**最小表示法**

int get\_min(char \*s) {

int i = 0, j = 1, k = 0;

while (i < n && j < n) {

for (k = 0; k < n && s[i + k] == s[j + k]; k++);

if (k == n) break;

if (s[i + k] > s[j + k]) {

i += k + 1;

if (i == j) i++;

} else {

j += k + 1;

if (i == j) j++;

}

}

return min(i, j);

}

**数字整除**

n 能被 3 整除 == 各位数字之和能被 3 整除

n 能被 9 整除 == 各位数字之和能被 9 整除

n 能被 5 整除 == 最后一位能被 5 整除

n 能被 2 整除 == 最后一位能被 2 整除

n 能被 4 整除 == 最后俩位能被整除

n 能被 8 整除 == 最后三位能被整除

**差分约束**

int main() {

//最小值用最长路

// a >= b + c

add(b, a, c);

// 最大值用最短路

// a <= b + c

add(b, a, c);

}

**三分**

while (l + 2 < r) {// 求最小值

int m1 = l + (r - l) / 3, m2 = l + (r - l) / 3 \* 2;

if (f(m1) > f(m2)) l = m1;

else r = m2;

}

while (l + 2 < r) { // 求最大值

int m1 = l + (r - l) / 3, m2 = l + (r - l) / 3 \* 2;

if (f(m1) < f(m2)) l = m1;

else r = m2;

}

**Int128**

inline void read(int &n){

int x=0,f=1;

char ch=getchar();

while(ch<'0'||ch>'9'){

if(ch=='-') f=-1;

ch=getchar();

}

while(ch>='0'&&ch<='9'){

x=(x<<1)+(x<<3)+(ch^48);

ch=getchar();

}

n=x\*f;

}

inline void print(int n){

if(n<0){

putchar('-');

n\*=-1;

}

if(n>9) print(n/10);

putchar(n % 10 + '0');

}

**树的直径**

int bfs(int u) {

memset(d, -1, sizeof d);

d[u] = 0;

int hh = 0, tt = 0;

q[0] = u;

while(hh <= tt) {

int t = q[hh++];

for(int i = h[t]; i != -1; i = ne[i]) {

int j = e[i];

if(d[j] == -1) {

d[j] = d[t] + 1;

pre[j] = i;

q[++tt] = j;

}

}

}

int p = u;

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

if(d[i] > d[p])

p = i;

return p;

}

int dfs(int u, int father) {

int dist1 = 0, dist2 = 0;

for(int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if(j == father) continue;

int dist = dfs(j, u) + w[i];

if(dist >= dist1) dist2 = dist1, dist1 = dist;

else if(dist > dist2) dist2 = dist;

}

ans = max(ans, dist1 + dist2);

return dist1;

}

void update(int ed, int st) { //从 ed 回推到 st，并将路上经过的边都取反

while(ed != st) {

w[pre[ed]] = -1; //正向边取反

w[pre[ed] ^ 1] = -1; //反向边取反

ed = e[pre[ed] ^ 1]; //退到这条边的入点，即上一步走到的点

}

}

**倍增求LCA**

int q[N], depth[N], fa[N][16];

void bfs(int root) {

memset(depth, 0x3f, sizeof depth);

int head = 0, tail = -1;

q[ ++ tail] = root, depth[0] = 0, depth[root] = 1;

while (head <= tail) {

int t = q[head ++ ];

for (int i = h[t]; ~i; i = ne[i])

{

int j = e[i];

if (depth[j] > depth[t] + 1)

{

depth[j] = depth[t] + 1;

q[ ++ tail] = j;

fa[j][0] = t;

for (int k = 1; k <= 15; k ++ )

fa[j][k] = fa[fa[j][k - 1]][k - 1];

}

}

}

}

int lca(int a, int b) {

if (depth[a] < depth[b]) swap(a, b);

for (int k = 15; k >= 0; k -- )

if (depth[fa[a][k]] >= depth[b])

a = fa[a][k];

if (a == b) return a;

for (int k = 15; k >= 0; k -- )

if (fa[a][k] != fa[b][k])

{

a = fa[a][k];

b = fa[b][k];

}

return fa[a][0];

}

**LCA次小生成树**

int dep[N], fa[N][17], d1[N][17], d2[N][17];

void bfs() {

memset(dep, 0x3f, sizeof dep);

dep[0] = 0;

dep[1] = 1;

queue<int> q;

q.push(1);

while (q.size()) {

int t = q.front();

q.pop();

for (int i = h[t]; i != -1; i = ne[i]) {

int j = e[i];

if (dep[j] > dep[t] + 1) {

q.push(j);

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

fa[j][0] = t;

d1[j][0] = w[i], d2[j][0] = -INF;

for (int k = 1; k <= 16; k++) {

fa[j][k] = fa[fa[j][k - 1]][k - 1];

int tmp[4] = {d1[j][k - 1], d2[j][k - 1], d1[fa[j][k - 1]][k - 1], d2[fa[j][k - 1]][k - 1]};

for (int u = 0; u < 4; u++) {

if (tmp[u] > d1[j][k]) {

d2[j][k] = d1[j][k];

d1[j][k] = tmp[u];

