

ICPC Template Manual



作者: 贺梦杰

June 14, 2019

Contents

1	基础 1.1	 IO 优化	Ł														3 4
2	搜索	ì															5
3	动态	规划															6
4	字符	串															7
	4.1	KMP.															8
5	数据	结构															10
	5.1	并查集															11
	5.2	线段树															12
		5.2.1	基础操作	· .													12
		5.2.2	单点更新	:													12
		5.2.3	区间更新	:													13
		5.2.4	区间查询	١.													14
	5.3	树状数	组														15
		5.3.1	单点修改	, Þ	区间	查询											15
		5.3.2	区间修改	,自	1点	查询	١.										15
		5.3.3	区间修改		了 区 回 区	 查询	١.										16
	5.4	二维树	状数组 .														16
		5.4.1	单点修改	`, ⊵	区间	查询											16
		5.4.2	区间修改	,													17
6	图论																20
Ū	6.1	· 最小生	成树				_		_	_	_			_	_		21
	···	6.1.1	Prim														21
		6.1.2	Kruskal														22
		6.1.3	次小生成														23
	6.2		共相先	., .													

CONTENTS CONTENTS

9	其他																39
8	计算	几何															38
	7.3	组合数	取模														37
	7.2	矩阵快	速幂														35
		7.1.2	循环形式	•													35
	••-	7.1.1	递归形式														35
7	数学 7.1	快速幂															34 35
			6.5.1.4	HLP	Ρ.						•		•	•			33
			6.5.1.3	ISAI													33
			6.5.1.2	Dini	с.												31
			6.5.1.1	Edm	onds	s Ka	arp										30
		6.5.1	最大流.														30
	6.5	网络流															30
	0.1	6.4.1	Dijkstra												•	•	28
	6.4	0.3.2 最短路	缩点 DA												•	•	28
		6.3.1 6.3.2	Tarjan .												٠	•	26 27
	6.3	强连通															26
		6.2.1	倍增														24

Chapter 1 基础

1.1 IO 优化

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

Chapter 2

搜索

Chapter 3 动态规划

Chapter 4 字符串

4.1 KMP

```
1 int *Get_next(string str)
2 {
3
       int *ptr = new int[str.length()];
 4
       //申请next数组
5
       ptr[0] = 0;
                               //首位next值为0
 6
       int i = 1;
                               //初始化
 7
       int j = 0;
                               //初始化
       int len = str.length(); //模式串长度
8
9
       while (i < len)</pre>
10
11
           if (str[i] == str[j])
12
           {
13
               ptr[i] = j + 1;
14
               j++;
               i++; //确定前缀后缀相同的长度
15
16
           }
17
           else
18
           {
19
               //不同时
20
               if (j != 0)
                   j = ptr[j - 1]; //j回到前一个字符的next值位置
21
22
               else
23
               {
24
                   ptr[i] = 0; //回到模式串的第一个字符
25
                   i++;
26
               }
27
           }
28
       }
29
       return ptr;
30 }
31 int KMP(string s, string p)
32 {
33
       int *next = Get_next(p);
       //获得next数组
34
35
       int i = 0;
36
       int j = 0;
37
       int len = s.length();
38
       while (i < len)</pre>
39
40
           if (s[i] == p[j])
41
           {
```

i++;

42

```
j++; //匹配
43
               if (j >= p.length())
44
                   return i - j;
45
46
           }
           else
47
48
           {
               //字符不相同回到前一个字符的next值位置
49
               if (j != 0)
50
51
                   j = next[j - 1];
52
               else
53
                   i++;
54
           }
55
56
       return -1;
57 }
```

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

Chapter 5 数据结构

5.1 并查集

```
1 #define MAX 1010
2 struct node
3 {
4
       int par;
5
       //int rank;
6
       //路径压缩后 rank=1或2 rank失去了意义
7
       int data;
8 };
9 node ns[MAX];
10 void Init()
11 {
       for (int i = 1; i < MAX; i++)</pre>
12
13
14
           ns[i].par = i;
15
       }
16
   }
17
   int Find(int i)
18
   {
       if (ns[i].par == i)
19
20
       {
21
           //返回根结点
22
           return i;
23
       }
24
       ns[i].par = Find(ns[i].par);
25
       //路径压缩
26
       return ns[i].par;
27 }
28 void Union(int i, int j)
29 {
30
       int pi = Find(i);
       int pj = Find(j);
31
32
       if (pi != pj)
33
34
           ns[pi].par = pj;
35
       }
36 }
```

