# Ccccccccccc模板

**初级搜索：**

1010 dfs；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cctype>

#include <cmath>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

struct node

{

int x;

int y;

int time;

};

node now, after;

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

int visit[10][10];

char ar[10][10];

int beginx, beginy, N, M, T, endx, endy;

bool dfs(int b, int e, int t);

int main()

{

int i, j, k, u, n, m;

while (scanf("%d %d %d", &N, &M, &T) != EOF && (N || M || T))

{

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

{

scanf("%s", ar[i] + 1);

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

{

if (ar[i][j] == 'S')

{

beginx = i;

beginy = j;

}

if (ar[i][j] == 'D')

{

endx = i;

endy = j;

}

}

}

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

if (dfs(beginx, beginy, 0))

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

bool dfs(int b, int e, int t)

{

int i, j, u, v, temp;

if (ar[b][e] == 'D' && t == T)

return true;

visit[b][e] = 1;

for (i = 0; i < 4; i++)

{

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

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

after.time = t + 1;

if (after.x < 1 || after.x > N || after.y < 1 || after.y > M || visit[after.x][after.y] || ar[after.x][after.y] == 'X')

continue;

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

continue;

temp = (T - after.time) - abs(after.x - endx) - abs(after.y - endy);

if (temp % 2 != 0 || temp < 0)

continue;

if (dfs(after.x, after.y, after.time))

return true;

}

visit[b][e] = 0;

return false;

}

1429 bfs + 状态压缩;

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <algorithm>

using namespace std;

const int MAX = 100000000;

struct node

{

int x;

int y;

int time;

int key;

friend bool operator < (node a, node b)

{

return a.time > b.time;

}

};

node now, after;

int visit[25][25][1050], keys[15];

char ar[25][25];

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

int a, b, T, xb, yb;

void bfs();

int main()

{

int i, j, k, u, n, m, doudou = 0;

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

{

xb = yb = MAX;

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

{

scanf("%s", ar[i] + 1);

if (xb == MAX && yb == MAX)

{

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

{

if (ar[i][j] == '@')

{

xb = i;

yb = j;

}

}

}

}

for (i = 0; i <= 11; i++)

keys[i] = 1 << i;

bfs();

//if (doudou)

// putchar('\n');

//doudou = 1;

if (now.time < T && ar[now.x][now.y] == '^')

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

else

printf("-1\n");

}

return 0;

}

void bfs()

{

int i, j, k, u;

priority\_queue<node>q;

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

now.x = xb;

now.y = yb;

now.time = 0;

now.key = 0;

visit[xb][yb][0] = 1;

q.push(now);

while (!q.empty())

{

now = q.top();

q.pop();

if (ar[now.x][now.y] == '^' && now.time < T)

return;

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;

after.key = now.key;

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

continue;

if (after.time == T && ar[after.x][after.y] != '^')

continue;

if (ar[after.x][after.y] == '^' && after.time >= T)

continue;

if (ar[after.x][after.y] >= 'a' && ar[after.x][after.y] <= 'z')

{

k = ar[after.x][after.y] - 'a';

if (!visit[after.x][after.y][after.key | keys[k]])

{

after.key = after.key | keys[k];

q.push(after);

visit[after.x][after.y][after.key] = 1;

}

}

if (ar[after.x][after.y] >= 'A' && ar[after.x][after.y] <= 'Z')

{

k = ar[after.x][after.y] - 'A';

if (!visit[after.x][after.y][after.key] && after.key & keys[k])

{

q.push(after);

visit[after.x][after.y][after.key] = 1;

}

}

if ((ar[after.x][after.y] == '.' || ar[after.x][after.y] == '^' || ar[after.x][after.y] == '@') && !visit[after.x][after.y][after.key])

{

q.push(after);

visit[after.x][after.y][after.key] = 1;

}

}

}

now.time = MAX;

}

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;

}

**图论：**

江苏省赛8；最短路模板；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <map>

#include <queue>

#include <vector>

using namespace std;

const int MAX = 100000000;

struct node

{

int to;

int next;

int val;

bool operator < (const node& p) const

{

return val > p.val;

}

};

node edge[200000];

map<string, int>q;

int head[200000], dist[200000];

int cnt, N, M, Count;

void addedge(int x, int y, int z);

void dijkstra(int begin);

int main()

{

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

char s[50], ss[50][50], begin[50], end[50];

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

{

for (m = 1; m <= n; m++)

{

q.clear();

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

Count = cnt = 0;

scanf("%d", &N);

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

{

scanf("%s", s);

if (strcmp(s, "B") == 0)

{

scanf("%d", &k);

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

{

scanf("%s", ss[j]);

if (q[ss[j]] == 0)

{

q[ss[j]] = Count++;

}

}

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

{

for (u = j + 1; u <= k; u++)

{

addedge(q[ss[j]], q[ss[u]], 0);

addedge(q[ss[u]], q[ss[j]], 0);

}

}

}

else

{

scanf("%d", &k);

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

{

scanf("%s", ss[j]);

if (q[ss[j]] == 0)

{

q[ss[j]] = Count++;

}

}

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

{

for (u = j + 1; u <= k; u++)

{

addedge(q[ss[j]], q[ss[u]], 2);

addedge(q[ss[u]], q[ss[j]], 2);

}

}

}

}

scanf("%s %s", begin, end);

dijkstra(q[begin]);

printf("Case #%d: %d\n", m, dist[q[end]]);

}

}

return 0;

}

void addedge(int x, int y, int z)

{

edge[cnt].to = y;

edge[cnt].val = z;

edge[cnt].next = head[x];

head[x] = cnt++;

}

void dijkstra(int begin)

{

priority\_queue<node>p;

int i, j, k;

node now, after;

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

{

dist[i] = MAX;

}

now.to = begin;

now.val = 0;

now.next = head[begin];

dist[begin] = 0;

p.push(now);

while (!p.empty())

{

now = p.top();

p.pop();

for (i = head[now.to]; i != -1; i = edge[i].next)

{

after = edge[i];

if (dist[after.to] > dist[now.to] + after.val)

{

dist[after.to] = dist[now.to] + after.val;

p.push(after);

}

}

}

}

最短路模板 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);

}

}

}

}

}

差分约束

spfa是判断是否成环，如果求最短路就是判断的负环（因为该环中的点在无限的更新最短距离），如果求最长路就是判断正环（因为该环中的点在无限的更新最长距离）；

至于为什么一定要入队次数大于顶点数，因为bfs在第一次入到环时并不认识她，为了防止是所有顶点都在环中（反正遇到了正/负环会一直更新下去肯定会到顶点数的），所以必须要入队次数大于顶点数（这就是为什么bfs判环比dfs慢，dfs判断的是一个点是否在一条路径中出现超过1次，所以在第一次遇到环时就能认识她）

1531 差分约束第6题 SPFA()判环，因为是把所有条件转化为求最短路 所以判负环（如果把所有条件转化为求最长路，就是判正环）；

最短路版

差分系统<=;

算出解的最大值；

差分约束系统的解法利用到了单源最短路径问题中的三角形不等式。即对于任何一条边u -> v，都有：

d(v) <= d(u) + w(u, v)

其中d(u)和d(v)是从源点分别到点u和点v的最短路径的权值，w(u, v)是边u->v的权值。

显然以上不等式就是d(v) - d(u) <= w(u, v)

另外还要有一个总源点，到所有点的距离为0， 防止图不连通；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <cctype>

#include <queue>

#include <deque>

#include <vector>

#include <map>

using namespace std;

const int MAX = 100000000;

struct node

{

int to;

int next;

int val;

};

node edge[500];

int head[500], dist[500], visit[500], in[500];

int cnt, n;

void addedge(int x, int y, int z);

bool SPFA(int s);

int main()

{

int i, j, k, m, a, b, c, u, v;

char ch, s[5];

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

{

scanf("%d", &m);

cnt = 0;

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

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

{

scanf("%d %d %s %d", &u, &v, s, &k);

if (s[0] == 'g')

{

addedge(u + v, u - 1, -k - 1);

}

else

{

addedge(u - 1, u + v, k - 1);

}

}

int st = n + 10;

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

addedge(st, i, 0);

if (SPFA(st))

{

printf("lamentable kingdom\n");

}

else

{

printf("successful conspiracy\n");

}

}

return 0;

}

void addedge(int x, int y, int z)

{

edge[cnt].to = y;

edge[cnt].val = z;

edge[cnt].next = head[x];

head[x] = cnt++;

}

bool SPFA(int s)

{

int i, j, k, u, v;

deque<int>q;

for (i = 0; i <= n + 1; i++)

dist[i] = MAX;

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

memset(in, 0, sizeof(in));

dist[s] = 0;

visit[s] = 1;

q.push\_back(s);

while (!q.empty())

{

u = q.front();

q.pop\_front();

visit[u] = 0;

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

{

v = edge[i].to;

if (dist[v] > dist[u] + edge[i].val)

{

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

if (!visit[v])

{

visit[v] = 1;

if (++in[v] > n + 1)

return 0;

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

q.push\_front(v);

else

q.push\_back(v);

}

}

}

}

return 1;

}

最长路版

差分系统>=；

算出解的最小值；

总源点到所有点的距离仍为0，dist初始化为最小值，if(dist[v] <dist[u] + edge[i].val) dist[v] = dist[u] + edge[i].val

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <cctype>

#include <queue>

#include <deque>

#include <vector>

#include <map>

using namespace std;

const int MAX = 100000000;

struct node

{

int to;

int next;

int val;

};

node edge[500];

int head[500], dist[500], visit[500], in[500];

int cnt, n;

void addedge(int x, int y, int z);

bool SPFA(int s);

int main()

{

int i, j, k, m, a, b, c, u, v;

char ch, s[5];

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

{

scanf("%d", &m);

cnt = 0;

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

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

{

scanf("%d %d %s %d", &u, &v, s, &k);

if (s[0] == 'g')

