, m, i, j, k, u, bag[105][105], mexp, mpati, mkind, mkill, doudou;

    struct monster xhd[105];

    while (scanf("%d %d %d %d", &mexp, &mpati, &mkind, &mkill) != EOF)

    {

        for (m = 1; m <= mkind; m++)

        {

            scanf("%d %d", &xhd[m].exp, &xhd[m].pati);

        }

        for (i = 0; i < 105; i++)

        {

            for (j = 0; j < 105; j++)

                bag[i][j] = -1000000;

        }

        bag[0][0] = 0;

        for (i = 1; i <= mkind; i++)

        {

            for (j = xhd[i].pati; j <= mpati; j++)

            {

                for (k = 1; k <= mkill; k++)

                {

                    bag[j][k] = (bag[j][k] > bag[j - xhd[i].pati][k - 1] + xhd[i].exp) ? bag[j][k] : (bag[j - xhd[i].pati][k - 1] + xhd[i].exp);

                }

            }

        }

        doudou = 0;

        for (j = 0; j <= mpati; j++)

        {

            for (k = 0; k <= mkill; k++)

            {

                if (bag[j][k] >= mexp)

                {

                    printf("%d\n", mpati - j);

                    doudou = 1;

                    break;

                }

            }

            if (doudou)

                break;

        }

        if (doudou == 0)

        {

            printf("-1\n");

        }

    }

    return 0;

}

1003 最长上升子列      if 要放到后面一起判断！！！；

#include <stdio.h>

#include <string.h>

int main()

{

    int n, m, i, j, k, u, c, temp, ar[100010], r[100010], cas, maxi, maxj, b, e;

    long long int max;

    while (scanf("%d", &n) != EOF)

    {

        cas = 0;

        for (m = 1; m <= n; m++)

        {

            cas++;

            scanf("%d", &c);

            for (i = 1; i <= c; i++)

            {

                scanf("%d", &ar[i]);

            }

            max = -1000000;

            r[0] = 0;

            b = 1;

            for (i = 1; i <= c; i++)

            {

                if (ar[i] > r[i - 1] + ar[i])

                {

                    r[i] = ar[i];

                    e = i;

                    b = i;

                }

                else

                {

                    r[i] = r[i - 1] + ar[i];

                    e = i;

                }

                if (max < r[i])

                {

                    max = r[i];

                    maxi = e;

                    maxj = b;

                }

            }

            printf("Case %d:\n", cas);

            printf("%lld %d %d\n", max, maxj, maxi);

            if(m != n)

                putchar('\n');

        }

    }

    return 0;

}

1160 mouse s speed   special judge  终于过了；   qsort模板；

第一个for遍历每一个老鼠i作为开头， 第二个for是每一个作为开头的老鼠i的下一个是j，那么num【i】= num【j】+1；

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int comp(const void \* p1, const void \* p2);

struct mice

{

    int speed;

    int wei;

    int index;

    int doudou;

};

int main()

{

    int n, m, i, j, k, u, sum, max, num[1010], maxi, b[1010], e;

    struct mice mouse[1010];

    n = 1;

    while (scanf("%d %d", &mouse[n].wei, &mouse[n].speed) != EOF)

    {

        mouse[n].index = n;

        mouse[n].doudou = 0;

        n++;

    }

    qsort(&mouse[1], (n - 1), sizeof(struct mice), comp);

    for (i = 0; i < 1010; i++)

    {

        b[i] = i;

        num[i] = 1;

    }

    max = 0;

    for (i = n-1; i >= 1; i--)

    {

        for (j = n-1; j > i; j--)

        {

            if (mouse[i].wei < mouse[j].wei && mouse[i].speed > mouse[j].speed && num[i] < num[j] + 1)

            {

                num[i] = num[j] + 1;

                b[i] = j;

            }

        }

        if (max < num[i])

        {

            max = num[i];

            maxi = i;

        }

    }

    printf("%d\n", max);

    for (i = maxi; b[i] != i; i=b[i])

    {

        printf("%d\n", mouse[i].index);

    }

    printf("%d\n", mouse[i].index);

    return 0;

}

int comp(const void \* p1, const void \* p2)

{

    const struct mice \* a1 = (const struct mice \*)p1;

    const struct mice \* a2 = (const struct mice \*)p2;

    if (a1->wei != a2->wei)

        return a1->wei - a2->wei;

    else

        return a2->speed - a1->speed;

}

1087  现在看来就是水题一道，hhh；

遍历每一个作为开头，遍历每一个开头的下一个， 找出最大和；

#include <stdio.h>

int main()

{

    int n, m, i, j, k, ar[1010];

    long long sum[1010], max;

    while (scanf("%d", &n) != EOF && n != 0)

    {

        for (i = 1; i <= n; i++)

            scanf("%d", &ar[i]);

        for (i = 1; i < 1010; i++)

            sum[i] = ar[i];

        max = 0;

        for (i = n; i >= 1; i--)

        {

            for (j = n; j > i; j--)

            {

                if (ar[i] < ar[j] && sum[i] < ar[i] + sum[j])

                    sum[i] = ar[i] + sum[j];

            }

            if (max < sum[i])

                max = sum[i];

        }

        printf("%lld\n", max);

    }

    return 0;

}

2111     分组背包；

#include <cstdio>

#include <cstring>

#include <cstdlib>

struct node

{

    int val;

    int m;

};

int main()

{

    struct node ar[150];

    int bag[150], i, j, k, u, n, a, b, cnt, count, tempp, tempm;

    while (scanf("%d", &a) != EOF && a)

    {

        scanf("%d", &b);

        for (i = 1; i <= b; i++)

        {

            scanf("%d %d", &ar[i].val, &ar[i].m);

        }

        memset(bag, 0, sizeof(bag));

        for (i = 1; i <= b; i++)

        {

            for (j = a; j >= 0; j--)

            {

                for (k = 1; k <= ar[i].m; k++)

                {

                    tempm = k;

                    tempp = k \* ar[i].val;

                    if (j < tempm)

                        continue;

                    bag[j] = (bag[j] > bag[j - tempm] + tempp) ? bag[j] : (bag[j - tempm] + tempp);

                }

            }

        }

        printf("%d\n", bag[a]);

    }

    return 0;

}

1305  字典树第一题， 只创建树就行了，不用查找；

#include <stdio.h>

#include <stdlib.h>

struct binary

{

    int sta;

    struct binary\* next[2];

};

int insert(struct binary \*root, char \*ar);

int main()

{

    int i, doudou = 0, cas = 0;

    char ar[50];

    struct binary \*root = (struct binary \*)malloc(sizeof(struct binary));

    for (i = 0; i < 2; i++)

    {

        root->next[i] = NULL;

    }

    root -> sta = 0;

    while (scanf("%s", ar) != EOF)

    {

        if (ar[0] == '9')

        {

            cas++;

            if (doudou == 0)

                printf("Set %d is immediately decodable\n", cas);

            else

                printf("Set %d is not immediately decodable\n", cas);

            continue;

        }

        getchar();

        if (insert(root, ar))

            doudou = 1;

        else

            doudou = 0;

    }

    return 0;

}

int insert(struct binary \*root, char \*ar)

{

    int i, doudou = 0;

    struct binary \*p, \*temp;

    p = root;

    while (\*ar != '\0')

    {

        if (p->next[\*ar - '0'] == NULL)

        {

            if (p->sta == 1)

                doudou = 1;

            temp = (struct binary \*)malloc(sizeof(struct binary));

            for (i = 0; i < 2; i++)

                temp->next[i] = NULL;

            temp->sta = 0;

            p->next[\*ar - '0'] = temp;

        }

        else

        {

            if (\*(ar + 1) == '\0')

                doudou = 1;

        }

        p = p->next[\*ar - '0'];

        ar++;

    }

    p->sta = 1;

    return doudou;

}

1251   字典树第二题，不能在结构体内初始化变量；

#include <stdio.h>

#include <stdlib.h>

struct pre

{

    int count;

    struct pre \*next[26];

};

void insert(struct pre \*root, char \*ar);

int search(struct pre \*root, char \*ar);

void del(struct pre \*root);

int main()

{

    int i, j, n, m;

    char ar[11];

    struct pre \* root;

    root = (struct pre \*)malloc(sizeof(struct pre));

    for (i = 0; i < 26; i++)

    {

        root->next[i] = NULL;

    }

    root->count = 0;

    while (gets(ar) != NULL && ar[0] != '\0')

    {

        insert(root, ar);

    }

    while (gets(ar) != NULL && ar[0] != '\0')

    {

        printf("%d\n", search(root, ar));

    }

    del(root);

    return 0;

}

void insert(struct pre \*root, char \*ar)