5.2 线段树

5.2.1 基础操作

```
1 const int N = 1e5 + 10;
 2 #define ls(a) (a << 1)</pre>
3 #define rs(a) (a << 1 | 1)</pre>
 4 struct node
5 {
        int val;
6
       int lazy;
7
8 };
9 node tree[N << 2];</pre>
10 int a[N];
11 void PushUp(int rt)
12 {
       tree[rt].val = tree[ls(rt)].val + tree[rs(rt)];
13
14 }
15 void PushDown(int ls, int rs, int rt)
16 {
17
       tree[ls(rt)].val += ls * tree[rt].lazy;
18
       tree[rs(rt)].val += rs * tree[rt].lazy;
19
       tree[ls(rt)].lazy += tree[rt].lazy;
20
        tree[rs(rt)].lazy += tree[rt].lazy;
21
       tree[rt].lazy = 0;
22 }
23 void Build(int left, int right, int rt)
24 {
25
        if (left == right)
26
        {
27
            tree[rt].val = a[left];
28
           return;
29
        }
30
        int mid = (left + right) >> 1;
31
        Build(left, mid, ls(rt));
        Build(mid + 1, right, rs(rt));
32
33
       PushUp(rt);
34
       //向上更新
35 }
```

5.2.2 单点更新

20

```
void Update(int left, int right, int rt, int pos, int val)
^2
   {
3
       if (left == right && left == pos)
 4
5
            tree[rt].val += val;
6
            return;
 7
8
       int mid = (left + right) >> 1;
9
       if (tree[rt].lazy)
10
       {
            PushDown(mid - left + 1, right - mid, rt);
11
12
13
       if (mid >= pos)
            Update(left, mid, ls(rt), pos, val);
14
       else if (pos > mid)
15
16
            Update(mid + 1, right, rs(rt), pos, val);
17
       PushUp(rt);
18 }
   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)
 1
 2
   {
3
       if (left >= s && right <= t)</pre>
 4
       {
5
            tree[rt].val += (right - left + 1) * val;
6
            tree[rt].lazy += val;
 7
            return;
8
       }
9
       int mid = (left + right) >> 1;
10
       if (tree[rt].lazy)
11
       {
            PushDown(mid - left + 1, right - mid, rt);
12
13
14
       if (mid < s)
15
            Update(mid + 1, right, rs(rt), s, t, val);
16
       else if (mid >= t)
17
            Update(left, mid, ls(rt), s, t, val);
18
       else
19
       {
```

Update(left, mid, ls(rt), s, t, val);

5.2.4 区间查询

```
1 void Query(int left, int right, int s, int t, int rt)
2
3
        if (left >= s && right <= t)</pre>
 4
        {
5
            return tree[rt].val;
6
7
        int mid = (left + right) >> 1;
8
        if (tree[rt].lazy)
9
            PushDown(mid - left + 1, right - mid, rt);
10
        long long sum = 0;
11
        if (mid < s)</pre>
12
            sum += Query(mid + 1, right, rs(rt), s, t, val);
13
        else if (mid >= t)
14
            sum += Query(left, mid, ls(rt), s, t, val);
15
        else
16
        {
            sum += Query(left, mid, ls(rt), s, t, val);
17
            sum += Query(mid + 1, right, rs(rt), s, t, val);
18
19
        }
20
        return sum;
21 }
```

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

```
1 #define N 1000100
 2 long long c[N];
 3 int n,q;
 4 int lowbit(int x)
 5
        return x&(-x);
 6
 7
   }
   void change(int x,int v)
9
10
        while(x<=n)</pre>
11
        {
12
            c[x]+=v;
            x+=lowbit(x);
13
14
        }
15
   }
   long long getsum(int x)
16
17
        long long ans=0;
18
        while(x>=1)
19
20
21
            ans+=c[x];
22
            x-=lowbit(x);
23
        }
24
        return ans;
25 }
```

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
```

7 getsum(x)

例题: https://loj.ac/problem/131

5.3.3 区间修改,区间查询