{

addedge(u -1, u +v, k+1);

}

else

{

addedge(u+v, u-1, -k + 1);

}

}

int st = n+10;

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

addedge(st, i, 0);

if (SPFA(st))

{

printf("lamentable kingdom\n");

}

else

{

printf("successful conspiracy\n");

}

}

return 0;

}

void addedge(int x, int y, int z)

{

edge[cnt].to = y;

edge[cnt].val = z;

edge[cnt].next = head[x];

head[x] = cnt++;

}

bool SPFA(int s)

{

int i, j, k, u, v;

deque<int>q;

for (i = 0; i <= n + 1; i++)

dist[i] = -MAX;

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

memset(in, 0, sizeof(in));

dist[s] = 0;

visit[s] = 1;

q.push\_back(s);

while (!q.empty())

{

u = q.front();

q.pop\_front();

visit[u] = 0;

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

{

v = edge[i].to;

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

{

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

if (!visit[v])

{

visit[v] = 1;

if (++in[v] > n + 1)

return 0;

if (!q.empty() && dist[v] > dist[q.front()])

q.push\_front(v);

else

q.push\_back(v);

}

}

}

}

return 1;

}

强连通题 注意出度和入度的使用；

3639 强连通第3题 缩点+反向建图+dfs；

#include <cstdio>

#include <cstring>

#include <cstdlib>

using namespace std;

int Stack[5010], Instack[5010];

int dfn[5010], low[5010];

int head[5010], hd[5010];

int belong[5010], num[5010], res[5010], visit[5010], in[5010];

int Index, top, cnt, Sccc, tot;

int ans;

struct node

{

int from;

int next;

int to;

};

node edge[60010], e[60010];

void add(int x, int y);

void ad(int x, int y);

void Tarjan(int f);

int dfs(int n);

void solve(int n, int m, int cas);

int main()

{

int i, j, k, n, m, a, b, c, p, q, N, M;

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

{

for (m = 1; m <= n; m++)

{

top = Index = cnt = Sccc = ans = tot = 0;

memset(dfn, 0, sizeof(dfn));

memset(Instack, 0, sizeof(Instack));

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

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

memset(num, 0, sizeof(num));

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

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

memset(in, 0, sizeof(in));

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

scanf("%d %d", &N, &M);

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

{

scanf("%d %d", &p, &q);

add(p, q);

}

solve(N, M, m);

}

}

return 0;

}

void add(int x, int y)

{

edge[cnt].from = x;

edge[cnt].to = y;

edge[cnt].next = head[x];

head[x] = cnt++;

}

void ad(int x, int y)

{

e[tot].to = y;

e[tot].from = x;

e[tot].next = hd[x];

hd[x] = tot++;

}

void Tarjan(int f)

{

int i, j, k, temp;

low[f] = dfn[f] = ++Index;

Stack[top++] = f;

Instack[f] = 1;

for (i = head[f]; i != -1; i = edge[i].next)

{

temp = edge[i].to;

if (!dfn[temp])

{

Tarjan(temp);

low[f] = (low[f] > low[temp]) ? low[temp] : low[f];

}

else if (Instack[temp] && low[f] > dfn[temp])

low[f] = dfn[temp];

}

if (low[f] == dfn[f])

{

Sccc++;

do

{

temp = Stack[--top];

Instack[temp] = 0;

num[Sccc]++;

belong[temp] = Sccc;

} while (temp != f);

}

}

int dfs(int n)

{

int i, j, k, t, sum = num[n];

visit[n] = 1;

for (i = hd[n]; i != -1; i = e[i].next)

if (!visit[e[i].to])

sum += dfs(e[i].to);

return sum;

}

void solve(int n, int m, int cas)

{

int i, j, k, s, t;

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

{

if (!dfn[i])

Tarjan(i);

}

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

{

for (j = head[i]; j != -1; j = edge[j].next)

{

if (belong[i] != belong[edge[j].to])

{

ad(belong[edge[j].to], belong[i]);

in[belong[i]]++;

}

}

}

ans = -1;

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

{

if (!in[i])

{

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

res[i] = dfs(i);

if (ans < res[i])

ans = res[i];

}

}

printf("Case %d: %d\n", cas, ans-1);

bool doudou = false;

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

if (res[belong[i]] == ans)

{

if (doudou)

putchar(' ');

doudou = true;

printf("%d", i);

}

putchar('\n');

}

最大流EK算法；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <algorithm>

using namespace std;

const int MAX = 100000000;

const int MIN = -100000000;

int ar[20][20];

int before[20], flow[20];

int n, m, Begin, End;

int bfs();

int EK();

int main()

{

int i, j, k, u, a, b, c, h, cas = 1;

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

{

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

{

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

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

Begin = 1;

End = n;

for (h = 1; h <= m; h++)

{

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

ar[a][b] += c;

}

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

}

}

return 0;

}

int EK()

{

int i, j, k, now, pre, maxflow = 0, temp;

while ((temp = bfs()) != -1)

{

now = End;

maxflow += temp;

while (now != Begin)

{

pre = before[now];

ar[pre][now] -= temp;

ar[now][pre] += temp;

now = pre;

}

}

return maxflow;

}

int bfs()

{

int i, j, k, t;

queue<int>q;

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

before[Begin] = 0;

flow[Begin] = MAX;

q.push(Begin);

while (!q.empty())

{

t = q.front();

q.pop();

if (t == End)

break;

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

{

if (i != Begin && before[i] == -1 && ar[t][i])

{

before[i] = t;

flow[i] = (flow[t] < ar[t][i]) ? flow[t] : ar[t][i];

q.push(i);

}

}

}

return (before[End] == -1) ? -1 : flow[n];

}

dinic算法

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <algorithm>

using namespace std;

const int MAX = 100000000;

int ar[505][505];

int Max;

int dinic();

int dfs(int st, int ed, int fl);

int bfs(int st, int ed);

int dep[600];

int main()

{

int i, j, k, u, n, m, a, b, c, p, q, r, cas = 1, sum;

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

{

for (m = 1; m <= n; m++)

{

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

Max = 0;

sum = 0;

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

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

{

scanf("%d %d %d", &r, &p, &q);

sum += r;

Max = (Max < q) ? q : Max;

ar[0][i] = r;

for (j = p; j <= q; j++)

{

ar[i][j] = 1;

}

}

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

{

ar[i][Max + 1] = b;

}

if ((k = dinic()) == sum)

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

else

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

}

}

return 0;

}

int dinic()

{

int i, j, res = 0, temp;

while (bfs(0, Max + 1))

{

while (1)

{

temp = dfs(0, Max + 1, MAX);

if (temp == 0)

break;

res += temp;

}

}

return res;

}

int dfs(int st, int ed, int fl)

{

int i, j, k, temp, flow;

if (st == ed)

return fl;

for (i = 0; i <= Max + 1; i++)

{

if (ar[st][i] && dep[i] == dep[st] + 1)

{

flow = (fl < ar[st][i]) ? fl : ar[st][i];

if (temp = dfs(i, ed, flow))

{

ar[st][i] -= temp;

ar[i][st] += temp;

return temp;

}

}

}

return 0;

}

int bfs(int st, int ed)

{

int i, j, k, u, t;

queue<int>q;

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

dep[st] = 0;

q.push(st);

while (!q.empty())

{

t = q.front();

q.pop();

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

{

if (dep[i] == -1 && ar[t][i])

{

dep[i] = dep[t] + 1;

q.push(i);

}

}

}

if (dep[ed] == -1)

return 0;

return 1;

}

1533 权值二分图 KM算法；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <algorithm>

#include <cmath>

using namespace std;

const int MAX = 100000000;

const int MIN = -100000000;

int ar[110][110];

char sa[110][110];

bool visitx[110], visity[110];

int lx[110], ly[110];

int link[110];

int N, M, Cntx, Cnty;

int KM();

bool dfs(int f);

int main()

{

int i, j, k, u, n, m, a, b, c, res;

while (scanf("%d %d", &N, &M) != EOF && (N || M))

{

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

{

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

}

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

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

Cntx = Cnty = 1;

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

{

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

{

if (sa[i][j] == 'm')

{

for (a = 1; a <= N; a++)

{

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

{

if (sa[a][b] == 'H')

{

ar[Cntx][Cnty++] = (-1)\*(abs(i - a) + abs(j - b));

}

}

}

Cntx++;

Cnty = 1;

}

}

}

//for (i = 1; i < Cntx; i++)

//{

// for (j = 1; j < Cntx; j++)

// {

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

// }

// putchar('\n');

//}

Cntx -= 1;

Cnty = Cntx;

printf("%d\n", -KM());

}

return 0;

}

int KM()

{

int i, j, k, u, n, m, temp, d, res;

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

lx[i] = MIN;

memset(ly, 0, sizeof(ly));

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

{

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

{

lx[i] = (lx[i] < ar[i][j]) ? ar[i][j] : lx[i];

}

}

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

{

while (1)

{

memset(visitx, false, sizeof(visitx));

memset(visity, false, sizeof(visity));

temp = MAX;

if (dfs(i))

break;

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

{

if (visitx[u])

{

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

{

if (!visity[j])

{

temp = (temp < lx[u] + ly[j] - ar[u][j]) ? temp : (lx[u] + ly[j] - ar[u][j]);

}

}

}

}

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

{

if (visitx[k])

lx[k] -= temp;

if (visity[k])

ly[k] += temp;

}

}

}

//printf("%d %d\n", Cntx, Cnty);

//for (i = 1; i <= Cntx; i++)

// printf("%d ", link[i]);

//putchar('\n');

res = 0;

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

if (link[i] != -1)

res += ar[link[i]][i];

return res;

}

bool dfs(int f)

{

int i, j, k, temp;

visitx[f] = true;

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

{

if (!visity[i] && lx[f] + ly[i] == ar[f][i])

{

visity[i] = true;