{

    int i, c;

    struct pre \*p = root, \*temp;

    while (\*ar != '\0')

    {

        if (p->next[\*ar - 'a'] == NULL)

        {

            temp = (struct pre \*)malloc(sizeof(struct pre));

            for (i = 0; i < 26; i++)

            {

                temp->next[i] = NULL;

            }

            temp->count = 0;

            p->next[\*ar - 'a'] = temp;

        }

        p = p->next[\*ar - 'a'];

        p->count++;

        ar++;

    }

}

int search(struct pre \*root, char \*ar)

{

    int i;

    struct pre \*p = root;

    while (\*ar != '\0' && p != NULL)

    {

        p = p->next[\*ar - 'a'];

        if (\*(ar + 1) == '\0' && p != NULL)

            return p->count;

        ar++;

    }

    if (p == NULL)

        return 0;

}

void del(struct pre \*root)

{

    int i;

    for (i = 0; i < 26; i++)

    {

        if (root->next[i] != NULL)

            del(root->next[i]);

    }

    free(root);

}

3460 字典树第5题，他输完一个名字必须删了才能输下一个，只是有的不用删完；公式：创建次数\*2 - 最长字符串长度 + 字符串数；

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct print

{

    struct print \* next[26];

};

void insert(struct print \* root, char \*s, long long int \*S);

void del(struct print \*root);

int main()

{

    int i, j, n, m, L, maxl;

    char ar[55];

    long long sum, \*S;

    struct print \*root;

    while (scanf("%d", &n) != EOF)

    {

        sum = 0;

        maxl = 0;

        S = &sum;

        root = (struct print \*)malloc(sizeof(struct print));

        for (i = 0; i < 26; i++)

            root->next[i] = NULL;

        for (m = 1; m <= n; m++)

        {

            scanf("%s", ar);

            if (maxl < (L = strlen(ar)))

                maxl = L;

            insert(root, ar, S);

        }

        printf("%lld\n", sum - maxl + n);

        del(root);

    }

    return 0;

}

void del(struct print \*root)

{

    int i;

    for (i = 0; i < 26; i++)

        if (root->next[i] != NULL)

            del(root->next[i]);

    free(root);

}

void insert(struct print \* root, char \*s, long long  int \*S)

{

    int i;

    struct print \*temp, \*p = root;

    while (\*s != '\0')

    {

        if (p->next[\*s - 'a'] == NULL)

        {

            temp = (struct print \*)malloc(sizeof(struct print));

            for (i = 0; i < 26; i++)

                temp->next[i] = NULL;

            \*S += 2;

            p->next[\*s - 'a'] = temp;

        }

        p = p->next[\*s - 'a'];

        s++;

    }

}

poj1   矩阵快速幂第1题，重点：1、构造矩阵；2、用结构体存矩阵；3、a矩阵和res矩阵的初始化；4、取模；

#include <stdio.h>

#include <string.h>

struct juz

{

    int ar[2][2];

};

struct juz multiply(struct juz a, struct juz b, int n);

int main()

{

    int i, j, u, k, n, m;

    struct juz a, res;

    while (scanf("%d", &n) != EOF && n != -1)

    {

        m = n;

        a.ar[0][0] = 1;

        a.ar[0][1] = 1;

        a.ar[1][0] = 1;

        a.ar[1][1] = 0;

        res.ar[0][0] = 0;

        res.ar[0][1] = 1;

        while (m > 0)

        {

            if (m % 2)

            {

                res = multiply(res, a, 2);

            }

            a = multiply(a, a, 2);

            m /= 2;

        }

        printf("%d\n", res.ar[0][0]);

    }

    return 0;

}

struct juz multiply(struct juz a, struct juz b, int n)

{

    struct juz temp;

    int i, j, k;

    memset(temp.ar, 0, sizeof(int)\*n\*n);

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

        {

            for (k = 0; k < n; k++)

            {

                temp.ar[i][j] += ((a.ar[i][k] % 10000) \* (b.ar[k][j] % 10000)) % 10000;

                temp.ar[i][j] %= 10000;

            }

        }

    }

    return temp;

}

2604   矩阵快速幂第4题

#include <stdio.h>

#include <string.h>

struct juz

{

    int ar[5][5];

};

struct juz multiply(struct juz a, struct juz b, int n, int m);

int main()

{

    int i, j, k, u, n, m, t;

    struct juz a, res;

    while (scanf("%d %d", &n, &u) != EOF)

    {

        memset(res.ar, 0, sizeof(res.ar));

        memset(a.ar, 0, sizeof(a.ar));

        a.ar[0][0] = 1;

        a.ar[0][1] = 1;

        a.ar[1][2] = 1;

        a.ar[2][0] = 1;

        a.ar[2][3] = 1;

        a.ar[3][0] = 1;

        res.ar[0][0] = 6;

        res.ar[0][1] = 4;

        res.ar[0][2] = 2;

        res.ar[0][3] = 1;

        if (n == 0)

        {

            printf("%d\n", 1 % u );

            continue;

        }

        else if (n == 1)

        {

            printf("%d\n", 2%u);

            continue;

        }

        else if (n == 2)

        {

            printf("%d\n", 3%u);

            continue;

        }

        else if (n == 3)

        {

            printf("%d\n", 6%u);

            continue;

        }

        t = n - 3;

        while (t > 0)

        {

            if (t % 2)

            {

                res = multiply(res, a, 4, u);

            }

            a = multiply(a, a, 4, u);

            t /= 2;

        }

        printf("%d\n", res.ar[0][0]);

    }

    return 0;

}

struct juz multiply(struct juz a, struct juz b, int n, int m)

{

    int i, j, k;

    struct juz temp;

    memset(temp.ar, 0, sizeof(temp.ar));

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

        {

            for (k = 0; k < n; k++)

            {

                temp.ar[i][j] += ((a.ar[i][k] % m)\*(b.ar[k][j] % m)) % m;

                temp.ar[i][j] %= m;

            }

        }

    }

    return temp;

}

2544                    最短路 第8题（dijkstra）

#include <stdio.h>

#define MAXINT 10000000

int dijkstra(int (\*ar)[101], int b, int e, int n);

int main()

{

    int i, j, k, u, n, m, a, b, c, temp, (\*s)[101], sa[101][101];

    while (scanf("%d %d", &n, &m) != EOF && (n || m))

    {

        for (u = 1; u <= n; u++)

            for (k = 1; k <= n; k++)

                sa[u][k] = MAXINT;

        for (u = 1; u <= m; u++)

        {

            scanf("%d %d %d", &a, &b, &c);

            sa[b][a] = c;

            sa[a][b] = c;

        }

        s = sa;

        printf("%d\n", dijkstra(s, 1, n, n));

    }

    return 0;

}

int dijkstra(int(\*ar)[101], int b, int e, int n)

{

    bool S[101];                                         // 判断是否已存入该点到S集合中

    int dist[101], pre[101];

    int i, j, k, u, mindist;

    for (i = 1; i <= e; i++)

    {

        dist[i] = ar[b][i];

        S[i] = false;                                    // 初始都未用过该点

        if (dist[i] == MAXINT)

            pre[i] = -1;

        else

            pre[i] = b;

    }

    dist[b] = 0;

    S[b] = true;

    for (i = 2; i <= e; i++)

    {

        mindist = MAXINT;

        for (j = 1; j <= e; j++)                                 // 找出当前未使用的点j的dist[j]最小值

        {

            if (S[j] == false && dist[j] < mindist)

            {

                u = j;                                              // u保存当前邻接点中距离最小的点的号码

                mindist = dist[j];

            }

        }

        S[u] = true;

        for (j = 1; j <= e; j++)

        {

            if (S[j] == false && dist[u] + ar[u][j] < dist[j])                      //在通过新加入的u点路径找到离begin点更短的路径

            {

                dist[j] = dist[u] + ar[u][j];                                                   //更新dist

                pre[j] = u;                                                                         //找到前一个点

            }

        }

    }

    return dist[e];

}

1217     最短路第1题；（floyd）  找出字符串的下标,用下标表示dist的横纵坐标;

#include <stdio.h>

#include <string.h>

void floyd(double(\*dist)[31], int(\*path)[31], int n);

int main()

{   int i, j, k, u, m, n, x, y, path[31][31], (\*pat)[31], doudou, cas = 0;

    double dist[31][31], temp, (\*dis)[31];

    char ar[31][100], a[100], b[100];

    while (scanf("%d", &n) != EOF && n)

    {

        cas++;

        for (i = 1; i <= n; i++)

        {

            scanf("%s", ar[i]);

        }

        for (i = 0; i <= n; i++)

        {

            for (j = 0; j <= n; j++)

