南京工業大學

NANJING TECH UNIVERSITY

ICPC Template Manual

作者: 贺梦杰

June 14, 2019

Contents

Chapter 1

基础

1.1 IO 优化

```
ios::sync_with_stdio(false);
cin.tie(0);
```

Chapter 2

搜索

Chapter 3

动态规划

Chapter 4

字符串

4.1 KMP

```
int *Get_next(string str)
{
    int *ptr = new int[str.length()];
    //申请next数组
    ptr[0] = 0;             //首位next值为0
    int i = 1;              //初始化
    int j = 0;              //初始化
    int len = str.length(); //模式串长度
    while (i < len)
    {
        if (str[i] == str[j])
        {
            ptr[i] = j + 1;
            j++;
            i++; //确定前缀后缀相同的长度
        }
        else
        {
            //不同时
            if (j != 0)
                j = ptr[j - 1]; //j回到前一个字符的next值位置
            else
            {
                ptr[i] = 0; //回到模式串的第一个字符
                i++;
            }
        }
    }
    return ptr;
}
int KMP(string s, string p)
{
    int *next = Get_next(p);
    //获得next数组
    int i = 0;
    int j = 0;
    int len = s.length();
    while (i < len)
    {
        if (s[i] == p[j])
        {
            i++;
            j++; //匹配
            if (j >= p.length())
                return i - j;
        }
        else
        {
            //字符不相同回到前一个字符的next值位置
            if (j != 0)
                j = next[j - 1];
            else
                i++;
        }
    }
    return -1;
}
```

来源：*https://www.bilibili.com/video/av47471886?from=search&seid=4651914725266859344*

Chapter 5

数据结构

5.1 并查集

```
#define MAX 1010
struct node
{
    int par;
    //int rank;
    //路径压缩后 rank=1或2 rank失去了意义
    int data;
};
node ns[MAX];
void Init()
{
    for (int i = 1; i < MAX; i++)
    {
        ns[i].par = i;
    }
}
int Find(int i)
{
    if (ns[i].par == i)
    {
        //返回根结点
        return i;
    }
    ns[i].par = Find(ns[i].par);
    //路径压缩
    return ns[i].par;
}
void Union(int i, int j)
{
    int pi = Find(i);
    int pj = Find(j);
    if (pi != pj)
    {
        ns[pi].par = pj;
    }
}
```

5.2 线段树

5.2.1 基础操作

```
const int N = 1e5 + 10;
#define ls(a) (a << 1)
#define rs(a) (a << 1 | 1)
struct node
{
    int val;
    int lazy;
};
node tree[N << 2];
int a[N];
void PushUp(int rt)
{
    tree[rt].val = tree[ls(rt)].val + tree[rs(rt)];
}
void PushDown(int ls, int rs, int rt)
{
    tree[ls(rt)].val += ls * tree[rt].lazy;
    tree[rs(rt)].val += rs * tree[rt].lazy;
    tree[ls(rt)].lazy += tree[rt].lazy;
    tree[rs(rt)].lazy += tree[rt].lazy;
    tree[rt].lazy = 0;
}
void Build(int left, int right, int rt)
{
    if (left == right)
    {
        tree[rt].val = a[left];
        return;
    }
    int mid = (left + right) >> 1;
    Build(left, mid, ls(rt));
    Build(mid + 1, right, rs(rt));
    PushUp(rt);
    //向上更新
}
```

5.2.2 单点更新

```
void Update(int left, int right, int rt, int pos, int val)
{
    if (left == right && left == pos)
    {
        tree[rt].val += val;
        return;
    }
    int mid = (left + right) >> 1;
    if (tree[rt].lazy)
    {
        PushDown(mid - left + 1, right - mid, rt);
    }
    if (mid >= pos)
        Update(left, mid, ls(rt), pos, val);
    else if (pos > mid)
        Update(mid + 1, right, rs(rt), pos, val);
    PushUp(rt);
}
```

例题：*https://www.luogu.org/problemnew/show/P3372*

5.2.3 区间更新

```
void Update(int left, int right, int rt, int s, int t, int val)
{
    if (left >= s && right <= t)
    {
        tree[rt].val += (right - left + 1) * val;
        tree[rt].lazy += val;
        return;
    }
    int mid = (left + right) >> 1;
    if (tree[rt].lazy)
    {
        PushDown(mid - left + 1, right - mid, rt);
    }
    if (mid < s)
        Update(mid + 1, right, rs(rt), s, t, val);
    else if (mid >= t)
        Update(left, mid, ls(rt), s, t, val);
    else
    {
        Update(left, mid, ls(rt), s, t, val);
        Update(mid + 1, right, rs(rt), s, t, val);
    }
    PushUp(rt);
}
```