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

{

link[i] = f;

return true;

}

}

}

return false;

}

匈牙利；

#include <cstdio>

#include <cstring>

#include <cstdlib>

using namespace std;

bool visit[510];

int ar[510][510];

int link[510];

int res, M, N;

bool dfs(int f);

int main()

{

int i, j, k, u, a, b, n, m;

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

{

scanf("%d %d", &M, &N);

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

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

res = 0;

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

{

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

ar[a][b] = 1;

}

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

{

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

if (dfs(i))

res++;

}

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

}

return 0;

}

bool dfs(int f)

{

int i, j;

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

{

if (ar[f][i] && !visit[i])

{

visit[i] = true;

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

{

link[i] = f;

return true;

}

}

}

return false;

}

二分匹配第3题 依然匈牙利；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

using namespace std;

bool visit[110];

int ar[110][110];

int leftN[110], rightN[110], res, ans, b;

int link[110];

bool dfs(int f);

int main()

{

int i, j, k, n, m, u, a, p, q, count, cas = 1;

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

{

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

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

memset(leftN, 0, sizeof(leftN));

memset(rightN, 0, sizeof(rightN));

res = 0;

ans = 0;

count = 0;

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

{

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

ar[leftN[i]][rightN[i]] = 1;

}

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

{

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

if (dfs(i))

res++;

}

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

{

ar[leftN[i]][rightN[i]] = 0;

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

ans = 0;

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

{

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

if (dfs(j))

ans++;

}

if (ans != res)

count++;

ar[leftN[i]][rightN[i]] = 1;

}

printf("Board %d have %d important blanks for %d chessmen.\n", cas++, count, res);

}

return 0;

}

bool dfs(int f)

{

int i, j, k, u;

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

{

if (!visit[i] && ar[f][i])

{

visit[i] = true;

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

{

link[i] = f;

return true;

}

}

}

return false;

}

**数据结构：**

字典树模板 指针只是门牌号的纸，不往上写号码怎么行？

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);

}

/\*Trie树(字典树) 2011.10.10\*/

#include <iostream>

#include<cstdlib>

#define MAX 26

using namespace std;

typedef struct TrieNode //Trie结点声明

{

bool isStr; //标记该结点处是否构成单词

struct TrieNode \*next[MAX]; //儿子分支

}Trie;

void insert(Trie \*root,const char \*s) //将单词s插入到字典树中

{

if(root==NULL||\*s=='\0')

return;

int i;

Trie \*p=root;

while(\*s!='\0')

{

if(p->next[\*s-'a']==NULL) //如果不存在，则建立结点

{

Trie \*temp=(Trie \*)malloc(sizeof(Trie));

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

{

temp->next[i]=NULL;

}

temp->isStr=false;

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

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

}

else

{

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

}

s++;

}

p->isStr=true; //单词结束的地方标记此处可以构成一个单词

}

int search(Trie \*root,const char \*s) //查找某个单词是否已经存在

{

Trie \*p=root;

while(p!=NULL&&\*s!='\0')

{

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

s++;

}

return (p!=NULL&&p->isStr==true); //在单词结束处的标记为true时，单词才存在

}

void del(Trie \*root) //释放整个字典树占的堆区空间

{

int i;

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

{

if(root->next[i]!=NULL)

{

del(root->next[i]);

}

}

free(root);

}

int main(int argc, char \*argv[])

{

int i;

int n,m; //n为建立Trie树输入的单词数，m为要查找的单词数

char s[100];

Trie \*root= (Trie \*)malloc(sizeof(Trie));

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

{

root->next[i]=NULL;

}

root->isStr=false;

scanf("%d",&n);

getchar();

for(i=0;i<n;i++) //先建立字典树

{

scanf("%s",s);

insert(root,s);

}

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

{

for(i=0;i<m;i++) //查找

{

scanf("%s",s);

if(search(root,s)==1)

printf("YES\n");

else

printf("NO\n");

}

printf("\n");

}

del(root); //释放空间很重要

return 0;

}

线段树 区间更新，线段长度最值问题；

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

struct node

{

int l;

int r;

int mid;

int lx;

int rx;

int max;

int visit;

};

node sa[50000 \* 4];

int ar[50000 \* 4];

int N, M;

void build(int root, int begin, int end);

void update(int root, int b, int e, int val);

void pushdown(int root);

void pushup(int root);

int query(int root, int num);

int main()

{

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

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

{

build(1, 1, N);

for (m = 1; m <= M; m++)

{

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

if (a == 1)

{

if (sa[1].max < b)

{

printf("0\n");

continue;

}

int x = query(1, b);

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

update(1, x, x + b - 1, m);

}

else

{

scanf("%d", &c);

update(1, b, b + c - 1, 0);

}

}

}

return 0;

}

void build(int root, int begin, int end)

{

sa[root].l = begin;

sa[root].r = end;

sa[root].mid = (begin + end) / 2;

sa[root].visit = -1;

if (begin == end)

{

sa[root].lx = sa[root].rx = sa[root].max = 1;

}

else

{

build(root \* 2, sa[root].l, sa[root].mid);

build(root \* 2 + 1, sa[root].mid + 1, sa[root].r);

pushup(root);

}

}

void pushup(int root)

{

sa[root].lx = sa[root \* 2].lx + ((sa[root \* 2].lx == (sa[root].mid - sa[root].l + 1)) ? sa[root \* 2 + 1].lx : 0);

sa[root].rx = sa[root \* 2 + 1].rx + ((sa[root \* 2 + 1].rx == (sa[root].r - sa[root].mid)) ? sa[root \* 2].rx : 0);

sa[root].max = max(max(sa[root \* 2].max, sa[root \* 2 + 1].max), sa[root \* 2].rx + sa[root \* 2 + 1].lx);

}

void update(int root, int b, int e, int val)

{

if (sa[root].l > e || sa[root].r < b)

return;

if (sa[root].l >= b && sa[root].r <= e)

{

sa[root].visit = val;

if (val != 0)

sa[root].lx = sa[root].rx = sa[root].max = 0;

else

{

sa[root].rx = sa[root].lx = sa[root].max = sa[root].r - sa[root].l + 1;

}

return;

}

pushdown(root);

update(root \* 2, b, e, val);

update(root \* 2 + 1, b, e, val);

pushup(root);

}

void pushdown(int root)

{

if (sa[root].visit != -1)

{

sa[root \* 2].visit = sa[root].visit;

sa[root \* 2 + 1].visit = sa[root].visit;

if (sa[root].visit == 0)

{

sa[root \* 2].max = (sa[root].mid - sa[root].l + 1);

sa[root \* 2].lx = (sa[root].mid - sa[root].l + 1);

sa[root \* 2].rx = (sa[root].mid - sa[root].l + 1);

sa[root \* 2 + 1].max = (sa[root].r - sa[root].mid);

sa[root \* 2 + 1].lx = (sa[root].r - sa[root].mid);

sa[root \* 2 + 1].rx = (sa[root].r - sa[root].mid);

}

else

{

sa[root \* 2].max = sa[root \* 2].lx = sa[root \* 2].rx = 0;

sa[root \* 2 + 1].max = sa[root \* 2 + 1].lx = sa[root \* 2 + 1].rx = 0;

}

sa[root].visit = -1;

}

}

int query(int root, int num)

{

//if (sa[root].max < num)

// return 0;

if (sa[root].l == sa[root].r)

return 1;

pushdown(root);

if (sa[root \* 2].max >= num)

return query(root \* 2, num);

if (sa[root \* 2].rx + sa[root \* 2 + 1].lx >= num)

return (sa[root].mid - sa[root \* 2].rx + 1);

return query(root \* 2 + 1, num);

}

3308 线段树第5题，LCIS，线段长度最值；

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

const int MAX = 100000000;

//const int max(int a, int b) { return (a > b) ? a : b; };

//const int min(int a, int b) { return (a < b) ? a : b; };

struct node

{

int l;

int r;

int mid;

int lx;

int rx;

int mx;

};

node edge[100000 \* 4];

int ar[100010];

void build(int root, int begin, int end);

void update(int root, int begin, int end, int index, int val);

int query(int root, int begin, int end, int b, int e);

void pushup(int root);

int N, M;

int main()

{

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

char s[5];

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

{

for (m = 1; m <= n; m++)

{

scanf("%d %d", &N, &M);

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

{

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

}

build(1, 0, N - 1);

//for(int i=1;i<=30;i++)

// printf("%d %d %d %d %d\n",edge[i].l,edge[i].r,edge[i].mx,edge[i].lx,edge[i].rx);

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

{

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

if (strcmp(s, "Q") == 0)

{

printf("%d\n", query(1, 0, N - 1, a, b));

}

else

{

ar[a] = b;

update(1, 0, N - 1, a, b);

}

}

}

}

return 0;

}

void build(int root, int begin, int end)

{

edge[root].l = begin;

edge[root].r = end;

edge[root].mid = (begin + end) / 2;

if (begin == end)

{

edge[root].lx = edge[root].rx = edge[root].mx = 1;

}

else

{

build(root \* 2, begin, edge[root].mid);

build(root \* 2 + 1, edge[root].mid + 1, end);

pushup(root);

}

}

void pushup(int root)

{

edge[root].lx = edge[root \* 2].lx + (((edge[root \* 2].r - edge[root \* 2].l + 1) == edge[root \* 2].lx && ar[edge[root \* 2].r] < ar[edge[root \* 2 + 1].l]) ? (edge[root \* 2 + 1].lx) : 0);

edge[root].rx = edge[root \* 2 + 1].rx + (((edge[root \* 2 + 1].r - edge[root \* 2 + 1].l + 1) == edge[root \* 2 + 1].rx && ar[edge[root \* 2 + 1].l] > ar[edge[root \* 2].r]) ? (edge[root \* 2].rx) : 0);

edge[root].mx = max(max(edge[root \* 2].mx, edge[root \* 2 + 1].mx), (ar[edge[root \* 2].r] < ar[edge[root \* 2 + 1].l]) ? (edge[root \* 2].rx + edge[root \* 2 + 1].lx) : 0);