            {

                dist[i][j] = 0;

                if (i == j)

                    dist[i][j] = 0;

                path[i][j] = j;

            }

        }

        scanf("%d", &m);

        for (i = 1; i <= m; i++)

        {

            scanf("%s %lf %s", a, &temp, b);

            for (j = 1; j <= n; j++)

            {

                if (strcmp(ar[j], a) == 0)

                {

                    x = j;

                    break;

                }

            }

            for (j = 1; j <= n; j++)

            {

                if (strcmp(ar[j], b) == 0)

                {

                    y = j;

                    break;

                }

            }

            dist[x][y] = temp;

        }

        dis = dist;

        pat = path;

        floyd(dis, pat, n);

        doudou = 0;

        for (i = 1; i <= n; i++)

            if (dist[i][i] > 1)

                doudou = 1;

        if (doudou)

            printf("Case %d: Yes\n", cas);

        else

            printf("Case %d: No\n", cas);

    }

    return 0;

}

void floyd(double(\*dist)[31], int(\*path)[31], int n)

{

    int i, j, k, u;

    for (k = 1; k <= n; k++)

    {

        for (i = 1; i <= n; i++)

        {

            for (j = 1; j <= n; j++)

            {

                if (dist[i][j] < dist[i][k] \* dist[k][j])

                {

                    dist[i][j] = dist[i][k] \* dist[k][j];

                    path[i][j] = path[i][k];

                }

            }

        }

    }

}

 太坑了  有可能相同的2个村庄有好多条长度不同的路

注意按字典序输出，只需多判断一步dist[i][j]==dist[i][k]+dist[k][j]+ar[k] && path[i][j] > path[k][j];

void floyd(int(\*dist)[1002], int(\*path)[1002], int \*ar, int n)

{

    int i, j, k;

    for (k = 1; k <= n; k++)

    {

        for (i = 1; i <= n; i++)

        {

            for (j = 1; j <= n; j++)

            {

                if (dist[i][j] > dist[i][k] + dist[k][j] + ar[k])

                {

                    dist[i][j] = dist[i][k] + dist[k][j] + ar[k];

                    path[i][j] = path[i][k];

                }

                else if (dist[i][j] == dist[i][k] + dist[k][j] + ar[k])

                {

                    if (path[i][j] > path[i][k])

                    {

                        path[i][j] = path[i][k];

                    }

                }

            }

        }

    }

}

1869   最短路第4题     没有真正理解Floyd的含义; (认识距离为0, 通过几个人认识距离为几就行了)

#include <stdio.h>

#define MAXINT 10000000

void floyd(int(\*dist)[101], int n);

int main()

{

    int i, j, k, u, n, m, a, b, dist[101][101], doudou;

    while (scanf("%d %d", &n, &m) != EOF)

    {

        for (i = 0; i < n; i++)

        {

            for (j = 0; j < n; j++)

            {

                dist[i][j] = MAXINT;

                if (i == j)

                    dist[i][j] = 0;

            }

        }

        for (u = 0; u < m; u++)

        {

            scanf("%d %d", &a, &b);

            dist[a][b] = 1;

            dist[b][a] = 1;

        }

        floyd(dist, n);

        doudou = 1;

        for (i = 0; i < n; i++)

        {

            for (j = 0; j < n; j++)

            {

                if (dist[i][j] > 7)

                {

                    doudou = 0;

                    break;

                }

            }

            if (doudou == 0)

                break;

        }

        if (doudou == 1)

            printf("Yes\n");

        else

            printf("No\n");

    }

}

void floyd(int(\*dist)[101], int n)

{

    int i, j, k, u;

    for (k = 0; k < n; k++)

    {

        for (i = 0; i < n; i++)

        {

            for (j = 0; j < n; j++)

            {

                if (dist[i][j] > dist[i][k] + dist[k][j])

                {

                    dist[i][j] = dist[i][k] + dist[k][j];

                }

            }

        }

    }

}

2544  朴素SPFA；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <queue>

#define MAX 100000000

using namespace std;

int num[110], ar[110][110], dist[110], n, m;

bool visit[110];

void SPFA();

int main()

{

    int i, j, k, u, a, b, c;

    while (scanf("%d %d", &n, &m) != EOF && (n || m))

    {

        for (i = 0; i <= n; i++)

        {

            for (j = 0; j <= n; j++)

            {

                ar[i][j] = MAX;

            }

        }

        for (u = 1; u <= m; u++)

        {

            scanf("%d %d %d", &a, &b, &c);

            ar[a][b] = c;

            ar[b][a] = c;

        }

        SPFA();

        printf("%d\n", dist[n]);

    }

    return 0;

}

void SPFA()

{

    int i, j, k, u, near;

    queue<int>q;

    memset(num, 0, sizeof(num));

    memset(visit, false, sizeof(visit));

    for (i = 0; i <= n; i++)

        dist[i] = MAX;

    dist[1] = 0;

    q.push(1);

    while (!q.empty())

    {

        near = q.front();

        q.pop();

        visit[near] = false;

        for (i = 1; i <= n; i++)

        {

            if (dist[i] > dist[near] + ar[near][i])

            {

                dist[i] = dist[near] + ar[near][i];

                if (!visit[i])

                {

                    visit[i] = true;

                    num[i]++;

                    if (num[i] >= n)

                        return;

                    q.push(i);

                }

            }

        }

    }

}

2544   dijkstra+堆优化 模板;

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <queue>

#include<vector>

#define MAX 100000000

using namespace std;

struct node

{

    int now;

    int val;

    int next;

};

struct node ar[20010];

typedef pair<int, int> P;

struct comp

{

    bool operator () (P a, P b)

    {

        return a.first > b.first;

    }

};

int dist[10010], head[10010], n, m, count;

void P\_dijkstra();

int main()

{

    int i, j, k, u, a, b, c, count;

    while (scanf("%d %d", &n, &m) != EOF && (n || m))

    {

        count = 0;

        memset(head, -1, sizeof(head));

        for (i = 0; i <= n; i++)

            dist[i] = MAX;

        for (u = 1; u <= m; u++)

        {

            scanf("%d %d %d", &a, &b, &c);

            ar[count].now = b;

            ar[count].val = c;

            ar[count].next = head[a];

            head[a] = count;

            count++;

            ar[count].now = a;

            ar[count].val = c;

            ar[count].next = head[b];

            head[b] = count;

            count++;

        }

        P\_dijkstra();

        printf("%d\n", dist[n]);

    }

    return 0;

}

void P\_dijkstra()

{

    int i, j, k, u;

    priority\_queue<P, vector<P>, comp>q;

    P temp, after;

    temp.first = 0;

    temp.second = 1;

    dist[1] = 0;

    q.push(temp);

    while (!q.empty())

    {

        temp = q.top();

        q.pop();

        if (dist[temp.second] < temp.first)

            continue;

        for (i = head[temp.second]; i != -1; i = ar[i].next)

        {

            after.first = ar[i].val;

            after.second = ar[i].now;

            if (dist[after.second] > dist[temp.second] + after.first)

            {

                dist[after.second] = dist[temp.second] + after.first;

                q.push(after);

            }

        }

    }

}

2544  SPFA 前向星+双向队列 优化 模板；

#include <cstdio>

#include <cstring>

#include <deque>

#include <vector>

#include <algorithm>

#define MAX 100000000

using namespace std;

struct node

{

    int now;

    int val;

    int next;

};

struct node ar[20010];

int head[10010], dist[10010], num[10010], n, m;

bool visit[10010];

void P\_SPFA();

int main()

{

    int i, j, k, u, a, b, c, count;

    while (scanf("%d %d", &n, &m) != EOF && (n || m))

    {

        count = 0;

        memset(head, -1, sizeof(head));

        memset(visit, false, sizeof(visit));

        memset(num, 0, sizeof(num));

        for (u = 1; u <= m; u++)

        {

            scanf("%d %d %d", &a, &b, &c);

            ar[count].now = b;

            ar[count].val = c;

            ar[count].next = head[a];

            head[a] = count;

            count++;

            ar[count].now = a;

            ar[count].val = c;

            ar[count].next = head[b];

            head[b] = count;

            count++;

        }

        P\_SPFA();

        printf("%d\n", dist[n]);

    }

    return 0;

}

void P\_SPFA()

{

    int i, j, k, u, temp, after;

    deque<int>q;

    for (i = 0; i <= n; i++)

        dist[i] = MAX;

    dist[1] = 0;

    temp = 1;

    q.push\_back(temp);

    while (!q.empty())

    {

        temp = q.front();

        q.pop\_front();

        visit[temp] = false;

        for (i = head[temp]; i != -1; i = ar[i].next)