5.2.4 区间查询

```
void Query(int left, int right, int s, int t, int rt)
{
    if (left >= s && right <= t)
    {
        return tree[rt].val;
    }
    int mid = (left + right) >> 1;
    if (tree[rt].lazy)
        PushDown(mid - left + 1, right - mid, rt);
    long long sum = 0;
    if (mid < s)
        sum += Query(mid + 1, right, rs(rt), s, t, val);
    else if (mid >= t)
        sum += Query(left, mid, ls(rt), s, t, val);
    else
    {
        sum += Query(left, mid, ls(rt), s, t, val);
        sum += Query(mid + 1, right, rs(rt), s, t, val);
    }
    return sum;
}
```

例题：*https://www.luogu.org/problemnew/show/P3373*

5.3 树状数组

推荐阅读：*https://www.cnblogs.com/RabbitHu/p/BIT.html*

5.3.1 单点修改，区间查询

```
#define N 1000100
long long c[N];
int n,q;
int lowbit(int x)
{
    return x&(-x);
}
void change(int x,int v)
{
    while(x<=n)
    {
        c[x]+=v;
        x+=lowbit(x);
    }
}
long long getsum(int x)
{
    long long ans=0;
    while(x>=1)
    {
        ans+=c[x];
        x-=lowbit(x);
    }
    return ans;
}
```

例题：*https://loj.ac/problem/130*

5.3.2 区间修改，单点查询

引入差分数组来解决树状数组的区间更新

```
//初始化
change(i,cur-pre);
//区间修改
change(l,x);
change(r+1,-x);
//单点查询
```

```
getsum(x)
```

例题：*https://loj.ac/problem/131*

5.3.3 区间修改，区间查询

```
//初始化
change(c1,i,cur-pre);
change(c2,i,i*(cur-pre));
//为什么这么写？ 你需要写一下前缀和的表达式
//区间修改
change(c1,l,x);
change(c2,l,l*x);
change(c1,r+1,-x);
change(c2,r+1,-(r+1)*x);
//区间查询
temp1=l*getsum(c1,l-1)-getsum(c2,l-1);
temp2=(r+1)*getsum(c1,r)-getsum(c2,r);
ans=temp2-temp1
```

例题：*https://loj.ac/problem/132*

5.4 二维树状数组

5.4.1 单点修改，区间查询

```
#define N 5050
long long tree[N][N];
long long n,m;
long long lowbit(long long x)
{
    return x&(-x);
}
void change(long long x,long long y,long long val)
{
    long long init_y=y;
    //这里注意n,m的限制
    while(x<=n)
    {
        y=init_y;
        while(y<=m)
        {
            tree[x][y]+=val;
            y+=lowbit(y);
        }
        x+=lowbit(x);
    }
}
long long getsum(long long x,long long y)
{
    long long ans=0;
    long long init_y=y;
    while(x>=1)
    {
        y=init_y;
        while(y>=1)
        {
            ans+=tree[x][y];
            y-=lowbit(y);
        }
        x-=lowbit(x);
    }
    //这里画图理解
    return ans;
}
//初始化
change(x,y,k);
//二维前缀和
ans = getsum(c,d)+getsum(a-1,b-1)-getsum(a-1,d)-getsum(c,b-1);
```

例题：*https://loj.ac/problem/133*

5.4.2 区间修改，区间查询

```
#define N 2050
long long t1[N][N];
long long t2[N][N];
long long t3[N][N];
long long t4[N][N];
long long n,m;
long long lowbit(long long x)
{
    return x&(-x);
}
long long getsum(long long x,long long y)
{
    long long ans=0;
    long long init_y=y;
    long long init_x=x;
    while(x>=1)
    {
        y=init_y;
        while(y>=1)
        {
            ans+=(init_x+1)*(init_y+1)*t1[x][y];
            ans-=(init_y+1)*t2[x][y];
            ans-=(init_x+1)*t3[x][y];
            ans+=t4[x][y];
            y-=lowbit(y);
        }
        x-=lowbit(x);
    }
    return ans;
}
void change(long long x,long long y,long long val)
{
    long long init_x=x;
    long long init_y=y;
    while(x<=n)
    {
        y=init_y;
        while(y<=m)
        {
            t1[x][y]+=val;
            t2[x][y]+=init_x*val;
            t3[x][y]+=init_y*val;
            t4[x][y]+=init_x*init_y*val;
            y+=lowbit(y);
        }
        x+=lowbit(x);
    }
}
//区间修改
change(c+1,d+1,x);
change(a,b,x);
change(a,d+1,-x);
change(c+1,b,-x);
//区间查询
ans=getsum(c,d)+getsum(a-1,b-1)-getsum(c,b-1)-getsum(a-1,d);
```