}

void update(int root, int begin, int end, int index, int val)

{

if (begin == end)

{

edge[root].lx = edge[root].rx = edge[root].mx = 1;

}

else

{

if (index <= edge[root].mid)

update(root \* 2, begin, edge[root].mid, index, val);

else

update(root \* 2 + 1, edge[root].mid + 1, end, index, val);

pushup(root);

}

}

int query(int root, int begin, int end, int b, int e)

{

if (begin >= b && end <= e)

return edge[root].mx;

if (begin > e || end < b)

return 0;

int res = 0;

if (ar[edge[root \* 2].r] < ar[edge[root \* 2 + 1].l])

{

res = min(edge[root].mid - b + 1, edge[root \* 2].rx) + min(e - edge[root].mid, edge[root \* 2 + 1].lx);

return max(max(query(root \* 2, begin, edge[root].mid, b, e), query(root \* 2 + 1, edge[root].mid + 1, end, b, e)), res);

}

else

return max(query(root \* 2, begin, edge[root].mid, b, e), query(root \* 2 + 1, edge[root].mid + 1, end, b, e));

}

线段树，区间更新；

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

const int MAX = 100000000;

int ar[100000 \* 4];

int sa[100000 \* 4];

int visit[100000 \* 4];

int lef[100000 \* 4], rig[100000 \* 4];

int N;

void build(int root, int begin, int end);

void update(int root, int begin, int end, int b, int e, int val);

void pushdown(int root);

int main()

{

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

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

{

for (m = 1; m <= n; m++)

{

scanf("%d", &N);

build(1, 1, N);

scanf("%d", &k);

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

{

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

update(1, 1, N, a, b, c);

}

printf("Case %d: The total value of the hook is %d.\n", m, sa[1]);

}

}

return 0;

}

void build(int root, int begin, int end)

{

visit[root] = 0;

lef[root] = begin;

rig[root] = end;

sa[root] = 0;

if (begin == end)

sa[root] = 1;

else

{

build(root \* 2, begin, (begin + end) / 2);

build(root \* 2 + 1, (begin + end) / 2+1, end);

sa[root] = sa[root \* 2] + sa[root \* 2 + 1];

}

}

void update(int root, int begin, int end, int b, int e, int val)

{

if (begin > e || end < b)

return;

if (begin >= b && end <= e)

{

sa[root] = val\*(rig[root] - lef[root] + 1);

visit[root] = val;

return;

}

pushdown(root);

update(root \* 2, begin, (begin + end) / 2, b, e, val);

update(root \* 2 + 1, (begin + end) / 2+1, end, b, e, val);

sa[root] = sa[root \* 2] + sa[root \* 2 + 1];

}

void pushdown(int root)

{

if (visit[root])

{

visit[root \* 2] = visit[root];

visit[root \* 2 + 1] = visit[root];

sa[root \* 2] = visit[root] \* (rig[root \* 2] - lef[root \* 2] + 1);

sa[root \* 2 + 1] = visit[root] \* (rig[root \* 2 + 1] - lef[root \* 2 + 1] + 1);

visit[root] = 0;

}

}

树状数组；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <algorithm>

using namespace std;

const int MAN = 100000000;

int main()

{

int i, j, k, n, m, u, a, b;

int ar[32010], sa[32010], sum;

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

{

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

memset(sa, 0, sizeof(sa));

for (m = 1; m <= n; m++)

{

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

a++;

k = a;

sum = 0;

while (k)

{

sum += ar[k];

k -= k & -k;

}

sa[sum]++;

k = a;

while (k <= 32000)

{

ar[k] += 1;

k += k & -k;

}

}

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

{

printf("%d\n", sa[i]);

}

}

return 0;

}

树状数组块状；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cmath>

using namespace std;

const int MAX = 100000000;

int main()

{

int i, j, k, n, m, a, b, c, p, q, ar[1010][1010], sa[1010][1010], cas, sum, doudou, x1, x2, y1, y2, temp;

char s[5];

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

{

cas = 1;

for (m = 1; m <= n; m++)

{

doudou = 0;

scanf("%d", &p);

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

memset(sa, 0, sizeof(sa));

for (i = 1; i <= 1005; i++)

{

for (j = 1; j <= 1005; j++)

{

ar[i][j] = 1;

for (a = i; a <= 1005; a += (a & (-a)))

{

for (b = j; b <= 1005; b += (b & (-b)))

{

sa[a][b] += 1;

}

}

}

}

for (q = 1; q <= p; q++)

{

scanf("%s", s);

if (!strcmp(s, "S"))

{

sum = 0;

scanf("%d %d %d %d", &x1, &y1, &x2, &y2);

x1++;

y1++;

x2++;

y2++;

if (x1 > x2)

{

temp = x1;

x1 = x2;

x2 = temp;

}

if (y1 > y2)

{

temp = y1;

y1 = y2;

y2 = temp;

}

for (i = x2; i > 0; i -= (i & (-i)))

{

for (j = y2; j > 0; j -= (j & (-j)))

{

sum += sa[i][j];

}

}

for (i = x1 - 1; i > 0; i -= (i & (-i)))

{

for (j = y2; j > 0; j -= (j & (-j)))

{

sum -= sa[i][j];

}

}

for (i = x2; i > 0; i -= (i & (-i)))

{

for (j = y1 - 1; j > 0; j -= (j & (-j)))

{

sum -= sa[i][j];

}

}

for (i = x1 - 1; i > 0; i -= (i & (-i)))

{

for (j = y1 - 1; j > 0; j -= (j & (-j)))

{

sum += sa[i][j];

}

}

if (!doudou)

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

doudou = 1;

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

}

else if (!strcmp(s, "A"))

{

scanf("%d %d %d", &x1, &y1, &c);

x1++;

y1++;

for (i = x1; i <= 1005; i += (i & (-i)))

{

for (j = y1; j <= 1005; j += (j & (-j)))

{

sa[i][j] += c;

}

}

ar[x1][y1] += c;

}

else if (!strcmp(s, "D"))

{

scanf("%d %d %d", &x1, &y1, &c);

x1++;

y1++;

if (c >= ar[x1][y1])

c = ar[x1][y1];

for (i = x1; i <= 1005; i += (i & (-i)))

{

for (j = y1; j <= 1005; j += (j & (-j)))

{

sa[i][j] -= c;

}

}

ar[x1][y1] -= c;

}

else if (!strcmp(s, "M"))

{

scanf("%d %d %d %d %d", &x1, &y1, &x2, &y2, &c);

x1++;

y1++;

x2++;

y2++;

if (c >= ar[x1][y1])

c = ar[x1][y1];

for (i = x1; i <= 1005; i += (i & (-i)))

{

for (j = y1; j <= 1005; j += (j & (-j)))

{

sa[i][j] -= c;

}

}

for (i = x2; i <= 1005; i += (i & (-i)))

{

for (j = y2; j <= 1005; j += (j & (-j)))

{

sa[i][j] += c;

}

}

ar[x1][y1] -= c;

ar[x2][y2] += c;

}

}

}

}

return 0;

}

3342 拓补排序，判断有没有环，点数>入队次数即有环；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <queue>

#include <vector>

using namespace std;

const int MAX = 100000000;

const int MAXN = 110;

vector<int>G[500];

int in[MAXN];

int N, M, Sum;

int main()

{

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

while (scanf("%d %d", &N, &M) != EOF && (N || M))

{

memset(in, 0, sizeof(in));

Sum = N;

for (i = 0; i < 500; i++)

G[i].clear();

for (m = 1; m <= M; m++)

{

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

G[a].push\_back(b);

in[b]++;

}

queue<int>q;

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

{

if (in[i] == 0)

{

q.push(i);

Sum--;

}

}

while (!q.empty())

{

u = q.front();

q.pop();

for (i = 0; i < G[u].size(); i++)

{

in[G[u][i]]--;

if (in[G[u][i]] == 0)

{

Sum--;

q.push(G[u][i]);

}

}

}

if (Sum == 0)

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

3357 拓补排序，判断有几条成环的边，分别以a，b为过度点；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <queue>

#include <vector>

using namespace std;

const int MAX = 100000000;

const int MAXN = 250;

int ar[MAXN][MAXN];

int N, M;

int main()

{

int i, j, k, u, n, m, a, b, c, sum, cas = 1;

while (scanf("%d %d", &N, &M) != EOF && (N || M))

{

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

sum = 0;

for (m = 1; m <= M; m++)

{

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

if (ar[b][a] || a == b)

{

sum++;

continue;

}

if (ar[a][b])

continue;

ar[a][b] = 1;

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

{

if (ar[i][a])

{

ar[i][b] = 1;

}

if (ar[b][i])

{

ar[a][i] = 1;

}

}

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

{

if (ar[i][a])

{

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

{

if (ar[b][j])

{

ar[i][j] = 1;

}

}

}

}

}

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

}

return 0;

}

1285 拓补排序第2题， 字典序，即优先队列即可；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <queue>

#include <vector>

using namespace std;

const int MAX = 100000000;

const int MAXN = 510;

struct node

{

int to;

int next;

bool operator < (const node &p) const

{

return to > p.to;

}

};

node edge[MAXN \* 10];

int head[MAXN], in[MAXN];

int cnt, N;

void addedge(int x, int y);

void tuobu();

int main()

{

int i, j, k, u, n, m, a, b;

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

{

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

memset(in, 0, sizeof(in));

cnt = 0;

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

{

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

addedge(a, b);

in[b]++;

}

tuobu();

putchar('\n');

}

return 0;

}

void addedge(int x, int y)

{

edge[cnt].to = y;

edge[cnt].next = head[x];

head[x] = cnt++;

}

void tuobu()