        {

            after = ar[i].now;

            if (dist[after] > dist[temp] + ar[i].val)

            {

                dist[after] = dist[temp] + ar[i].val;

                if (!visit[after])

                {

                    visit[after] = true;

                    num[after]++;

                    if (num[after] >= n)

                        return;

                    if (!q.empty() && dist[after] < dist[q.front()])

                        q.push\_front(after);

                    else

                        q.push\_back(after);

                }

            }

        }

    }

}

1224   SPFA + 链式；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <vector>

#include <algorithm>

#include <deque>

#define MIN -10000000

#define MAX 100000000

using namespace std;

struct node

{

    int now;

    int val;

    int next;

};

struct node ar[20010];

int dist[110], head[110], pre[110], set[110], visit[110], k;

void SPFA();

void out(int q);

int main()

{

    int i, j, u, a, b, c, n, m, count;

    while (scanf("%d", &n) != EOF)

    {

        for (m = 1; m <= n; m++)

        {

            scanf("%d", &k);

            for (u = 1; u <= k; u++)

            {

                scanf("%d", &set[u]);

            }

            set[k + 1] = MAX;

            count = 0;

            memset(head, -1 , sizeof(head));

            scanf("%d", &c);

            for (u = 1; u <= c; u++)

            {

                scanf("%d %d", &a, &b);

                ar[count].now = b;

                ar[count].val = set[b];

                ar[count].next = head[a];

                head[a] = count++;

            }

            SPFA();

            printf("CASE %d#\n", m);

            printf("points : %d\n", dist[k + 1] - MAX);

            printf("circuit : ");

            out(pre[k+1]);

            printf("1\n");

            if (m < n)

                printf("\n");

        }

    }

    return 0;

}

void SPFA()

{

    int i, j, u, h, v;

    queue<int>q;

    memset(visit, false, sizeof(visit));

    memset(pre, -1, sizeof(pre));

    for (i = 1; i <= k + 1; i++)

        dist[i] = MIN;

    dist[1] = 0;

    visit[1] = true;

    q.push(1);

    while (!q.empty())

    {

        u = q.front();

        q.pop();

        visit[u] = false;

        for (i = head[u]; i != -1; i = ar[i].next)

        {

            v = ar[i].now;

            if (dist[v] < dist[u] + ar[i].val)

            {

                dist[v] = dist[u] + ar[i].val;

                pre[v] = u;

                if (!visit[v])

                {

                    visit[v] = true;

                    q.push(v);

                }

            }

        }

    }

}

void out(int q)

{

    if (pre[q] != -1)

        out(pre[q]);

    printf("%d->", q);

    return;

}

2112    地名是字符串的   最短路第7题 dijkstra + 堆优化；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <vector>

#include <algorithm>

#define MAX 100000000

using namespace std;

struct node

{

    int now;

    int val;

    int next;

};

struct node ar[20010];

int dist[200], head[200], n, m, countt;

char sa[200][80], a[80], b[80];

typedef pair<int, int>P;

struct comp

{

    bool operator () (P a, P b)

    {

        return a.first > b.first;

    }

};

void P\_dijkstra();

int main()

{

    int i, j, k, u, x1, x2, cnt, c;

    while (scanf("%d", &n) != EOF && n != -1)

    {

        countt = 0;

        scanf("%s %s", sa[countt], sa[countt + 1]);

        countt += 2;

        cnt = 0;

        memset(head, -1, sizeof(head));

        for (m = 1; m <= n; m++)

        {

            scanf("%s %s %d", a, b, &c);

            x1 = MAX;

            x2 = MAX;

            for (j = 0; j < countt; j++)

            {

                if (strcmp(a, sa[j]) == 0)

                {

                    x1 = j;

                    break;

                }

            }

            if (x1 == MAX)

            {

                strcpy(sa[countt], a);

                x1 = countt;

                countt++;

            }

            for (j = 0; j < countt; j++)

            {

                if (strcmp(b, sa[j]) == 0)

                {

                    x2 = j;

                    break;

                }

            }

            if (x2 == MAX)

            {

                strcpy(sa[countt], b);

                x2 = countt;

                countt++;

            }

            ar[cnt].now = x2;

            ar[cnt].val = c;

            ar[cnt].next = head[x1];

            head[x1] = cnt++;

            ar[cnt].now = x1;

            ar[cnt].val = c;

            ar[cnt].next = head[x2];

            head[x2] = cnt++;

        }

        if (strcmp(sa[0], sa[1]) == 0)

        {

            printf("0\n");

            continue;

        }

        P\_dijkstra();

        if (dist[1] != MAX)

            printf("%d\n", dist[1]);

        else

            printf("-1\n");

    }

    return 0;

}

void P\_dijkstra()

{

    int i, j, k, u;

    P temp, after;

    for (i = 0; i <= countt; i++)

        dist[i] = MAX;

    dist[0] = 0;

    priority\_queue<P, vector<P>, comp>q;

    temp.first = 0;

    temp.second = 0;

    q.push(temp);

    while (!q.empty())

    {

        temp = q.top();

        q.pop();

        if (dist[temp.second] < temp.first)

            continue;

        for (i = head[temp.second]; i != -1; i = ar[i].next)

        {

            after.first = ar[i].val;

            after.second = ar[i].now;

            if (dist[after.second] > dist[temp.second] + after.first)

            {

                dist[after.second] = dist[temp.second] + after.first;

                q.push(after);

            }

        }

    }

}

3790  dijkstra+堆优化  最短路第13题（双权值问题， 多加一个和dist一样功能的数组）；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <vector>

#include <algorithm>

#define MAX 100000000

using namespace std;

struct node

{

    int now;

    int val;

    int next;

    int cost;

};

typedef pair <int, int> P;

struct comp

{

    bool operator () (P a, P b)

    {

        return a.first > b.first;

    }

};

struct node ar[200010];

int dist[1010], head[1010], cost[1010];

int n, m, s, t;

void P\_dijkstra();

int main()

{

    int i, j, k, u, a, b, c, d, count;

    while (scanf("%d %d", &n, &m) != EOF && (n || m))

    {

        count = 0;

        memset(head, -1, sizeof(head));

        for (u = 1; u <= m; u++)

        {

            scanf("%d %d %d %d", &a, &b, &c, &d);

            ar[count].now = b;

            ar[count].val = c;

            ar[count].cost = d;

            ar[count].next = head[a];

            head[a] = count++;

            ar[count].now = a;

            ar[count].val = c;

            ar[count].cost = d;

            ar[count].next = head[b];

            head[b] = count++;

        }

        scanf("%d %d", &s, &t);

        P\_dijkstra();

        printf("%d %d\n", dist[t], cost[t]);

    }

    return 0;

}

void P\_dijkstra()

{

    int i, j, k, u;

    P temp, after;

    priority\_queue<P, vector<P>, comp>q;

    for (i = 0; i <= n; i++)

    {

        dist[i] = MAX;

        cost[i] = MAX;

    }

    dist[s] = 0;

    cost[s] = 0;

    temp.first = 0;

    temp.second = s;

    q.push(temp);

    while (!q.empty())

    {

        temp = q.top();

        q.pop();

        if (dist[temp.second] < temp.first)

            continue;

        for (i = head[temp.second]; i != -1; i = ar[i].next)

        {

            after.first = ar[i].val;

            after.second = ar[i].now;

            if (dist[after.second] > dist[temp.second] + after.first)

            {

                dist[after.second] = dist[temp.second] + after.first;

                cost[after.second] = cost[temp.second] + ar[i].cost;

                q.push(after);

            }

            if (dist[after.second] == dist[temp.second] + after.first && cost[after.second] > cost[temp.second] + ar[i].cost)

            {

                dist[after.second] = dist[temp.second] + after.first;

                cost[after.second] = cost[temp.second] + ar[i].cost;

                q.push(after);

            }

        }

    }

}

2923   WA17次！！！    复习floyd + 认识map；

#include <cstdio>

#include <string>

#include <cstring>

#include <algorithm>

#include <map>

#define MAX 999999999

using namespace std;

int n, c, r;

map<string, int> ar;

int dist[1100][1100];

char sa[1100][110];

void floyd();

int main()

{

    int i, j, k, u, sum, count, cas = 1, d, x, y;

    char a[110], b[110], f, g;

    while (scanf("%d %d %d", &n, &c, &r) != EOF && (n || c || r))

    {

        ar.clear();

        sum = 0;

        for (i = 0; i <= n; i++)

        {

            for (j = 0; j <= n; j++)

            {

                dist[i][j] = MAX;

            }

        }

        for (i = 0; i <= c; i++)

        {

            scanf("%s", sa[i]);