```
1 //初始化
2 change(c1,i,cur-pre);
3 change(c2,i,i*(cur-pre));
4 //为什么这么写? 你需要写一下前缀和的表达式
5 //区间修改
6 change(c1,l,x);
7 change(c2,l,l*x);
8 change(c1,r+1,-x);
9 change(c2,r+1,-(r+1)*x);
10 //区间查询
11 temp1=l*getsum(c1,l-1)-getsum(c2,l-1);
12 temp2=(r+1)*getsum(c1,r)-getsum(c2,r);
13 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

```
1 #define N 5050
 2 long long tree[N][N];
 3 long long n,m;
 4 long long lowbit(long long x)
5 {
       return x&(-x);
 6
 7 }
8 void change(long long x,long long y,long long val)
9 {
10
        long long init_y=y;
        //这里注意n,m的限制
11
       while(x<=n)</pre>
12
13
14
           y=init_y;
15
           while(y<=m)</pre>
16
            {
17
                tree[x][y]+=val;
```

```
18
               y+=lowbit(y);
19
20
           x+=lowbit(x);
21
       }
22 }
   long long getsum(long long x,long long y)
24 {
25
       long long ans=0;
26
       long long init_y=y;
27
       while(x>=1)
28
       {
29
           y=init_y;
30
           while(y>=1)
31
32
               ans+=tree(x)(y);
33
               y-=lowbit(y);
34
           }
35
           x-=lowbit(x);
36
       }
37
       //这里画图理解
38
       return ans;
39 }
40 //初始化
41 change(x,y,k);
42 //二维前缀和
43 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

```
1 #define N 2050
2 long long t1[N][N];
3 long long t2[N][N];
4 long long t3[N][N];
5 long long t4[N][N];
6 long long n,m;
7 long long lowbit(long long x)
8 {
9    return x&(-x);
10 }
11 long long getsum(long long x,long long y)
12 {
```

```
13
        long long ans=0;
        long long init_y=y;
14
15
        long long init_x=x;
       while(x>=1)
16
17
        {
18
            y=init_y;
19
            while(y>=1)
20
            {
21
                ans+=(init_x+1)*(init_y+1)*t1[x][y];
22
                ans-=(init_y+1)*t2[x][y];
23
                ans-=(init_x+1)*t3[x][y];
24
                ans+=t4[x][y];
25
                y-=lowbit(y);
26
27
            x-=lowbit(x);
28
29
        return ans;
30 }
31 void change(long long x,long long y,long long val)
32 {
33
        long long init_x=x;
34
        long long init_y=y;
35
       while(x<=n)</pre>
36
37
            y=init_y;
38
            while(y<=m)</pre>
39
            {
40
                t1[x][y]+=val;
41
                t2[x][y]+=init_x*val;
42
                t3[x][y]+=init_y*val;
43
                t4[x][y]+=init_x*init_y*val;
44
                y+=lowbit(y);
45
46
            x+=lowbit(x);
47
        }
48 }
49 //区间修改
50 change(c+1,d+1,x);
51 change(a,b,x);
52 change(a,d+1,-x);
53 change(c+1,b,-x);
54 //区间查询
55 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

```
1 #define inf 0x3f3f3f3f
 2 \text{ const int N} = 2e5 + 20;
3 struct node
4 {
 5
        long long u, v, w;
        node(int uu, int vv, int ww) : u(uu), v(vv), w(ww)
 6
 7
        {
8
9
        bool operator<(node n) const</pre>
10
11
            return w > n.w;
12
        }
13 };
14 long long n, m;
15 priority_queue<node> q;
16 vector<pair<long long, long long>> G[N];
17 bool vis[N];
18 long long Prim()
19 {
20
        long long ans = 0;
21
        for (auto ele : G[1])
22
        {
23
            q.push(node(1, ele.first, ele.second));
24
        }
25
        memset(vis, 0, sizeof(vis));
26
        vis[1] = 1;
27
        int t = n - 1;
28
        while (t--)
29
        {
30
            node top = q.top();
31
            q.pop();
32
            while (vis[top.v])
33
34
                top = q.top();
35
                q.pop();
36
            }
37
            ans += top.w;
38
            cout<<ans<<endl;</pre>
39
            vis[top.v] = 1;
```