{

int i, j, k, u, v, t, doudou = 0;

node now, after;

priority\_queue<node>q;

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

{

if (in[i] == 0)

{

now.to = i;

now.next = head[i];

q.push(now);

in[i]--;

}

}

while (!q.empty())

{

now = q.top();

q.pop();

if (doudou)

putchar(' ');

doudou = 1;

printf("%d", now.to);

for (i = head[now.to]; i != -1; i = edge[i].next)

{

after = edge[i];

in[after.to]--;

if (in[after.to] == 0)

{

q.push(after);

}

}

}

}

4857 拓补排序， 反向建图 小的在大的前面且小的数优先级高，最前面的不一定是最小的，但是最后面的一定是最大的

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#include <cctype>

#include <queue>

#include <deque>

#include <vector>

#include <map>

using namespace std;

const int MAX = 100000000;

const int MAXN = 30010;

struct node

{

int to;

int next;

};

node edge[100010];

int head[MAXN], in[MAXN];

int cnt, N, M;

void tuobu();

void addedge(int x, int y);

int main()

{

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

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

{

for (m = 1; m <= n; m++)

{

memset(in, 0, sizeof(in));

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

cnt = 0;

scanf("%d %d", &N, &M);

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

{

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

addedge(b, a);

in[a]++;

}

tuobu();

printf("\n");

}

}

return 0;

}

void addedge(int x, int y)

{

edge[cnt].to = y;

edge[cnt].next = head[x];

head[x] = cnt++;

}

void tuobu()

{

vector<int>ar;

ar.clear();

int i, j, k, u, v, sta = 0;

priority\_queue<int>q;

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

if (in[i] == 0)

{

in[i]--;

q.push(i);

}

while (!q.empty())

{

u= q.top();

q.pop();

ar.push\_back(u);

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

{

in[edge[i].to]--;

if (in[edge[i].to] == 0)

{

q.push(edge[i].to);

}

}

}

for (i = ar.size() - 1; i >= 0; i--)

{

if (i != ar.size() - 1)

putchar(' ');

printf("%d", ar[i]);

}

}

4300 KMP第8题，EKMP+MAP；

#include <stdio.h>

#include <cstring>

#include <map>

using namespace std;

int Mainl, Assl;

char ar[100010], sa[100010], s[100010];

int Next[100010], EX[100010];

void Cal\_Next();

void EKMP();

map<char, char>q;

int main()

{

int i, j, k, n, m;

char ch;

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

{

getchar();

q.clear();

for (m = 1; m <= n; m++)

{

scanf("%s", ar);

for (i = 0; i < 26; i++)

{

q[ar[i]] = 'a'+i;

}

scanf("%s", sa);

Mainl = strlen(sa);

strcpy(s, sa);

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

{

sa[i] = q[sa[i]];

}

EKMP();

//for (i = 0; i < Mainl; i++)

//{

// printf("%d ", EX[i]);

//}

//for (i = 0; i < Mainl; i++)

//{

// printf("%d ", Next[i]);

//}

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

{

if (i >= EX[i] && i + EX[i] == Mainl)

break;

}

//puts(sa);

//puts(s);

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

putchar(s[j]);

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

putchar(sa[j]);

putchar('\n');

}

}

return 0;

}

void EKMP()

{

int i = 0, j = 0, k, po = 0;

Cal\_Next();

while (s[i] == sa[i] && i < Mainl)

i++;

EX[0] = i;

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

{

if (Next[i - po] + i < EX[po] + po)

EX[i] = Next[i - po];

else

{

j = EX[po] + po - i;

if (j < 0)

j = 0;

while (s[i + j] == sa[j] && i + j < Mainl)

j++;

EX[i] = j;

po = i;

}

}

}

void Cal\_Next()

{

int i, j = 0, po = 1;

Next[0] = Mainl - 1;

while (sa[j] == sa[j + 1] && j < Mainl)

j++;

Next[1] = j;

for (i = 2; i < Mainl; i++)

{

if (Next[i - po] + i < Next[po] + po)

Next[i] = Next[i - po];

else

{

j = Next[po] + po - i;

if (j < 0)

j = 0;

while (sa[i + j] == sa[j] && i + j < Mainl)

j++;

Next[i] = j;

po = i;

}

}

}

回文串匹配

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<string>

#include<queue>

#include<algorithm>

#include<map>

#include<iomanip>

#define INF 99999999

using namespace std;

const int MAX = 500000 + 10;

char s1[MAX], s2[MAX];

int Next[MAX], extend1[MAX], extend2[MAX];

int sum[MAX], val[27];

void get\_Next(char \*a, int len) {

int k = 0, i = 1, j;

Next[0] = len;

while (k + 1<len && a[k] == a[k + 1])++k;

Next[1] = k;

k = 1;

for (i = 2; i < len; i++)

{

if (Next[i - k] + i < Next[k] + k)

Next[i] = Next[i - k];

else

{

j = Next[k] + k - i;

if (j < 0)

j = 0;

while (a[i + j] == a[j] && i + j < len)

j++;

Next[i] = j;

k = i;

}

}

}

void EKMP(char \*a, char \*b, int \*extend, int len) {

get\_Next(a, len);

int k = 0, i = 0, j;

while (k < len && a[k] == b[k])++k;

extend[0] = k;

k = 0;

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

{

if (Next[i - k] + i < extend[k] + k)

extend[i] = Next[i - k];

else

{

j = extend[k] + k - i;

if (j < 0)

j = 0;

while (b[i + j] == a[j] && i + j < len)

j++;

extend[i] = j;

k = i;

}

}

}

int main() {

int n;

cin >> n;

while (n--) {

for (int i = 0; i<26; ++i)cin >> val[i];

scanf("%s", s1);

int len = strlen(s1);

for (int i = 1; i <= len; ++i) {

sum[i] = sum[i - 1] + val[s1[i - 1] - 'a'];

s2[i - 1] = s1[len - i];

}

EKMP(s1, s2, extend1, len);

EKMP(s2, s1, extend2, len);

int ans = 0, temp = 0;

for (int i = 1; i<len; ++i) {

if (extend1[len - i] == i)temp += sum[i];

if (extend2[i] == len - i)temp += sum[len] - sum[i];

if (temp>ans)ans = temp;

temp = 0;

}

cout << ans << endl;

}

return 0;

}

**dp：**

3853 概率dp第2题，dp数组记录的是走向答案的结果；

#include <cstdio>

#include <cstring>

#include <cmath>

#include <cctype>

using namespace std;

struct node

{

double stay;

double right;

double below;

};

node ar[1010][1010];

double sa[1010][1010];

int N, M;

int main()

{

int i, j, k, u, n, m, a, b;

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

{

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

{

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

{

scanf("%lf %lf %lf", &ar[i][j].stay, &ar[i][j].right, &ar[i][j].below);

}

}

memset(sa, 0, sizeof(sa));

for (i = N; i >= 1; i--)

{

for (j = M; j >= 1; j--)

{

if (i == N && j == M)

continue;

if (1.0 - ar[i][j].stay < 1.0e-9)

continue;

sa[i][j] = (ar[i][j].right\*sa[i][j + 1] + ar[i][j].below\*sa[i + 1][j] + 2.0) / (1.0 - ar[i][j].stay);

}

}

printf("%.3f\n", sa[1][1]);

}

return 0;

}

数位dp 每个数位上的数地位不平等，需要dp数组有pre；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <algorithm>

#include <cctype>

#include <cmath>

using namespace std;

typedef long long ll;

const int MAX = 11;

ll dp[11][11];

int ar[11];

int Count;

ll dfs(int pos, int pre, int limit, int first);

ll solve(ll n);

int main()

{

int i, j, u, max, a, b;

ll n, m, k, res;

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

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

{

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

{

scanf("%lld %lld", &n, &m);

//printf("%lld %lld\n", solve(m), solve(n - 1));

printf("%lld\n", solve(m) - solve(n - 1));

}

}

return 0;

}

ll solve(ll n)

{

int i, j;

Count = 0;

while (n)

{

ar[++Count] = n % 10;

n /= 10;

}

return dfs(Count, 0, 1, 1);

}

ll dfs(int pos, int pre, int limit, int first)

{

int i, j, k, max;

ll res = 0;

if (pos == 0)

return 1;

if (!limit && dp[pos][pre] != -1)

return dp[pos][pre];

max = limit ? ar[pos] : 9;

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

{

if (first)

res += dfs(pos - 1, i, limit && i == ar[pos], first && i == 0);

else if (i == 0)

continue;

else if (i > pre || pre % i)

continue;

else

res += dfs(pos - 1, i, limit && i == ar[pos], first && i == 0);

}

if (!limit)

dp[pos][pre] = res;

return res;

}

数位dp 不仅记录个数，还要求平方和，需要count，sum， square三个同时记录， cur.square = （next.count \* i² + 2 \* next.sum \* i + next.square）；

如果求和，就是next.count \* i + next.sum;

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cmath>

#include <cctype>

#include <algorithm>

using namespace std;

typedef long long ll;

const int Mod = 1000000007;

struct node

{

ll count;

ll sum;

ll square;

ll visit;

node() {}

node(ll c, ll s, ll sq, ll v) : count(c), sum(s), square(sq), visit(v) {}

};

node dp[20][8][8];

ll Pow[20];

int ar[20];

ll solve(ll n);

node dfs(int pos, int limit, int sta1, int sta2);

int main()

{

int i, j, u, count, k;

ll n, m, res;

Pow[0] = 1;

for (i = 1; i < 20; i++)

Pow[i] = Pow[i - 1] \* 10;

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

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

{

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

{

scanf("%lld %lld", &n, &m);

printf("%lld\n", ((solve(m) - solve(n - 1)) % Mod + Mod) % Mod);

}

}

return 0;

}

ll solve(ll n)

{

int count = 0, i, j;

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

while (n)

{

ar[++count] = n % 10;

n /= 10;