        }

        count = 1;

        for (i = 0; i < r; i++)

        {

            scanf("%s %c-%d-%c %s", a, &f, &d, &g, b);

            if (!ar[a])

                ar[a] = count++;

            x = ar[a];

            if (!ar[b])

                ar[b] = count++;

            y = ar[b];

            if (f == '<' && d < dist[y][x])

                dist[y][x] = d;

            if (g == '>' && d < dist[x][y])

                dist[x][y] = d;

        }

        floyd();

        for (i = 1; i <= n; i++)

        {

            for (j = 1; j <= n; j++)

            {

                printf("%d ", dist[i][j]);

            }

            putchar('\n');

        }

        for (i = 1; i <= c; i++)

            sum += dist[ar[sa[0]]][ar[sa[i]]] + dist[ar[sa[i]]][ar[sa[0]]];

        printf("%d. %d\n", cas++, sum);

    }

    return 0;

}

void floyd()

{

    int i, j, k;

    for (k = 1; k <= n; k++)

    {

        for (i = 1; i <= n; i++)

        {

            for (j = 1; j <= n; j++)

            {

                if (dist[i][j] > dist[i][k] + dist[k][j])

                    dist[i][j] = dist[i][k] + dist[k][j];

            }

        }

    }

}

3339   dijkstra堆优化 + 01背包，化找最小为找最大， 代价和体积换位置；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <vector>

#include <algorithm>

#define MAX 999999999

using namespace std;

struct node

{

    int now;

    int val;

    int next;

};

struct node ar[20010];

typedef pair<int, int> P;

struct comp

{

    bool operator () (P a, P b)

    {

        return a.first > b.first;

    }

};

int dist[110], head[110], n, m;

void P\_dijkstra();

int main()

{

    int t, cas, i, j, k, u, a, b, c, count, sum, S, power[110], bag[10010];

    while (scanf("%d", &t) != EOF)

    {

        for (cas = 1; cas <= t; cas++)

        {

            scanf("%d %d", &n, &m);

            count = 0;

            memset(head, -1, sizeof(head));

            for (i = 1; i <= m; i++)

            {

                scanf("%d %d %d", &a, &b, &c);

                ar[count].now = b;

                ar[count].val = c;

                ar[count].next = head[a];

                head[a] = count++;

                ar[count].now = a;

                ar[count].val = c;

                ar[count].next = head[b];

                head[b] = count++;

            }

            sum = 0;

            for (i = 1; i <= n; i++)

            {

                scanf("%d", &power[i]);

                sum += power[i];

            }

            P\_dijkstra();

            memset(bag, 0, sizeof(bag));

            S = 0;

            for (i = 0; i <= n; i++)

                if (dist[i] != MAX)

                    S += dist[i];

            for (i = 1; i <= n; i++)

            {

                for (j = S; j >= dist[i]; j--)

                {

                    bag[j] = (bag[j] > bag[j - dist[i]] + power[i]) ? bag[j] : (bag[j - dist[i]] + power[i]);

                }

            }

            for (i = 0; i < S; i++)

            {

                if (bag[i] > sum / 2)

                {

                    printf("%d\n", i);

                    break;

                }

            }

            if (i == S)

                printf("impossible\n");

        }

    }

    return 0;

}

void P\_dijkstra()

{

    int i, j, k;

    P temp, after;

    priority\_queue<P, vector<P>, comp>q;

    for (i = 0; i <= n; i++)

        dist[i] = MAX;

    dist[0] = 0;

    temp.first = 0;

    temp.second = 0;

    q.push(temp);

    while (!q.empty())

    {

        temp = q.top();

        q.pop();

        if (dist[temp.second] < temp.first)

            continue;

        for (i = head[temp.second]; i != -1; i = ar[i].next)

        {

            after.second = ar[i].now;

            after.first = ar[i].val;

            if (dist[after.second] > dist[temp.second] + after.first)

            {

                dist[after.second] = dist[temp.second] + after.first;

                q.push(after);

            }

        }

    }

}

1242  搜索第5题 bfs   普通队列  一 ！ 定！要！ 搜！完！  不能中间return；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

using namespace std;

struct node

{

    int x;

    int y;

    int time;

};

struct node now, after;

int visit[205][205], a, b;

int dir[4][2] = { 0, 1, 0, -1, -1, 0, 1, 0 }; //右左上下

char ar[205][205];

void bfs(int x1, int y1, int x2, int y2);

int main()

{

    int i, j, k, temp, min, x1, y1, x2, y2, n, m;

    while (scanf("%d %d", &a, &b) != EOF)

    {

        getchar();

        for (i = 1; i <= a; i++)

        {

            for (j = 1; j <= b; j++)

            {

                scanf("%c", &ar[i][j]);

                if (ar[i][j] == 'a')

                {

                    x2 = i;

                    y2 = j;

                }

            }

            getchar();

        }

        min = 100000000;

        for (i = 1; i <= a; i++)

        {

            for (j = 1; j <= b; j++)

            {

                if (ar[i][j] == 'r')

                {

                    x1 = i;

                    y1 = j;

                    for (n = 0; n <= a; n++)

                    {

                        for (m = 0; m <= b; m++)

                            visit[n][m] = 1000000;

                    }

                    bfs(x1, y1, x2, y2);

                    temp = visit[x2][y2];

                    if (temp != 1000000)

                        min = (min <= temp) ? min : temp;

                }

            }

        }

        if (min != 100000000)

            printf("%d\n", min);

        else

            printf("Poor ANGEL has to stay in the prison all his life.\n");

    }

    return 0;

}

void bfs(int x1, int y1, int x2, int y2)

{

    queue<node>q;

    int i, j;

    now.x = x1;

    now.y = y1;

    now.time = 0;

    q.push(now);

    while (!q.empty())

    {

        now = q.front();

        q.pop();

        for (i = 0; i < 4; i++)

        {

            after.x = now.x + dir[i][0];

            after.y = now.y + dir[i][1];

            after.time = now.time + 1;

            if (after.x > a || after.x < 1 || after.y > b || after.y < 1)

                continue;

            if (ar[after.x][after.y] == '#')

                continue;

            if (ar[after.x][after.y] == 'x')

            {

                after.time++;

                /\*ar[after.x][after.y] = '.';\*/

            }

            if ((ar[after.x][after.y] == '.' || ar[after.x][after.y] == 'a' || ar[after.x][after.y] == 'r' || ar[after.x][after.y] == 'x') && after.time < visit[after.x][after.y])

            {

                q.push(after);

                visit[after.x][after.y] = after.time;

            }

        }

    }

    return;

}

1180  搜索第3题bfs；梯子那题

#include <cstdio>

#include <cstring>

#include <queue>

using namespace std;

struct node

{

    int x;

    int y;

    int time;

    int sta;

};

struct node now, after, temp;

queue<node>q;

int a, b, visit[25][25], dir[4][2] = { { 0, 1 },{ 0, -1 },{ -1, 0 },{ 1, 0 } };

char ar[25][25];

int bfs(int x1, int y1, int x2, int y2);

int main()

{

    int i, j, k, n, m, x1, x2, y1, y2;

    struct node temp;

    while (scanf("%d %d", &a, &b) != EOF)

    {

        getchar();

        for (i = 1; i <= a; i++)

        {

            for (j = 1; j <= b; j++)

            {

                scanf("%c", &ar[i][j]);

                if (ar[i][j] == 'S')

                {

                    x1 = i;

                    y1 = j;

                }

                if (ar[i][j] == 'T')

                {

                    x2 = i;

                    y2 = j;

                }

            }

            getchar();

        }

        /\*printf("%d %d %d %d\n", x1, y1, x2, y2);

        printf("%c %c\n", ar[x1][y1], ar[x2][y2]);\*/

        for (i = 0; i < 25; i++)

        {

            for (j = 0; j < 25; j++)

                visit[i][j] = 1000;

        }

        visit[x1][y1] = 0;

        while (!q.empty())

            q.pop();

        for (i = 0; i < 4; i++)

        {

            temp.x = x1;

            temp.y = y1;

            temp.sta = i;

            temp.time = 0;

            q.push(temp);

        }

        i = bfs(x1, y1, x2, y2);

        printf("%d\n", visit[x2][y2]);

    }

    return 0;

}

int bfs(int x1, int y1, int x2, int y2)

{

    int i, j;

    /\*now.x = x1;

    now.y = y1;

    now.sta = -1;

    now.time = 0;

    visit[x1][y1] = 0;

    q.push(now);\*/

    while (!q.empty())

    {

        now = q.front();

        q.pop();

        /\*printf("%d %d\n", now.x, now.y);

        printf("%c\n", ar[now.x][now.y]);\*/

        for (i = 0; i < 4; i++)