} else if (d1[j][k] != tmp[u] && tmp[u] > d2[j][k]) {

d2[j][k] = tmp[u];

}

}

}

}

}

}

}

int stk[N \* 2];

ll lca(int a, int b, int c) {

if (dep[a] > dep[b]) swap(a, b);

int top = 0;

for (int k = 16; k >= 0; k--) {

if (dep[fa[b][k]] >= dep[a]) {

stk[++top] = d1[b][k], stk[++top] = d2[b][k];

b = fa[b][k];

}

}

if (a != b) {

for (int k = 16; k >= 0; k--) {

if (fa[a][k] != fa[b][k]) {

stk[++top] = d1[a][k], stk[++top] = d1[b][k];

stk[++top] = d2[a][k], stk[++top] = d2[b][k];

a = fa[a][k];

b = fa[b][k];

}

}

stk[++top] = d1[a][0], stk[++top] = d1[b][0];

stk[++top] = d2[a][0], stk[++top] = d2[b][0];

}

ll t1 = -INF, t2 = -INF;

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

if (stk[i] > t1) {

t2 = t1;

t1 = stk[i];

} else if (t1 != stk[i] && stk[i] > t2) {

t2 = stk[i];

}

}

if (c > t1) return sum + c - t1;

if (c > t2) return sum + c - t2;

return INF;

}

**数位dp**

ll l, r;

ll f[11][N][M];

vector<int> ve;

ll dp(int u, int state, int pre, int limit) {

if (u < 0) {

return !state;

}

if (!limit && !pre && f[k][u][state] != -1) return f[k][u][state];

int up = limit ? ve[u] : k - 1;

ll ans = 0;

for (int i = 0; i <= up; i++) {

int t = state;

if (!pre || i != 0) t = t ^ (1 << i);

ans += dp(u - 1, t, pre && i == 0, limit && i == up);

}

if (!limit && !pre) f[k][u][state] = ans;

return ans;

}

ll calc(ll x) {

ve.clear();

while (x) {

ve.push\_back(x % k);

x /= k;

}

return dp(ve.size() - 1, 0, 1, 1);

}

void solve() {

cin >> k >> l >> r;

cout << calc(r) - calc(l - 1) << endl;

}

**Int128**

inline void read(int &n){

int x=0,f=1;

char ch=getchar();

while(ch<'0'||ch>'9'){

if(ch=='-') f=-1;

ch=getchar();

}

while(ch>='0'&&ch<='9'){

x=(x<<1)+(x<<3)+(ch^48);

ch=getchar();

}

n=x\*f;

}

inline void print(int n){

if(n<0){

putchar('-');

n\*=-1;

}

if(n>9) print(n/10);

putchar(n % 10 + '0');

}

**DFS序求LCA**

typedef long long ll;

typedef pair<int, int> PII;

const int N = 500010, M = 2000010, INF = 1e9 + 7, Hash = 13331, MOD = 998244353;

int T, n, m, k, s;

vector<int> G[N];

vector<PII> f[M];

int dfn[N], d[N], p[N][20], tot;

void dfs1(int u, int fa) {

d[u] = d[fa] + 1;

dfn[u] = ++tot;

p[tot][0] = fa;

for (int v : G[u]) {

if (v == fa) continue;

dfs1(v, u);

}

}

int RMQ(int l, int r) {

if (l > r) swap(l, r);

int k = log(r - l + 1) / log(2);

int x = p[l][k], y = p[r - (1 << k) + 1][k];

return d[x] <= d[y] ? x : y;

}

int lca(int x, int y) {

if (x == y) return x;

if (dfn[x] > dfn[y]) swap(x, y);

return RMQ(dfn[x] + 1, dfn[y]);

}

void init() {

dfs1(s, 0);

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

for (int i = 1; i + (1 << j) - 1 <= n; i++) {

int x = p[i][j - 1], y = p[i + (1 << j - 1)][j - 1];

p[i][j] = d[x] <= d[y] ? x : y;

}

}

}

int main() {

return 0;

}

**树刨求LCA**

typedef long long ll;

typedef int PII;

const int N = 500010, M = 2 \* N, MOD = 1e9 + 7;

int n, m, k, tot, s;

vector<int> G[N];

int d[N], fa[N];

int hs[N], top[N], sz[N];

void dfs1(int u, int fr) {

sz[u] = 1;

d[u] = d[fr] + 1;

fa[u] = fr;

for (auto v: G[u]) {

if (v == fr) continue;

dfs1(v, u);

sz[u] += sz[v];

if (hs[u] == 0 || sz[hs[u]] < sz[v]) {

hs[u] = v;

}

}

}

void dfs2(int u, int t) {

top[u] = t;

if (hs[u]) {

dfs2(hs[u], t);

}

for (auto v : G[u]) {

if (v == fa[u] || v == hs[u]) continue;

dfs2(v, v);

}

}

int lca(int x, int y) {

while (top[x] != top[y]) {

if (d[top[x]] > d[top[y]]) {

x = fa[top[x]];

} else {

y = fa[top[y]];

}

}

return d[x] <= d[y] ? x : y;

}

void init() {

dfs1(1, 0);

dfs2(1, 1);

}

int main() {

return 0;

}