}

return dfs(count, 1, 0, 0).square;

}

node dfs(int pos, int limit, int sta1, int sta2)

{

int i, j, u, max;

node res(0, 0, 0, 0);

node next;

if (pos == 0)

return (sta1 && sta2) ? node(1, 0, 0, 0) : node(0, 0, 0, 0);

if (!limit && dp[pos][sta1][sta2].visit != -1)

return dp[pos][sta1][sta2];

max = limit ? ar[pos] : 9;

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

{

if (i == 7)

continue;

next = dfs(pos - 1, limit && i == ar[pos], (i + sta1) % 7, (i + 10 \* sta2) % 7);

ll temp = (i \* Pow[pos - 1]) % Mod;

res.count = (res.count + next.count) % Mod;

res.sum = (res.sum + (next.sum + next.count \* temp) % Mod) % Mod;

res.square = (res.square + ((next.count\*temp%Mod\*temp%Mod + 2 \*temp\*next.sum%Mod) % Mod + next.square) % Mod) % Mod;

}

if (!limit)

{

res.visit = 1;

dp[pos][sta1][sta2] = res;

}

return res;

}

加减背包， 无非就是范围开成2倍的sum；

#include<cstdio>

#include<cstring>

#include<cmath>

using namespace std;

int main()

{

int i, j, k, u, n, m, ar[110], dp[30010], sum, p, q;

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

{

for (m = 1; m <= n; m++)

{

scanf("%d", &p);

sum = 0;

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

{

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

sum += ar[i];

}

for (i = 0; i <= sum \* 2; i++)

{

dp[i] = -sum;

}

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

{

for (k = 1; k <= 2; k++)

{

for (j = 2 \* sum; j >= ar[i]; j--)

{

dp[j] = (dp[j] > dp[j - ar[i]] + ar[i]) ? dp[j] : (dp[j - ar[i]] + ar[i]);

}

}

}

//for (i = 0; i <= sum \* 2; i++)

// printf("%d ", dp[i]);

//putchar('\n');

scanf("%d", &k);

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

{

scanf("%d", &q);

if (dp[q+sum] == q)

printf("YES\n");

else

printf("NO\n");

}

}

}

return 0;

}

3496 二维背包， 初始化一维都要初始化，因为只要买的数量是0，不管time给多少都是0，因为需要的time有负数，所以初始化time时要赋为负无穷,

数组要开大一点；

#include <stdio.h>

#include <string.h>

struct moive

{

int time;

int value;

};

int main()

{

int n, m, i, j, k, u, bag[120][1200], mtime, mnum, c;

struct moive duoduo[120];

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

{

for (m = 1; m <= n; m++)

{

scanf("%d %d %d", &c, &mnum, &mtime);

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

{

scanf("%d %d", &duoduo[u].time, &duoduo[u].value);

}

for (i = 0; i < 120; i++)

{

for (j = 0; j < 1200; j++)

bag[i][j] = -10000000;

}

for (j = 0; j < 1200; j++)

bag[0][j] = 0;

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

{

for (j = mnum; j >= 1; j--)

{

for (k = mtime; k >= duoduo[i].time && k >= 0; k--)

bag[j][k] = (bag[j][k] > bag[j - 1][k - duoduo[i].time] + duoduo[i].value) ? bag[j][k] : (bag[j - 1][k - duoduo[i].time] + duoduo[i].value);

}

}

if (bag[mnum][mtime] < 0)

bag[mnum][mtime] = 0;

printf("%d\n", bag[mnum][mtime]);

}

}

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;

}

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;

}

//1160 mouse's speed

//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;

}

1003 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;

}

最长公共子列；

#include <cstdio>

#include <cstring>

#include <string>

#include <cctype>

using namespace std;

int main()

{

int i, j, k, u, n, m, max, L1, L2;

char s1[1000], s2[1000];

int ar[1000][1000];

while (scanf("%s %s", s1+1, s2+1) != EOF)

{

L1 = strlen(s1+1);

L2 = strlen(s2+1);

for (i = 0; i <= L1; i++)

{

for (j = 0; j <= L2; j++)

{

if (i == 0 || j == 0)

{

ar[i][j] = 0;

continue;

}

if (s1[i] != s2[j])

{

ar[i][j] = (ar[i - 1][j] > ar[i][j - 1]) ? ar[i - 1][j] : ar[i][j - 1];

}

if (s1[i] == s2[j])

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

}

}

printf("%d\n", ar[L1][L2]);

}

return 0;

}

最长公共上升子列；

#include <cstdio>

#include <cstring>

using namespace std;

int main()

{

int i, j, k, n, m, max, L1, L2;

char ar[100];

char a[100], b[100];

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

{

for (m = 1; m <= n; m++)

{

scanf("%s %s", a + 1, b + 1);

L1 = strlen(a + 1);

L2 = strlen(b + 1);

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

for (i = 1; i <= L1; i++)

{

max = 0;

for (j = 1; j <= L2; j++)

{

if (a[i] > b[j] && max < ar[j])

max = ar[j];

if (a[i] == b[j])

ar[j] = max + 1;

}

}

max = 0;

for (i = 1; i <= L2; i++)

if (max < ar[i])

max = ar[i];

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

}

}

return 0;

}

**杂题：**

二分查找

2446 对二分查找不够熟练啊！！！！整数 搞了好久；判断条件是r>l，结果是mid，如果是double，不用加==条件，是int，需要加==条件；

#include <cstdio>

#include <cstring>

#include <cmath>

using namespace std;

typedef long long ll;

ll Floor[2002000], num[2002000];

int main()

{

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

ll N;

Floor[0] = num[0] = 0;

for (i = 1; i < 2002000; i++)

{

num[i] = num[i - 1] + i;

Floor[i] = Floor[i - 1] + num[i];

}

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

{

for (m = 1; m <= n; m++)

{

scanf("%lld", &N);

ll l = 1;

ll r = 2002000;

ll mid = (l + r) / 2;

while (r > l)

{

if (Floor[mid] < N)

l = mid + 1;

else if (Floor[mid] > N)

r = mid - 1;

else

break;

mid = (l + r) / 2;

}

if (Floor[mid] < N)

mid++;

ll F = mid;

l = 1;

r = F;

mid = (l + r) / 2;

while (r > l)

{

if (num[mid] < N - Floor[F - 1])

l = mid + 1;

else if (num[mid] > N - Floor[F - 1])

r = mid - 1;

else

break;

mid = (l + r) / 2;

}

ll row = mid;

if (num[row] == N - Floor[F - 1])

printf("%lld %lld %lld\n", F, row, row);

else

{

if (num[row] < N - Floor[F - 1])

row++;

printf("%lld %lld %lld\n", F, row, N - Floor[F - 1] - num[row - 1]);

}

}

}

return 0;

}

1969 二分查找，实数；

#include <cstdio>

#include <cstring>

#include <cmath>

#include <cctype>

#include <algorithm>

using namespace std;

const double PI = acos(-1.0);

int main()

{

int i, j, k, u, n, m, a, b, c, r[10010];

double x, y, s[10010];

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

{

for (m = 1; m <= n; m++)

{

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

b++;

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

{

scanf("%d", &r[u]);

s[u] = PI\*r[u] \* r[u];

}

sort(s + 1, s + a+1);

//for (i = 1; i <= a; i++)

// printf("%lf ", s[i]);

//printf("%lf\n", PI);

double l = 0;

double R = s[a];

double mid = (R + l) / 2.0;

int sum = 0;

while (R - l > 1.0e-6)

{

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

{

sum = sum + (int)(s[i] / mid);

}

if (sum < b)

R = mid;

else

l = mid;

mid = (l + R) / 2.0;

sum = 0;

}

printf("%.4lf\n", mid);

}

}

return 0;

}

//6度分离，强数据版；

/\*\*

Problem C Friends

题意:给n个人，有n-1个朋友关系，形成一棵树，如果两个人能够通过不超过5个人可以联系到，那么那两个人也是朋友,问你每个人有多少个朋友,

比如(1-2-3-4-5-6-7-8)，1可以联系到2 3 4 5 6 7,所以1有6个朋友

思路：先对树进行dfs搜索形成有向树，根节点为1(任意),在设son[o][i]表示以o为节点在通过恰好i个人联系可以成为朋友的个数，只计算在根为o的子树中

的结果，那么设p1,p2,p3,p4,p5,p6为o的一级祖先、二级祖先、..六级祖先那么对于节点o来说，

不通过p1能成为朋友的个数为 sigma {son[o][i] | 0 <= i <= 6}， 计算的是包含o节点的

不通过p2而通过p1能成为朋友的个数为 sigma {son[p1][i] | 0 <= i <= 5}

但是这里再次计算了经过o的情况， 所以需要减去 sigma {son[o][i] | 0 <= i <= 4}

同理需要经过Pj祖先的时候， 总个数为 sigma {son[Pj][i] | 0 <= i <= 6 - j}

但是不能再通过Pj-1, 所以减去 sigma {son[Pj-1][i] | 0 <= i <= 6 - j - 1}

所有pj的结果加起来再减去1就是答案了， 因为把o节点自己计算了在内

\*\*/

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cmath>

#include <cctype>

#include <algorithm>

#include <queue>

#include <map>

#include <vector>

#include <deque>

using namespace std;

const int MIN = -100000000;

const int MAX = 100000000;

typedef long long ll;

vector<int>G[100010];

int pre[100010], node[100010][7];

void dfs(int \*fa, int n);

int main()

{

int i, j, k, u, n, m, a, b, c, res, doudou;

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

{

for (m = 1; m <= n; m++)

{

memset(node, 0, sizeof(node));

scanf("%d", &k);

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

{

G[i].clear();

node[i][0] = 1;

}

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

{

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

G[a].push\_back(b);

G[b].push\_back(a);

}

int fa[] = { 0, 0, 0, 0, 0, 0 };

dfs(fa, 1);

res = 0;