30

while (!q.empty())

```
for (auto ele : G[top.v])
40
41
42
                q.push(node(top.v, ele.first, ele.second));
43
           }
44
       }
45
       return ans;
46 }
   6.1.2
          Kruskal
   基于并查集
 1 const long long MAXN = 2e5 + 20;
 2 struct Edge
3 {
 4
       long long u, v, w;
       bool operator<(Edge e) const</pre>
 5
 6
 7
           return w > e.w;
 8
       }
9
       Edge(long long uu, long long vv, long long ww) : u(uu), v(vv
       ), w(ww)
10
       {
11
       }
12 };
13 long long fa[MAXN];
14 long long Find(long long x)
15 {
16
       if (fa[x] == -1)
17
           return x;
18
       else
19
           return fa[x] = Find(fa[x]);
20 }
21 priority_queue<Edge> q;
22 long long n; //点
23 long long m; //边
24 long long Kruskal()
25 {
26
       memset(fa, -1, sizeof(fa));
27
       long long cnt = 0;
28
       long long ans = 0;
29
       long long fu, fv;
```

```
31
        {
32
            Edge now = q.top();
            q.pop();
33
            fu = Find(now.u);
34
35
            fv = Find(now.v);
            if (fu != fv)
36
37
            {
                 //cout<<"add:"<<now.u<<" "<<now.v<<" "<<now.w<<endl;</pre>
38
                 ans += now.w;
39
40
                 fa[fu] = fv;
41
                 cnt++;
42
            if (cnt == n - 1)
43
                 break;
44
45
        if (cnt < n - 1)</pre>
46
47
            return -1;
48
        return ans;
49 }
```

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

```
1 #define N 500050
2 int depth[N];
3 int fa[N][20];
 4 vector<int> v[N];
 5 int lg[N]; //log2(n) floor
6 int n, m, s;
7 void init()
8
9
        lg[0] = -1;
10
        //floor
11
       for (int i = 1; i <= n; i++)
12
13
            lg[i] = lg[i >> 1] + 1;
14
        }
15
16
   void DFS(int cur, int pre)
17
18
        depth[cur] = depth[pre] + 1;
19
        fa[cur][0] = pre;
20
        for (int i = 1; (1 << i) <= depth[cur]; i++)</pre>
21
        {
22
            fa[cur][i] = fa[fa[cur][i - 1]][i - 1];
23
        }
24
       for (auto ele : v[cur])
25
26
            if (ele != pre)
27
            {
28
                DFS(ele, cur);
29
            }
30
        }
31
32 int LCA(int a, int b)
33 {
34
        // assume depth[a]>=depth[b]
35
        if (depth[a] < depth[b])</pre>
            swap(a, b);
36
37
        // reset to the same depth
38
       while (depth[a] > depth[b])
39
        {
```

```
40
            a = fa[a][lg[depth[a] - depth[b]]];
41
            //up
42
        }
       if (a == b)
43
44
           return a;
       for (int k = \lg[depth[a]] + 1; k >= 0; k--)
45
46
            if (fa[a][k] != fa[b][k])
47
48
                a = fa[a][k];
49
50
                b = fa[b][k];
51
                //up
52
            }
53
54
       return fa[a][0];
55 }
```

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

6.3 强连通分量

6.3.1 Tarjan

```
1 const int N = 10500;
2 vector<int> G[N];
3 bool vis[N];
4 int dfn[N];
5 int low[N];
6 int cnt;
7 stack<int> s;
8 int n, m;
9 void init()
10 {
       memset(vis, 0, sizeof(vis));
11
12
       memset(dfn, 0, sizeof(dfn));
13
       cnt = 0;
14 }
15 void DFS(int cur)
16 {
17
       dfn[cur] = low[cur] = ++cnt;
18
       s.push(cur);
       vis[cur] = 1;
19
20
       for (auto ele : G[cur])
21
22
            if (!dfn[ele])
23
            {
24
                DFS(ele);
25
                low[cur] = min(low[cur], low[ele]);
26
            }
27
            else if (vis[ele])
28
29
                low[cur] = min(low[cur], dfn[ele]);
30
31
       if (dfn[cur] == low[cur])
32
33
       {
34
            while (1)
35
            {
36
                int t = s.top();
37
                s.pop();
38
                vis[t] = 0;
39
                if (t == cur)
```

```
40 break;
41 }
42 }
43 }
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