        {

            after.x = now.x + dir[i][0];

            after.y = now.y + dir[i][1];

            after.sta = i;

            after.time = now.time + 1;

            if (after.x < 1 || after.x > a || after.y < 1 || after.y > b)

                continue;

            if (ar[after.x][after.y] == '\*')

                continue;

            if (ar[after.x][after.y] == '|')

            {

                if (now.time % 2 == 0)

                {

                    if (after.sta == 0)

                        if (ar[after.x][after.y + 1] != '\*')

                        {

                            after.y = after.y + 1;

                            after.time++;

                        }

                    if (after.sta == 1)

                        if (ar[after.x][after.y - 1] != '\*')

                        {

                            after.y = after.y - 1;

                            after.time++;

                        }

                    if (after.sta == 2)

                    {

                        if (ar[after.x - 1][after.y] != '\*')

                            after.x = after.x - 1;

                    }

                    if (after.sta == 3)

                    {

                        if (ar[after.x + 1][after.y] != '\*')

                            after.x = after.x + 1;

                    }

                }

                if (now.time % 2 != 0)

                {

                    if (after.sta == 0)

                    {

                        if (ar[after.x][after.y + 1] != '\*')

                            after.y = after.y + 1;

                    }

                    if (after.sta == 1)

                    {

                        if (ar[after.x][after.y - 1] != '\*')

                            after.y = after.y - 1;

                    }

                    if (after.sta == 2)

                    {

                        if (ar[after.x - 1][after.y] != '\*')

                        {

                            after.x = after.x - 1;

                            after.time++;

                        }

                    }

                    if (after.sta == 3)

                    {

                        if (ar[after.x + 1][after.y] != '\*')

                        {

                            after.x = after.x + 1;

                            after.time++;

                        }

                    }

                }

            }

            if (ar[after.x][after.y] == '-')

            {

                if (now.time % 2 == 0)

                {

                    if (after.sta == 0)

                    {

                        if (ar[after.x][after.y + 1] != '\*')

                            after.y = after.y + 1;

                    }

                    if (after.sta == 1)

                    {

                        if (ar[after.x][after.y - 1] != '\*')

                            after.y = after.y - 1;

                    }

                    if (after.sta == 2)

                    {

                        if (ar[after.x - 1][after.y] != '\*')

                        {

                            after.x = after.x - 1;

                            after.time++;

                        }

                    }

                    if (after.sta == 3)

                    {

                        if (ar[after.x + 1][after.y] != '\*')

                        {

                            after.x = after.x + 1;

                            after.time++;

                        }

                    }

                }

                if (now.time % 2 != 0)

                {

                    if (after.sta == 0)

                        if (ar[after.x][after.y + 1] != '\*')

                        {

                            after.y = after.y + 1;

                            after.time++;

                        }

                    if (after.sta == 1)

                        if (ar[after.x][after.y - 1] != '\*')

                        {

                            after.y = after.y - 1;

                            after.time++;

                        }

                    if (after.sta == 2)

                    {

                        if (ar[after.x - 1][after.y] != '\*')

                            after.x = after.x - 1;

                    }

                    if (after.sta == 3)

                    {

                        if (ar[after.x + 1][after.y] != '\*')

                            after.x = after.x + 1;

                    }

                }

            }

            if ((ar[after.x][after.y] == '.' || ar[after.x][after.y] == 'T') && after.time < visit[after.x][after.y])

            {

                q.push(after);

                visit[after.x][after.y] = after.time;

            }

        }

    }

    return -1;

}

1181  搜索第4题，dfs；

#include <cstdio>

#include <cstring>

#include <cstdlib>

using namespace std;

struct node

{

    char begin;

    char end;

    char word;

};

struct node ar[10000];

bool visit[10000];

int k;

bool doudou = false;

bool dfs(char ch);

int main()

{

    int i, j, u, n, m;

    char sa[100];

    while (scanf("%s", sa) != EOF)

    {

        k = 1;

        while (strcmp(sa, "0") != 0)

        {

            ar[k].begin = sa[0];

            ar[k].end = sa[strlen(sa) - 1];

            k++;

            scanf("%s", sa);

        }

        doudou = false;

        for (i = 1; i < k; i++)

        {

            if (ar[i].begin == 'b')

            {

                memset(visit, false, sizeof(visit));

                visit[i] = true;

                if (dfs(ar[i].end) == true)

                {

                    doudou = true;

                    break;

                }

            }

        }

        if (doudou)

            printf("Yes.\n");

        else

            printf("No.\n");

    }

    return 0;

}

bool dfs(char ch)

{

    int i, j;

    if (ch == 'm')

        return true;

    for (i = 1; i < k; i++)

    {

        if (ar[i].begin == ch && !visit[i])

        {

            visit[i] = true;

            if (dfs(ar[i].end) == true)

            {

                return true;

            }

            visit[i] = false;

        }

    }

    return false;

}

1253    搜索第6题 bfs   三维的；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <queue>

using namespace std;

struct node

{

    int x;

    int y;

    int z;

    int time;

};

struct node now, after;

int a, b, c, t;

int dir[6][3] = { { 0, 0, 1 },{ 0, 0, -1 },{ 0, -1, 0 },{ 0, 1, 0 },{ 1, 0, 0 },{ -1, 0, 0 } };

int visit[55][55][55], ar[55][55][55];

void bfs();

int main()

{

    int i, j, k, u, n, m;

    while (scanf("%d", &n) != EOF)

    {

        for (m = 0; m < n; m++)

        {

            scanf("%d %d %d %d", &a, &b, &c, &t);

            for (u = 1; u <= a; u++)

            {

                for (i = 1; i <= b; i++)

                {

                    for (j = 1; j <= c; j++)

                    {

                        scanf("%d", &ar[u][i][j]);

                        visit[u][i][j] = 10000000;

                    }

                }

            }

            bfs();

            if (visit[a][b][c] != 10000000 && visit[a][b][c] <= t)

                printf("%d\n", visit[a][b][c]);

            else

                printf("-1\n");

        }

    }

    return 0;

}

void bfs()

{

    int i, j, k, u;

    queue<node>q;

    now.x = 1;

    now.y = 1;

    now.z = 1;

    now.time = 0;

    q.push(now);

    while (!q.empty())

    {

        now = q.front();

        q.pop();

        for (i = 0; i < 6; i++)

        {

            after.x = now.x + dir[i][0];

            after.y = now.y + dir[i][1];

            after.z = now.z + dir[i][2];

            after.time = now.time + 1;

            if (after.x < 1 || after.y < 1 || after.z < 1 || after.x > b || after.y > c || after.z > a)

                continue;

            if (ar[after.z][after.x][after.y] == 1)

                continue;

            if (after.time > t || (after.time == t && (after.x != b || after.y != c || after.z != a)))

                continue;

            if (ar[after.z][after.x][after.y] == 0 && after.time < visit[after.z][after.x][after.y])

            {

                q.push(after);

                visit[after.z][after.x][after.y] = after.time;

            }

        }

    }

    return;

}

1010   dfs  终于他妈的水过去了，奇偶性剪枝；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <cmath>

const int MAX = 100000000;

using namespace std;

int dir[4][2] = { 0, -1, 0, 1, -1, 0, 1, 0 };

struct Point

{

    int x, y;

    Point() {}

    Point(int a, int b) :x(a), y(b) {}

    Point go(int direction\_id) { return Point(x + dir[direction\_id][0], y + dir[direction\_id][1]); }

};

struct Point start, nextp;

bool visit[10][10];

bool doudou;    //标记；

char ar[10][10];

int T, a, b, x1, Y1, x2, y2;

void dfs(int x, int y, int time);

int main()

{

    int i, j, k, u, n, m;

    while (scanf("%d %d %d", &a, &b, &T) != EOF && (a || b || T))

    {

        x1 = Y1 = x2 = y2 = MAX;

        for (i = 1; i <= a; i++)

        {

            scanf("%s", ar[i] + 1);

            if (x1 != MAX && x2 != MAX && Y1 != MAX && y2 != MAX)

                continue;

            for (j = 1; j <= b; j++)

            {

                if (ar[i][j] == 'S')

                {

                    x1 = i;

                    Y1 = j;

                }

                if (ar[i][j] == 'D')

                {

                    x2 = i;

                    y2 = j;

                }

            }

        }

        if (T < abs(x1 - x2) + abs(Y1 - y2) || (x1 + Y1 + x2 + y2 + T) % 2 != 0)

        {

            printf("NO\n");

            continue;

        }

        doudou = false;

        memset(visit, false, sizeof(visit));

        visit[x1][Y1] = true;

        dfs(x1, Y1, 0);

        if (doudou)