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

{

res = 0;

for (j = 1; j <= 6; j++)

res += node[i][j];

c = i;

for (u = 1; u <= 6; u++)

{

b = c;

c = pre[c];

if (c == 0)

break;

for (j = 0; j <= 6 - u; j++)

res += node[c][j];

for (j = 0; j <= 6 - u - 1; j++)

res -= node[b][j];

}

if (i == 1)

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

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

}

}

}

return 0;

}

void dfs(int \*fa, int n)

{

int i, j, k;

pre[n] = fa[5];

for (i = 0; i <= 5; i++)

{

node[fa[i]][6 - i]++;

}

int f[] = { fa[1], fa[2], fa[3], fa[4], fa[5], n };

for (i = 0; i < G[n].size(); i++)

{

if (G[n][i] != fa[5])

{

dfs(f, G[n][i]);

}

}

}

双指针，滑动区间；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

#include <algorithm>

#include <cmath>

using namespace std;

int main()

{

long long int i, j, k, u, n, m, a, b, c, t, sum, max, cost, ar[50100], sa[50100], hap, M, doudou;

long long int \*p, \*q;

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

{

for (m = 1; m <= n; m++)

{

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

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

{

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

}

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

{

scanf("%lld", &sa[i]);

}

sum = 0;

max = 0;

hap = 0;

M = 0;

for (p = ar+1, q = ar, i = 0, j = 1; i <= c;)

{

if (sum - max <= cost)

{

if (M < hap)

M = hap;

q++;

i++;

if (i > c)

break;

sum += \*q;

hap += sa[i];

max = (max > \*q) ? max : \*q;

}

else

{

sum -= \*p;

hap -= sa[j];

p++;

j++;

max = 0;

for (k = j; k <= i; k++)

{

if (max < ar[k])

max = ar[k];

}

}

}

printf("%lld\n", M);

}

}

return 0;

}

归并排序；

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <algorithm>

#include <cmath>

using namespace std;

typedef long long ll;

ll res;

int ar[1000010], temp[1000010];

void merge(int left, int mid, int right);

void merge\_sort(int left, int right);

int main()

{

int i, j, k, u, n, m, b;

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

{

for (m = 1; m <= n; m++)

{

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

}

res = 0;

merge\_sort(1, n);

//for (i = 1; i <= n; i++)

// printf("%d\n", ar[i]);

printf("%lld\n", res);

}

return 0;

}

void merge\_sort(int left, int right)

{

if (left >= right)

return;

int mid;

mid = (left + right) / 2;

merge\_sort(left, mid);

merge\_sort(mid + 1, right);

merge(left, mid, right);

}

void merge(int left, int mid, int right)

{

int i, j, k;

i = left;

j = mid + 1;

for (k = left; k <= right; k++)

{

temp[k] = ar[k];

}

for (k = left; k <= right; k++)

{

if (i > mid)

ar[k] = temp[j++];

else if (j > right)

ar[k] = temp[i++];

else if (temp[i] <= temp[j])

ar[k] = temp[i++];

else

{

ar[k] = temp[j++];

res += mid - i + 1;

}

}

}

同源数； 如果x和y的质因子集合完全相同，那么我们就说他们是同源的；

令 d=gcd(x,y)，将x和y不断与d进行约分。

若此时x=y=1，那么说明其质因子集合相同。

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cmath>

#include <cctype>

using namespace std;

typedef long long ll;

const int MAX = 100000000;

const int MIN = -100000000;

ll gcd(ll a, ll b);

int main()

{

ll i, j, k, u, n, m, p, q, a, b, temp, doudou;

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

{

doudou = 0;

q = gcd(n, m);

temp = q;

while (temp != 1)

{

temp = gcd(n, temp);

n /= temp;

if (n == 1)

{

doudou = 1;

break;

}

}

if (!doudou)

{

printf("No\n");

continue;

}

temp = q;

while (temp != 1)

{

temp = gcd(m, temp);

m /= temp;

if (temp == 1)

{

if (m != 1)

doudou = 0;

break;

}

}

if (doudou)

printf("Yes\n");

else

printf("No\n");

}

return 0;

}

ll gcd(ll a, ll b)

{

ll temp;

while (b)

{

temp = b;

b = a % b;

a = temp;

}

return a;

}

**快速幂模板；**

bool IsPn(ll a, ll p, ll mod)

{

ll res = 1, b = a;

while (p)

{

if (p & 1)

{

res = (res \* a) % mod;

}

a = (a \* a) % mod;

p >>= 1;

}

res %= mod;

if (res == b)

return true;

return false;

}

快速乘法；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cmath>

using namespace std;

long long MUL(long long a, long long b, long long c);

int main()

{

int i, j, k, n, m, u;

long long ar[100010], res, S, T, temp;

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

{

for (m = 1; m <= n; m++)

{

scanf("%d", &k);

S = 0;

T = 1;

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

{

scanf("%lld", &ar[u]);

S += ar[u];

}

res = ar[1];

for (i = 2; i <= k; i++)

{

res = MUL(res, ar[i], S);

}

printf("%lld\n", res);

}

}

return 0;

}

long long MUL(long long a, long long b, long long c)

{

long long res = 0, temp, i, k;

while (b)

{

if (b % 2)

{

res = (res + a) % c;

}

a = (a + a) % c;

b /= 2;

}

return res;

}

矩阵快速幂；

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cctype>

#include <cmath>

using namespace std;

const int MAX = 100000000;

struct node

{

double ar[205][205];

};

struct node temp, res, start;

void multiply(node &a, node &b, int c);

int main()

{

int i, j, k, u, n, m, sa[205], a, b, x, y, s;

double p, t, ans;

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

{

memset(start.ar, 0.0, sizeof(start.ar));

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

{

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

}

scanf("%d", &k);

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

{

scanf("%d %d %lf", &x, &y, &p);

start.ar[x+1][y+1] += p;

}

t = 0.0;

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

{

t = 0.0;

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

{

t += start.ar[i][j];

}

start.ar[i][i] = 1.0 - t;

}

memset(res.ar, 0.0, sizeof(res.ar));

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

{

res.ar[i][i] = 1.0;

}

while (m)

{

if (m % 2)

{

multiply(res, start, n);

}

multiply(start, start, n);

m /= 2;

}

ans = 0.0;

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

{

ans += sa[i] \* res.ar[i][n];

}

printf("%.0lf\n", ans);

}

return 0;

}

void multiply(node &a, node &b, int c)

{

int i, j, k;

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

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

{

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

{

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

{

temp.ar[i][j] += a.ar[i][k] \* b.ar[k][j];

}

}

}

a = temp;

}

//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;

}

s\_gets：

#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;

}

**大数运算（高精度）：**

**高精度加**：

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <cmath>

#include <cctype>

using namespace std;

const int LEN = 10000;

char t[LEN + 10];

char \* G\_plus(char \*a, char \*b);

int main()

{

int i, j, k, n, m, doudou;

char a[LEN], b[LEN];

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

{

doudou = 0;

for (m = 1; m <= n; m++)

{

scanf("%s %s", a, b);

if (doudou)

putchar('\n');

doudou = 1;

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

printf("%s + %s = %s\n", a, b, G\_plus(a, b));

}

}

return 0;

}

char \* G\_plus(char \*a, char \*b)

{

int L1, L2, L3, i, j, k, u;

char temp[LEN + 10], c[LEN + 10];

memset(c, 48, sizeof(c));

memset(t, 48, sizeof(t));

//puts(a);

//puts(b);

L1 = strlen(a);

L2 = strlen(b);

//不管谁长谁短一起搞，一个没了只加入另一个就行了，两个都没了结束；

for (i = L1 - 1, j = L2 - 1, k = 0; i >= 0 || j >= 0; i--, j--, k++)

{

if (j < 0)

{

u = c[k] + a[i] - '0' - '0';

if (u >= 10)

{

c[k + 1] += 1;

c[k] = u % 10 + '0';

}

else

c[k] = u + '0';

}

if (i < 0)

{

u = c[k] + b[j] - '0' - '0';

if (u >= 10)

{

c[k + 1] += 1;

c[k] = u % 10 + '0';

}

else

c[k] = u + '0';

}

if (i >= 0 && j >= 0)

{

u = c[k] + a[i] + b[j] - '0' - '0' - '0';

//printf("%d\n", u);

if (u >= 10)

{

c[k + 1] += 1;

c[k] = u % 10 + '0';

}

else

{

c[k] = u + '0';

}

}

}

c[k+1] = '\0';

i = strlen(c) - 1;

//去掉前导0；

while (c[i] == '0')

i--;

//如果结果是0，输出“0”

if (i == -1)

return "0";

//把反向改为正向输出

for (j = 0; i >= 0; i--, j++)

t[j] = c[i];

t[j] = '\0';

return t;

}

**高精度减：**

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <cmath>

#include <cctype>

using namespace std;

const int LEN = 10010;

char p[LEN], q[LEN], c[LEN + 10];

char ta[LEN], tb[LEN];

char t[LEN + 10];

char \* G\_minus(char \*a, char \*b);

char \* G\_plus(char \*a, char \*b);

void Print\_G\_minus(char \*a, char \*b);

int main()

{

int i, j, k, n, m;

printf("请输入测试组数：\n");

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

{

for (m = 1; m <= n; m++)

{

printf("请输入两个数做减法：（例如 a b）\n");

scanf("%s %s", p, q);

Print\_G\_minus(p, q);

}

}

return 0;

}

void Print\_G\_minus(char \*a, char \*b)

{

bool flag1 = false, flag2 = false;

if (a[0] == '-')

flag1 = true;

if (b[0] == '-')

flag2 = true;

if (!flag1 && !flag2)

{

printf("%s - %s = %s\n", a, b, G\_minus(a, b));

}

else if (flag1 && !flag2)

{

printf("%s - %s = -%s\n", a, b, G\_plus(a + 1, b));