```
1 if (dfn[cur] == low[cur])
 2
 3
        sum++;
 4
       while (1)
 5
        {
 6
            int t = s.top();
 7
            s.pop();
8
            color[t] = sum;
9
            pnum[sum]++;
10
            vis[t] = 0;
            if (t == cur)
11
12
                break;
13
        }
14 }
```

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)
 1 #define N 100100
 2 #define 11 long long
3 #define inf 2147483647
 4 vector<pair<int, ll>> G[N];
5 struct edge
6 {
 7
       int to;
       ll weight;
8
9
       edge(int i, ll w) : to(i), weight(w){}
10
       bool operator<(const edge &e) const{return weight > e.weight
       ;}
11 };
12 ll dis[N];
13 bool vis[N];
14 int n, m, s;
15 void Dijkstra(int start)
16 {
17
       for (int i = 1; i <= n; i++){dis[i] = inf;}</pre>
18
       priority_queue<edge> q;
       q.push(edge(start, 0));
19
20
       dis[start] = 0;
21
       while (!q.empty())
22
       {
23
           edge now = q.top();
24
           q.pop();
25
           if (!vis[now.to])
26
27
               vis[now.to] = 1;
               for (auto ele : G[now.to])
28
29
30
                    if (!vis[ele.first] && now.weight + ele.second <</pre>
        dis[ele.first])
31
                    {
32
                        dis[ele.first] = now.weight + ele.second;
33
                        q.push(edge(ele.first, dis[ele.first]));
34
                    }
35
               }
```

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

```
1 #define ll long long
 2 #define inf 0x3f3f3f3f
3 const int M = 205;
4 11 c[M][M];
5 int pre[M];
6 11 flow[M];
7 int n, m;
8 11 BFS(int s, int t)
9
   {
10
        queue<int> q;
       memset(pre, -1, sizeof(pre));
11
12
        flow[s] = inf;
        pre[s] = 0;
13
14
        q.push(s);
        while (!q.empty())
15
16
        {
17
            int u = q.front();
18
            q.pop();
            if (u == t)
19
20
            {
21
                break;
22
23
            for (int v = 1; v <= m; v++)</pre>
24
25
                if (c[u][v] > 0 \&\& pre[v] == -1\&\&v!=s)
26
                {
27
                    pre[v] = u;
28
                    flow[v] = min(c[u][v], flow[u]);
29
                    q.push(v);
30
                }
31
            }
32
33
        if (pre[t] == -1)
34
            return -1;
35
        return flow[t];
36
37 ll Edmonds_Karp(int s, int t)
```

```
38
   {
39
       ll inc;
40
       11 \text{ ans} = 0;
41
       int k, last;
42
       while ((inc = BFS(s, t)) != -1)
43
       {
           k = t;
44
45
           while (k != s)
46
47
               last = pre[k];
               c[last][k] -= inc;
48
49
               c[k][last] += inc;
50
               k = last;
51
           }
52
           ans += inc;
           //cout<<"cur:"<<inc<<endl;</pre>
53
54
       }
55
       return ans;
56 }
   例题: https://www.luogu.org/problemnew/show/P2740
   6.5.1.2 Dinic
   多路增广, 当前弧优化
 1 #define inf 0x3f3f3f3f3f
 2 #define N 1200
3 #define M 245000
 4 #define ll long long
5 //反向边的存在 实际边数的两倍
 6 struct Edge
7 {
8
       11 to;
9
       11 w;
10
       11 next;
11 };
12 Edge edge[M];
13 ll dep[N];
14 ll head[N];
15 ll cur[N]; //当前弧优化
16 	 11 	 cnt = 0;
17 11 n, m;
18 void Init()
19 {
```

```
20
       memset(head, -1, sizeof(head));
21 }
22 void add(ll u, ll v, ll w)
23 {
24
       edge[cnt].to = v;
25
       edge[cnt].w = w;
26
       edge[cnt].next = head[u];
27
       head[u] = cnt++;
28 }
29 void Add(ll u, ll v, ll w)
30 {
31
       add(u, v, w);
32
       add(v, u, 0); //反向边
33 }
34 11 s, t;
35 bool BFS()
36 {
37
       queue<ll> q;
       memset(dep, -1, sizeof(dep));
38
39
       dep[s] = 0;
40
       q.push(s);
41
       while (!q.empty())
42
43
            11 now = q.front();
44
            q.pop();
45
            for (ll i = head[now]; i != -1; i = edge[i].next)
46
            {
                if (edge[i].w > 0 && dep[edge[i].to] == -1)
47
48
                {
49
                    dep[edge[i].to] = dep[now] + 1;
50
                    q.push(edge[i].to);
51
                }
52
            }
53
54
       if (dep[t] == -1)
55
            return false; //不存在分层图
56
       return true;
57 }
58 11 DFS(11 now, 11 flow)
59 {
60
       if (now == t)
61
            return flow;
62
       11 \text{ used} = 0;
```

```
63
        for (ll &i = cur[now]; i != -1; i = edge[i].next)
64
65
            //&: 当前弧优化
66
            if (dep[edge[i].to] == dep[now] + 1 && edge[i].w)
67
                11 inc = DFS(edge[i].to, min(flow - used, edge[i].w)
68
       );
69
                if (inc > 0)
70
71
                    edge[i].w -= inc;
72
                    edge[i ^1].w += inc;
73
                    used += inc;
74
                     if (flow == used)
75
                         break;
76
                }
77
            }
78
        }
79
        if (!used)
            dep[now] = -1;
80
        return used;
81
82
83
   11 Dinic()
84
   {
        11 \text{ ans} = 0;
85
       while (BFS())
86
87
            for (ll i = 1; i <= n; i++)
88
89
90
                cur[i] = head[i];
91
            }
92
            ans += DFS(s, inf);
93
94
        return ans;
95 }
```