        {

            printf("YES\n");

        }

        else

            printf("NO\n");

    }

    return 0;

}

void dfs(int x, int y, int time)

{

    int i, temp;

    Point start(x, y);

    if (doudou)

        return;

    if (start.x == x2 && start.y == y2 && time == T)

    {

        doudou = true;

        return;

    }

    if (start.x == x2 && start.y == y2 && time != T)

        return;

    if (start.x < 1 || start.y < 1 || start.x > a || start.y > b || ar[start.x][start.y] == 'X')

        return;

    if ((start.x != x2 || start.y != y2) && time == T)

        return;

    if (visit[start.x][start.y] && time != 0)

        return;

    temp = T - time - abs(start.x - x2) - abs(start.y - y2);

    if (temp < 0 || temp % 2 != 0)

        return;

    for (i = 0; i < 4; i++)

    {

        nextp = start.go(i);

        visit[start.x][start.y] = true;

        dfs(nextp.x, nextp.y, time + 1);

        visit[start.x][start.y] = false;

        if (doudou)

            return;

    }

}

1150 二分匹配第1题， 匈牙利算法dfs；

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <cstdlib>

const int MAX = 100000000;

using namespace std;

int visit[110];

int ar[110][110];

int link[110];

int leftN, rightN, res;

bool dfs(int f);

int main()

{

    int i, j, k, u, n, m, num;

    while (scanf("%d", &leftN) != EOF && leftN)

    {

        scanf("%d %d", &rightN, &num);

        memset(ar, 0, sizeof(ar));

        memset(link, -1, sizeof(link));

        res = 0;

        for (i = 0; i < num; i++)

        {

            scanf("%d %d %d", &k, &n, &m);

            ar[n][m] = 1;

        }

        for (i = 1; i < leftN; i++)

        {

            memset(visit, false, sizeof(visit));

            if (dfs(i))

                res++;

        }

        printf("%d\n", res);

    }

    return 0;

}

bool dfs(int f)

{

    int i, j, k, u;

    for (i = 1; i <= rightN; i++)

    {

        if (!visit[i] && ar[f][i])

        {

            visit[i] = true;

            if (link[i] == -1 || dfs(link[i]))

            {

                link[i] = f;

                return true;

            }

        }

    }

    return false;

}

1045   一道深搜dfs 套 DP；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

using namespace std;

struct point

{

    int x;

    int y;

};

struct point ar[20];

char sa[10][10];

int n, res, cnt;

int dir[4][2] = { 1, 0, -1, 0, 0, 1, 0, -1 };

void dfs(int a, int num);

bool check(int c);

int main()

{

    int i, j, k, u, a, b;

    while (scanf("%d", &n) != EOF && n)

    {

        cnt = 0;

        for (i = 1; i <= n; i++)

        {

            scanf("%s", sa[i] + 1);

            for (j = 1; j <= n; j++)

            {

                if (sa[i][j] == '.')

                {

                    ar[cnt].x = i;

                    ar[cnt++].y = j;

                }

            }

        }

        res = 0;

        dfs(0, 0);

        printf("%d\n", res);

    }

    return 0;

}

void dfs(int a, int num)

{

    if (num > res)

        res = num;

    if (a == cnt)

        return;

    dfs(a + 1, num);

    if (check(a))

    {

        sa[ar[a].x][ar[a].y] = '#';

        dfs(a + 1, num + 1);

        sa[ar[a].x][ar[a].y] = '.';

    }

}

bool check(int c)

{

    int i, doudou = 0;

    struct point now;

    for (i = 0; i < 4; i++)

    {

        if (doudou)

            return false;

        now.x = ar[c].x + dir[i][0];

        now.y = ar[c].y + dir[i][1];

        while (now.x <= n && now.y <= n && now.x >= 1 && now.y >= 1)

        {

            if (sa[now.x][now.y] == '#')

            {

                doudou = 1;

                break;

            }

            if (sa[now.x][now.y] == 'X')

            {

                break;

            }

            now.x += dir[i][0];

            now.y += dir[i][1];

        }

    }

    if (doudou)

        return false;

    return true;

}

1241  dfs(连成块问题)；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

using namespace std;

const int MAX = 10000000;

int visit[110][110];

char ar[110][110];

int cnt, a, b, N;

int dir[8][2] = { -1, -1, -1, 0, -1, 1, 0, -1, 0, 1, 1, -1, 1, 0, 1, 1 };

bool dfs(int x, int y);

int main()

{

    int i, j, k, u, n, m;

    while (scanf("%d %d", &a, &b) != EOF && (a || b))

    {

        for (i = 1; i <= a; i++)

        {

            scanf("%s", ar[i] + 1);

        }

        memset(visit, 0, sizeof(visit));

        cnt = 0;

        N = 1;

        for (i = 1; i <= a; i++)

        {

            for (j = 1; j <= b; j++)

            {

                if (ar[i][j] == '@' && !visit[i][j])

                {

                    dfs(i, j);

                    cnt++;

                }

                N++;

            }

        }

        printf("%d\n", cnt);

    }

    return 0;

}

bool dfs(int x, int y)

{

    int i, j, k, u, doudou = 0;

    int nowx, nowy;

    for (i = 0; i < 8; i++)

    {

        nowx = x + dir[i][0];

        nowy = y + dir[i][1];

        if (nowx > a || nowy > b || nowx < 1 || nowy < 1 || ar[nowx][nowy] == '\*')

            continue;

        if (visit[nowx][nowy])

            continue;

        visit[nowx][nowy] = N;

        doudou = 1;

        dfs(nowx, nowy);

    }

    if (doudou)

        return true;

    else

        return false;

}

1254    搜索第9题  bfs+bfs 可以走回头路只需visit[i][j][u]三维  多一个方向就行了；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <queue>

#include <algorithm>

using namespace std;

const int MAX = 100000000;

const int MIN = -10000000;

int visit[10][10][4];

int vis[10][10][4];

int ar[10][10];

int dir[4][2] = { -1, 0, 1, 0, 0, 1, 0, -1 };

int xb, yb, a, b, xr, yr;

struct point

{

    int x;

    int y;

    int step;

    int renx;

    int reny;

    friend bool operator < (point a, point b)

    {

        return a.step > b.step;

    }

};

point now, after;

struct node

{

    int x;

    int y;

    int step;

    friend bool operator < (node t, node s)

    {

        return t.step > s.step;

    }

};

bool r\_bfs();

void x\_bfs();

int main()

{

    int i, j, k, u, n, m;

    while (scanf("%d", &n) != EOF)

    {

        for (m = 1; m <= n; m++)

        {

            scanf("%d %d", &a, &b);

            for (i = 1; i <= a; i++)

            {

                for (j = 1; j <= b; j++)

                {

                    scanf("%d", &ar[i][j]);

                    if (ar[i][j] == 2)

                    {

                        xb = i;

                        yb = j;

                    }

                    if (ar[i][j] == 4)

                    {

                        xr = i;

                        yr = j;

                    }

                }

            }

            x\_bfs();

            if (ar[now.x][now.y] == 3)

                printf("%d\n", now.step);

            else

                printf("-1\n");

        }

    }

    return 0;

}

void x\_bfs()

{

    int i, j, k, u;

    priority\_queue<point>q;

    for (i = 0; i <= a + 1; i++)

    {

        for (j = 0; j <= b + 1; j++)

        {

            for (u = 0; u < 4; u++)

            {

                visit[i][j][u] = MAX;

            }

        }

    }

    for (u = 0; u < 4; u++)

    {

        now.x = xb;

        now.y = yb;

        now.step = 0;

        now.renx = xr;

        now.reny = yr;

        visit[xb][yb][u] = 0;

        q.push(now);

        while (!q.empty())

        {

            now = q.top();

            q.pop();

            if (ar[now.x][now.y] == 3)

                return;

            for (i = 0; i < 4; i++)

            {

                after.x = now.x + dir[i][0];

                after.y = now.y + dir[i][1];

                after.step = now.step + 1;

                after.renx = now.x - dir[i][0];

                after.reny = now.y - dir[i][1];

                if (after.x > a || after.y > b || after.x < 1 || after.y < 1 || ar[after.x][after.y] == 1)

                    continue;

                if (after.step <= visit[after.x][after.y][i] && r\_bfs())

                {

                    after.renx = now.x;

                    after.reny = now.y;

                    q.push(after);

                    visit[after.x][after.y][i] = after.step;

                }

            }

        }

    }

    now.step = MAX;

    return;

}

bool r\_bfs()