例题：*https://loj.ac/problem/135*

Chapter 6

图论

6.1 最小生成树

6.1.1 Prim

```
#define inf 0x3f3f3f3f
const int N = 2e5 + 20;
struct node
{
    long long u, v, w;
    node(int uu, int vv, int ww) : u(uu), v(vv), w(ww)
    {
    }
    bool operator<(node n) const
    {
        return w > n.w;
    }
};
long long n, m;
priority_queue<node> q;
vector<pair<long long, long long>> G[N];
bool vis[N];
long long Prim()
{
    long long ans = 0;
    for (auto ele : G[1])
    {
        q.push(node(1, ele.first, ele.second));
    }
    memset(vis, 0, sizeof(vis));
    vis[1] = 1;
    int t = n - 1;
    while (t--)
    {
        node top = q.top();
        q.pop();
        while (vis[top.v])
        {
            top = q.top();
            q.pop();
        }
        ans += top.w;
        cout<<ans<<endl;
        vis[top.v] = 1;
        for (auto ele : G[top.v])
        {
            q.push(node(top.v, ele.first, ele.second));
        }
    }
    return ans;
}
```

6.1.2 Kruskal

基于并查集

```
const long long MAXN = 2e5 + 20;
struct Edge
{
    long long u, v, w;
    bool operator<(Edge e) const
    {
        return w > e.w;
    }
    Edge(long long uu, long long vv, long long ww) : u(uu), v(vv
    ), w(ww)
    {
    }
};
long long fa[MAXN];
long long Find(long long x)
{
    if (fa[x] == -1)
        return x;
    else
        return fa[x] = Find(fa[x]);
}
priority_queue<Edge> q;
long long n; //点
long long m; //边
long long Kruskal()
{
    memset(fa, -1, sizeof(fa));
    long long cnt = 0;
    long long ans = 0;
    long long fu, fv;
    while (!q.empty())
    {
        Edge now = q.top();
        q.pop();
        fu = Find(now.u);
        fv = Find(now.v);
        if (fu != fv)
        {
            //cout<<"add:"<<now.u<<" "<<now.v<<" "<<now.w<<endl;
            ans += now.w;
            fa[fu] = fv;
            cnt++;
        }
        if (cnt == n - 1)
            break;
    }
    if (cnt < n - 1)
        return -1;
    return ans;
}
```

例题：*https://loj.ac/problem/123*

6.1.3 次小生成树

倍增 LCA 维护最小生成树
代码过长……
例题：*https://loj.ac/problem/10133*

6.2 最近公共祖先

6.2.1 倍增

```cpp
#define N 500050
int depth[N];
int fa[N][20];
vector<int> v[N];
int lg[N]; //log2(n) floor
int n, m, s;
void init()
{
    lg[0] = -1;
    //floor
    for (int i = 1; i <= n; i++)
    {
        lg[i] = lg[i >> 1] + 1;
    }
}
void DFS(int cur, int pre)
{
    depth[cur] = depth[pre] + 1;
    fa[cur][0] = pre;
    for (int i = 1; (1 << i) <= depth[cur]; i++)
    {
        fa[cur][i] = fa[fa[cur][i - 1]][i - 1];
    }
    for (auto ele : v[cur])
    {
        if (ele != pre)
        {
            DFS(ele, cur);
        }
    }
}
int LCA(int a, int b)
{
    // assume depth[a]>=depth[b]
    if (depth[a] < depth[b])
        swap(a, b);
    // reset to the same depth
    while (depth[a] > depth[b])
    {
        a = fa[a][lg[depth[a] - depth[b]]];
        //up
    }
    if (a == b)
        return a;
    for (int k = lg[depth[a]] + 1; k >= 0; k--)
    {
        if (fa[a][k] != fa[b][k])
        {
            a = fa[a][k];
            b = fa[b][k];
            //up
        }
    }
    return fa[a][0];
}
```