}

else if (!flag1 && flag2)

{

printf("%s - %s = %s\n", a, b, G\_plus(a, b + 1));

}

else

{

printf("%s - %s = %s\n", a, b, G\_minus(b + 1, a + 1));

}

}

char \* G\_minus(char \*a, char \*b)

{

int L1, L2, L3, i, j, k, u;

bool flag = false;

memset(c, 48, sizeof(c));

memset(t, 48, sizeof(t));

L1 = strlen(a);

L2 = strlen(b);

if (L1 < L2 || (L1 == L2 && strcmp(a, b) < 0))

{

strcpy(ta, b);

strcpy(tb, a);

flag = true;

}

else

{

strcpy(ta, a);

strcpy(tb, b);

}

L1 = strlen(ta);

L2 = strlen(tb);

for (i = L1 - 1, j = L2 - 1, k = 0; i >= 0 || j >= 0; i--, j--, k++)

{

if (j < 0)

{

u = c[k] + ta[i] - '0' - '0';

if (u < 0)

{

c[k + 1] -= 1;

c[k] = u + 10 + '0';

}

else

c[k] = u + '0';

}

if (i < 0)

{

u = c[k] + tb[j] - '0' - '0';

if (u < 0)

{

c[k + 1] -= 1;

c[k] = u + 10 + '0';

}

else

c[k] = u + '0';

}

if (i >= 0 && j >= 0)

{

u = c[k] + ta[i] - tb[j] + '0' - '0' - '0';

//printf("%d\n", u);

if (u < 0)

{

c[k + 1] -= 1;

c[k] = u + 10 + '0';

}

else

{

c[k] = u + '0';

}

}

}

c[k + 1] = '\0';

i = strlen(c) - 1;

while (c[i] == '0')

i--;

if (i == -1)

return "0";

if (flag)

{

t[0] = '-';

for (j = 1; i >= 0; i--, j++)

t[j] = c[i];

t[j] = '\0';

return t;

}

else

{

for (j = 0; i >= 0; i--, j++)

t[j] = c[i];

t[j] = '\0';

return t;

}

}

char \* G\_plus(char \*a, char \*b)

{

int L1, L2, L3, i, j, k, u;

char temp[LEN + 10], c[LEN + 10];

memset(c, 48, sizeof(c));

memset(t, 48, sizeof(t));

//puts(a);

//puts(b);

L1 = strlen(a);

L2 = strlen(b);

//不管谁长谁短一起搞，一个没了只加入另一个就行了，两个都没了结束；

for (i = L1 - 1, j = L2 - 1, k = 0; i >= 0 || j >= 0; i--, j--, k++)

{

if (j < 0)

{

u = c[k] + a[i] - '0' - '0';

if (u >= 10)

{

c[k + 1] += 1;

c[k] = u % 10 + '0';

}

else

c[k] = u + '0';

}

if (i < 0)

{

u = c[k] + b[j] - '0' - '0';

if (u >= 10)

{

c[k + 1] += 1;

c[k] = u % 10 + '0';

}

else

c[k] = u + '0';

}

if (i >= 0 && j >= 0)

{

u = c[k] + a[i] + b[j] - '0' - '0' - '0';

//printf("%d\n", u);

if (u >= 10)

{

c[k + 1] += 1;

c[k] = u % 10 + '0';

}

else

{

c[k] = u + '0';

}

}

}

c[k + 1] = '\0';

i = strlen(c) - 1;

//去掉前导0；

while (c[i] == '0')

i--;

//如果结果是0，输出“0”

if (i == -1)

return "0";

//把反向改为正向输出

for (j = 0; i >= 0; i--, j++)

t[j] = c[i];

t[j] = '\0';

return t;

}

**大数：**

#include <iostream>

#include <string>

using namespace std;

// 实现大数相加 结果存放在num中

void bigIntergerAdd(string &num, string add) {

int goBit = 0; // 存放进位

// 先交换下顺序 加数的位数要比较少

if (num.length() < add.length()) {

string tmp = num;

num = add;

add = tmp;

}

string tmp(num.length() - add.length(), '0');

add = tmp + add;

// 利用string的+号特性 不采用逆序相加法

int len1 = num.length(), len2 = add.length();

for (int i = len1 - 1; i >= 0; --i) {

int tmp = ((num[i] - '0') + (add[i] - '0') + goBit);

num[i] = tmp % 10 + '0';

goBit = tmp / 10;

}

// 特殊情况处理

if (goBit != 0)

num.insert(0, string(1, (char)goBit + '0'));

}

int main()

{

int i, j;

string s1, s2;

cin >> s1 >> s2;

bigIntergerAdd(s1, s2);

for (i = 0; i <= s1.size() - 1; i++)

{

if (s1[i] != '0')

{

break;

}

}

cout << &s1[i] << endl;

cout << s1 << endl;

return 0;

}

#include <iostream>

#include <string>

using namespace std;

// 大数相乘

string bigIntegerPlus(string res, string plusN) {

string ret;

if (res.length()< plusN.length()) {

string tmp = res;

res = plusN;

plusN = tmp;

}

int len1 = res.length(), len2 = plusN.length();

for (int i = len2-1; i>=0; --i ) {

string tmp(len1, '0'); // 存放相乘的中间结果

int goBit =0;

for (int j= len1-1; j >=0; --j) {

int mid = (res[j] -'0') \* (plusN[i] -'0') + goBit;

tmp[j] = mid%10 + '0';

goBit = mid /10;

}

if (goBit != 0)

tmp.insert(0, string(1,goBit +'0'));

for (int m=0; m< len2 -1-i; ++m)

tmp.push\_back('0'); // 补位

// 相乘后就相加 大数相加

if (i == len2-1)

ret = tmp;

else {

int goBit2 =0;

string s(tmp.length() - ret.length() ,'0');

ret = s + ret;

for (int m = tmp.length()-1; m>=0; --m) {

int mid = (tmp[m] -'0')+(ret[m] - '0') + goBit2;

ret[m] = mid %10 +'0';

goBit2 = mid/ 10;

}

if (goBit2 != 0)

ret.insert(0, string(1,goBit +'0'));

}

}

// 去掉前导0

while (ret.length() >1 && ret[0] == '0')

ret.erase(0,1);

return ret;

}

int main()

{

string s1, s2, res;

cin >> s1 >> s2;

res = bigIntegerPlus(s1, s2);

cout << res << endl;

return 0;

}

//00000000000000000001546834444444443412332131351313113131313

//00000000000000000000000000000546468468497978945613321515565561233

//845296249875477088332644014107880443737808280124537602414883530808871188929

#include <iostream>

#include <cstring>

using namespace std;

void bigDivision(char \*src, int num, char sign) {

int i, j;

long long rem = 0; // 存放新余数

long long prem = 0; // 原余数

char res[10000] = "";

bool flag = true;

int k = 0;

for (i = 0; i< strlen(src); ++i) {

rem = prem \* 10/\*向后退一位\*/ + src[i] - '0';

if (rem / num >0 || rem == 0) {

res[k++] = rem / num + '0';

prem = rem %num;

flag = false;

}

else {

prem = rem;

if (!flag)

res[k++] = '0';

}

}

if (sign == '%') {

cout << prem << endl;

return;

}

for (i = 0; i <= strlen(res) - 1; i++)

{

if (res[i] != '0')

{

break;

}

}

for (j = i; j< k; ++j)

cout << res[j];

cout << endl;

}

int main(int argc, char\*\* argv)

{

int i, j;

char src[10000] = "";

int num;

char sign;

while (scanf("%s%d", src, &num) != EOF)

{

bigDivision(src, num, '/');

bigDivision(src, num, '%');

}

return 0;

}

#include <iostream>

#include <string>

using namespace std;

// 求幂 思路: 先变成整数相乘 然后根据小数的位数 结合幂 算出小数点该结果字符串的位置 即可

string bigIntegerPlus(string src, string num) {

string tmp = src;

for (int i =num.length() -1; i >= 0 ; --i) {

string mid(tmp.length(),'0');

int goBit =0;

for (int j = tmp.length()-1; j >= 0; --j) {

int tm = goBit + (tmp[j] -'0')\* (num[i] - '0');

mid[j] = tm% 10 +'0';

goBit = tm /10;

}

for (int q = num.length()-1; q> i; --q)

mid.push\_back('0');

if (goBit != 0)

mid.insert(0, string(1, (char)goBit +'0'));

// 加法运算

if (i == num.length()-1)

src = mid;

else {

goBit =0;

string s(mid.length() - src.length(), '0');

src = s + src;

for (int j = mid.length()-1; j>=0; --j) {

int tm = (mid[j] - '0') +(src[j] - '0') + goBit;

src[j] = tm %10 + '0';

goBit = tm /10;

}

if (goBit !=0)

src.insert(0, string(1, (char)goBit +'0'));

}

}

return src;

}

int main(int argc, char\*\* argv) {

string str;

while ( getline(cin, str)) {

// 分割出待积数 和 幂 以及小数点位置

int i =0;

int index = 0;// 小数位置

int count = 0;//幂次数

string num;

while ( i< str.length()) {

if ( str[i] != ' ') {

if (str[i] == '.')

index = i;

else

num.push\_back(str[i]);

++i;

continue;

}

while ( !isdigit(str[i]))

++i;

if (i + 1 == str.length())

count = str[i] - '0';

else

count = (str[i] - '0') \* 10 + str[i+1] - '0';

break;

}

index = num.length() - index;

string res = num;

for (int i =0; i< count-1; ++i) {

res = bigIntegerPlus( res, num);

}

index = index \* count;

res.insert(res.length() - index, string("."));

while (res.length() >1 && res[0] == '0')

res.erase(0, 1);

for (int i =res.length()-1; i>=0; --i) {

if (res[i] == '0' )

res.erase(i, i+1);

else

break;

}

cout<< res<< endl;

}

return 0;

}