{

    int i, j, k, u;

    node qq, hh;

    priority\_queue<node>p;

    memset(vis, 0, sizeof(vis));

    qq.x = now.renx;

    qq.y = now.reny;

    qq.step = 0;

    for (u = 0; u < 4; u++)

    {

        vis[qq.x][qq.y][u] = 1;

        p.push(qq);

        while (!p.empty())

        {

            qq = p.top();

            p.pop();

            if (qq.x == after.renx && qq.y == after.reny)

                return true;

            for (i = 0; i < 4; i++)

            {

                hh.x = qq.x + dir[i][0];

                hh.y = qq.y + dir[i][1];

                hh.step = qq.step + 1;

                if (hh.x > a || hh.y > b || hh.x < 1 || hh.y < 1 || ar[hh.x][hh.y] == 1 || (hh.x == now.x && hh.y == now.y))

                    continue;

                if (vis[hh.x][hh.y][i])

                    continue;

                p.push(hh);

                vis[hh.x][hh.y][i] = 1;

            }

        }

    }

    return false;

}

//朴素欧几里得 非递归

int gcd(int x,int y)

{

    int t;

    while(y)

    {

        t=x%y;

        x=y;

        y=t;

    }

    return  x;

}

//朴素欧几里得 递归

int gcd(int a,int b)

{

    return b ? gcd(b, a%b):a;

}

//扩展欧几里得

int exGcd(int a, int b, int &x, int &y)

{

    int res, temp;

    if(b==0)

    {

        x=1;

        y=0;

        return a;

    }

    res = exGcd(b, a%b, x, y);

    temp = x;

    x = y;

    y = temp-a/b\*y;

    return res;

}

/\*

int gcd(int a, int b, int &x, int &y)

{

    if (b==0)

    {

        x=1；

        y=0;

        return a;

    }

    int q=gcd(b, a%b, y, x);

    y -= a/b\*x;

    return q;

}

\*/

//a[] 是余数， m[]是除数，M是总乘数，Mi是去掉m[i]的乘数， x, y是扩展欧几里得的系数，ans是结果；

//中国剩余定理 互质模板；

#include<iostream>

#include<cstdio>

#include<cstring>

typedef long long LL;

const int N = 1000;

using namespace std;

LL M ;

void extend\_Euclid(LL a, LL b, LL &x, LL &y)

{

    if (b == 0)

    {

        x = 1;

        y = 0;

        return;

    }

    extend\_Euclid(b, a % b, x, y);

    LL tmp = x;

    x = y;

    y = tmp - (a / b) \* y;

}

LL CRT(LL a[],LL m[],LL n)

{

    LL ans = 0;

    for(int i=1; i<=n; i++)

        M \*= m[i];

    for(int i=1; i<=n; i++)

    {

        LL x, y;

        LL Mi = M / m[i];

        extend\_Euclid(Mi, m[i], x, y);

        ans = (ans + Mi \* x \* a[i]) % M;

    }

    if(ans < 0) ans += M;

    return ans;

}

//中国剩余定理 非互质模板；

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

typedef long long int LL;

const int N = 15;

LL M;

LL gcd(LL a, LL b)

{

    if (b == 0)

        return a;

    return gcd(b, a%b);

}

LL Extend\_Euclid(LL a, LL b, LL&x, LL& y)

{

    if (b == 0)

    {

        x = 1, y = 0;

        return a;

    }

    LL d = Extend\_Euclid(b, a%b, x, y);

    LL t = x;

    x = y;

    y = t - a / b\*y;

    return d;

}

//a在模n乘法下的逆元，没有则返回-1

LL inv(LL a, LL n)

{

    LL x, y;

    LL t = Extend\_Euclid(a, n, x, y);

    if (t != 1)

        return -1;

    return (x%n + n) % n;

}

//将两个方程合并为一个

bool merge(LL a1, LL n1, LL a2, LL n2, LL& a3, LL& n3)

{

    LL d = gcd(n1, n2);

    LL c = a2 - a1;

    if (c%d)

        return false;

    c = (c%n2 + n2) % n2;

    c /= d;

    n1 /= d;

    n2 /= d;

    c \*= inv(n1, n2);

    c %= n2;

    c \*= n1\*d;

    c += a1;

    n3 = n1\*n2\*d;

    a3 = (c%n3 + n3) % n3;

    return true;

}

//求模线性方程组x=ai(mod ni),ni可以不互质

LL China\_Reminder2(int len, LL\* a, LL\* n)

{

    LL a1 = a[0], n1 = n[0];

    LL a2, n2;

    for (int i = 1; i < len; i++)

    {

        LL aa, nn;

        a2 = a[i], n2 = n[i];

        if (!merge(a1, n1, a2, n2, aa, nn))

            return -1;

        a1 = aa;

        n1 = nn;

    }

    M = n1;

    return (a1%n1 + n1) % n1;

}

int main()

{

    LL a[N], m[N], res;

}

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

    int number=123456;

    char string[25];

    itoa(number,string,10);

    printf("integer=%d string=%s\n",number,string);

    return 0;

}

//char\*itoa(int value,char\*string,int radix);

//atoi

//int atoi(const char \*nptr);

/\*atoi( ) 函数会扫描参数 nptr字符串，跳过前面的空白字符（例如空格，tab缩进等，可以通过isspace( )函数来检测），

直到遇上数字或正负符号才开始做转换，而再遇到非数字或字符串结束时('\0')才结束转换，并将结果返回。

如果 nptr不能转换成 int 或者 nptr为空字符串，那么将返回 0  \*/

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

    int n;

    char \*str = "12345.67";

    n = atoi(str);

    printf("n=%d\n",n);

    return 0;

}

#include <string.h>

#include <stdlib.h>

//原型 void memmove(void \*s1, const void \*s2, size\_t n);

int \*p1, \*p2;

memmove(p1, p2, n\*sizeof(int));

//从p2指向的位置，拷贝n字节，到p1指向的位置； p1指向的那个值也被覆盖；

#include <stdlib.h>

int comp(const void \* p1, const void \* p2);

qsort(ar, num, sizeof(...), comp);

int comp(const void \* p1, const void \* p2)

{

    return \*(... \*)p1 - \*(... \*)p2;

}

//对一个二维数组进行排序：int a[1000][2]; 其中按照a[0]的大小进行一个整体的排序，其中a[1]必须和a[0]一起移动交换。

//即第一行和第二行（a[0]和a[1]分别代表第一行和第二行的首地址）。

qsort(a, 1000, sizeof(int)\*2, comp);

int comp(const void \* a,const void \* b)

{

    return ((int\*)a)[0] - ((int\*)b)[0];

}

//对字符串进行排序

int comp(const void \* p1,const void \* p2)

{

    return strcmp((char\*)p2, (char\*)p1);

}

int main()

{

    char a[MAX1][MAX2];

    initial(a);

    qsort(a, lenth, sizeof(a[0]), comp);

}

//lenth为数组a的长度

//按结构体中某个关键字排序（对结构体一级排序）：

struct Node

{

    double data;

    int other;

}s[100];

qsort(s, 100, sizeof(s[0]), comp);

int comp(const void \* p1,const void \* p2)

{

    return (Node\*)p2->data > (Node\*)p1->data ? 1:-1;

}

//按结构体中多个关键字排序（对结构体多级排序）[以二级为例]：

struct Node

{

    int x;

    int y;

}s[100];

//按照x从小到大排序，当x相等时按y从大到小排序

qsort(s, 100, sizeof(s[0]), comp);

int Comp(const void\*p1,const void\*p2)

{

    struct Node \* c = (Node\*)p1;

    struct Node \* d = (Node\*)p2;

    if(c->x != d->x)

        return c->x - d->x;

    else

        return d->y - c->y;

}

//对结构体中字符串进行排序：

struct Node

{

    int data;

    char str[100];

}s[100];

//按照结构体中字符串str的字典序排序

qsort(s, 100, sizeof(s[0]), Comp);

int Comp(const void \* p1,const void \* p2)

{

    return strcmp((Node\*)p1->str, (Node\*)p2->str);

}

#include <string.h>

char \* s\_gets(char \* st, int n);

char \* s\_gets(char \* st, int n)

{

    char \* ret\_val;

    char \* find;

    ret\_val = fgets(st, n, stdin)

    if(ret\_val)

    {

        find = strchr(st, '\n');

        if(find)

            \*find = '\0';

        else

            while (getchar() != '\n')

                continue;

    }

    return ret\_val;

}

字符串数字倒置

m = cnt;

while (n)

{

    sa[cnt++] = n % 10 + '0';

    n /= 10;

}

for (j = 0; j < (cnt - m) / 2; j++)

{

    t = sa[m + j];

    sa[m + j] = sa[cnt - j - 1];

    sa[cnt - j - 1] = t;

}