例题： *https://www.luogu.org/problemnew/show/P3379*

6.3 强连通分量

6.3.1 Tarjan

```cpp
const int N = 10500;
vector<int> G[N];
bool vis[N];
int dfn[N];
int low[N];
int cnt;
stack<int> s;
int n, m;
void init()
{
    memset(vis, 0, sizeof(vis));
    memset(dfn, 0, sizeof(dfn));
    cnt = 0;
}
void DFS(int cur)
{
    dfn[cur] = low[cur] = ++cnt;
    s.push(cur);
    vis[cur] = 1;
    for (auto ele : G[cur])
    {
        if (!dfn[ele])
        {
            DFS(ele);
            low[cur] = min(low[cur], low[ele]);
        }
        else if (vis[ele])
        {
            low[cur] = min(low[cur], dfn[ele]);
        }
    }
    if (dfn[cur] == low[cur])
    {
        while (1)
        {
            int t = s.top();
            s.pop();
            vis[t] = 0;
            if (t == cur)
```

```
                break;
        }
    }
}
```

例题：*https://www.luogu.org/problemnew/show/P2863*

6.3.2 缩点 DAG

利用强连通分量缩点为有向无环图

Tarjan 加入染色

```
if (dfn[cur] == low[cur])
{
    sum++;
    while (1)
    {
        int t = s.top();
        s.pop();
        color[t] = sum;
        pnum[sum]++;
        vis[t] = 0;
        if (t == cur)
            break;
    }
}
```

例题：*https://www.luogu.org/problemnew/show/P2341*

6.4 最短路

6.4.1 Dijkstra

非负权图，单源最短路径
$O((N+M)\log M)$

```
#define N 100100
#define ll long long
#define inf 2147483647
vector<pair<int, ll>> G[N];
struct edge
{
    int to;
    ll weight;
    edge(int i, ll w) : to(i), weight(w){}
    bool operator<(const edge &e) const{return weight > e.weight;}
};
ll dis[N];
bool vis[N];
int n, m, s;
void Dijkstra(int start)
{
    for (int i = 1; i <= n; i++){dis[i] = inf;}
    priority_queue<edge> q;
    q.push(edge(start, 0));
    dis[start] = 0;
    while (!q.empty())
    {
        edge now = q.top();
        q.pop();
        if (!vis[now.to])
        {
            vis[now.to] = 1;
            for (auto ele : G[now.to])
            {
                if (!vis[ele.first] && now.weight + ele.second < dis[ele.first])
                {
                    dis[ele.first] = now.weight + ele.second;
                    q.push(edge(ele.first, dis[ele.first]));
                }
            }
            }
        }
}
```

例题：*https://www.luogu.org/problemnew/show/P3371*
例题：*https://www.luogu.org/problemnew/show/P4779*

6.5 网络流

6.5.1 最大流

6.5.1.1 Edmonds Karp

```
#define ll long long
#define inf 0x3f3f3f3f
const int M = 205;
ll c[M][M];
int pre[M];
ll flow[M];
int n, m;
ll BFS(int s, int t)
{
    queue<int> q;
    memset(pre, -1, sizeof(pre));
    flow[s] = inf;
    pre[s] = 0;
    q.push(s);
    while (!q.empty())
    {
        int u = q.front();
        q.pop();
        if (u == t)
        {
            break;
        }
        for (int v = 1; v <= m; v++)
        {
            if (c[u][v] > 0 && pre[v] == -1&&v!=s)
            {
                pre[v] = u;
                flow[v] = min(c[u][v], flow[u]);
                q.push(v);
            }
        }
    }
    if (pre[t] == -1)
        return -1;
    return flow[t];
}
ll Edmonds_Karp(int s, int t)
```

```
{
    ll inc;
    ll ans = 0;
    int k, last;
    while ((inc = BFS(s, t)) != -1)
    {
        k = t;
        while (k != s)
        {
            last = pre[k];
            c[last][k] -= inc;
            c[k][last] += inc;
            k = last;
        }
        ans += inc;
        //cout<<"cur:"<<inc<<endl;
    }
    return ans;
}
```

例题：*https://www.luogu.org/problemnew/show/P2740*

6.5.1.2 Dinic

多路增广，当前弧优化

```
#define inf 0x3f3f3f3f3f
#define N 1200
#define M 245000
#define ll long long
//反向边的存在 实际边数的两倍
struct Edge
{
    ll to;
    ll w;
    ll next;
};
Edge edge[M];
ll dep[N];
ll head[N];
ll cur[N]; //当前弧优化
ll cnt = 0;
ll n, m;
void Init()
{
    memset(head, -1, sizeof(head));
}
void add(ll u, ll v, ll w)
{
    edge[cnt].to = v;
    edge[cnt].w = w;
    edge[cnt].next = head[u];
    head[u] = cnt++;
}
void Add(ll u, ll v, ll w)
{
    add(u, v, w);
    add(v, u, 0); //反向边
}
ll s, t;
bool BFS()
{
    queue<ll> q;
    memset(dep, -1, sizeof(dep));
    dep[s] = 0;
    q.push(s);
    while (!q.empty())
    {
        ll now = q.front();
        q.pop();
        for (ll i = head[now]; i != -1; i = edge[i].next)
        {
            if (edge[i].w > 0 && dep[edge[i].to] == -1)
            {
                dep[edge[i].to] = dep[now] + 1;
                q.push(edge[i].to);
            }
        }
    }
    if (dep[t] == -1)
        return false; //不存在分层图
    return true;
}
ll DFS(ll now, ll flow)
{
    if (now == t)
        return flow;
    ll used = 0;
    for (ll &i = cur[now]; i != -1; i = edge[i].next)
    {
        //&: 当前弧优化
        if (dep[edge[i].to] == dep[now] + 1 && edge[i].w)
        {
            ll inc = DFS(edge[i].to, min(flow - used, edge[i].w)
    );
            if (inc > 0)
            {
                edge[i].w -= inc;
                edge[i ^ 1].w += inc;
                used += inc;
                if (flow == used)
                    break;
            }
        }
    }
    if (!used)
        dep[now] = -1;
    return used;
}
ll Dinic()
{
    ll ans = 0;
    while (BFS())
    {
        for (ll i = 1; i <= n; i++)
        {
            cur[i] = head[i];
        }
        ans += DFS(s, inf);
    }
    return ans;
}
```

例题：*https://www.luogu.org/problemnew/show/P3376*

6.5.1.3 ISAP

6.5.1.4 HLPP

最高标号预留推进

Chapter 7

数学

7.1 快速幂

7.1.1 递归形式

```
#define mod 1000000007
long long quick_pow(long long a, long long b)
{
    if (b == 0)
        return 1;
    long long temp = quick_pow(a, b >> 1);
    if (b & 1)
        return a % mod * temp % mod * temp % mod;
    else
        return temp % mod * temp % mod; //快速幂
}
```

7.1.2 循环形式

如果递归形式栈溢出可使用循环形式

```
#define mod 1000000007
long long quick_pow(long long x,long long n)
{
    long long ret=1;
    long long temp=x%mod;
    while(n)
    {
        if(n&1)
        {
            ret=(ret*temp)%mod;
        }
        temp=(temp*temp)%mod; //偶次
        n>>=1;
    }
    return ret;//结果
}
```

7.2 矩阵快速幂

```
#define N 105
int m;//矩阵阶
```

```
struct matrix
{
    long long a[N][N];
    matrix(){memset(a, 0, sizeof(a));}
};
matrix matrix_mul(matrix m1, matrix m2)
{
    matrix ans;
    for (int k = 0; k < m; k++)
    {
        for (int i = 0; i < m; i++)
        {
            if (m1.a[i][k]) //剪枝
            {
                for (int j = 0; j < m; j++)
                {
                    if (m2.a[k][j]) //剪枝
                    {
                        ans.a[i][j] = (ans.a[i][j] + (m1.a[i][k] * m2.a[k][j]) % mod) % mod;
                    }
                }
            }
        }
    }
    return ans;
}
matrix quick_pow_matrix(matrix m1, long long k)
{
    matrix ans;//递归写法 可能堆栈溢出
    for (int i = 0; i < m; i++)
        ans.a[i][i] = 1;
    while (k)
    {
        if (k & 1)
            ans = matrix_mul(ans, m1);
        m1 = matrix_mul(m1, m1);
        k >>= 1;
    }
    return ans;
}
```

7.3 组合数取模

```
#define N 100010
#define mod 1000000007
#define ll long long
ll fac[N];//阶乘
ll inv[N];//阶乘逆元
void init()
{
    fac[0]=1;
    for(int i=1;i<N;i++)
    {
        fac[i]=(fac[i-1]*i)%mod;
    }
    inv[N-1]=quick_pow(fac[N-1],mod-2);//费马小定理 求逆元a^(p-1)%p=1 a的逆元为a^(p-2)
    //如果 ax%p=1 ， 那么x的最小正整数解就是 a 模p的逆元
    for(int i=N-2;i>=0;i--)
    {
        inv[i]=inv[i+1]*(i+1)%mod;
    }
}
ll C(ll a,ll b)
{
    if(b>a) return 0;
    if(b==0) return 1;
    return fac[a]*inv[b]%mod*inv[a-b]%mod;
}
```

Chapter 8

计算几何

Chapter 9

其他