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

```
1 #define mod 1000000007
 2 long long quick_pow(long long a, long long b)
3 {
 4
       if (b == 0)
 5
           return 1;
 6
       long long temp = quick_pow(a, b >> 1);
 7
       if (b & 1)
8
           return a % mod * temp % mod * temp % mod;
9
       else
10
           return temp % mod * temp % mod; //快速幂
11 }
```

7.1.2 循环形式

如果递归形式栈溢出可使用循环形式

```
1 #define mod 1000000007
 2 long long quick_pow(long long x,long long n)
 3
   {
 4
       long long ret=1;
       long long temp=x%mod;
 5
       while(n)
 6
 7
       {
 8
            if(n&1)
 9
            {
                ret=(ret*temp)%mod;
10
11
12
            temp=(temp*temp)%mod; //偶次
13
            n>>=1;
14
       }
15
       return ret;//结果
16 }
```

7.2 矩阵快速幂

```
1 #define N 105
2 int m;//矩阵阶
```

```
struct matrix
4
   {
 5
        long long a[N][N];
6
        matrix(){memset(a, 0, sizeof(a));}
7
   };
   matrix matrix_mul(matrix m1, matrix m2)
9
   {
10
       matrix ans;
        for (int k = 0; k < m; k++)
11
12
            for (int i = 0; i < m; i++)</pre>
13
14
            {
15
                if (m1.a[i][k]) //剪枝
16
17
                    for (int j = 0; j < m; j++)
18
19
                         if (m2.a[k][j]) //剪枝
20
21
                             ans.a[i][j] = (ans.a[i][j] + (m1.a[i][k]
        * m2.a[k][j]) % mod) % mod;
22
23
                    }
24
                }
25
            }
26
        }
27
        return ans;
28
   }
   matrix quick_pow_matrix(matrix m1, long long k)
29
30
   {
31
        matrix ans;//递归写法 可能堆栈溢出
32
        for (int i = 0; i < m; i++)</pre>
33
            ans.a[i][i] = 1;
34
       while (k)
35
        {
36
            if (k & 1)
37
                ans = matrix_mul(ans, m1);
38
            m1 = matrix_mul(m1, m1);
39
            k \gg 1;
40
        }
41
        return ans;
42 }
```

7.3 组合数取模

```
1 #define N 100010
2 #define mod 1000000007
3 #define ll long long
4 ll fac[N];//阶乘
5 ll inv[N];//阶乘逆元
6 void init()
7
   {
8
       fac[0]=1;
9
       for(int i=1;i<N;i++)</pre>
10
           fac[i]=(fac[i-1]*i)%mod;
11
12
13
       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
14
15
       for(int i=N-2;i>=0;i--)
16
       {
17
           inv[i]=inv[i+1]*(i+1)%mod;
18
       }
19 }
20 ll C(ll a,ll b)
21 {
22
       if(b>a) return 0;
       if(b==0) return 1;
23
24
       return fac[a]*inv[b]%mod*inv[a-b]%mod;
25 }
```

Chapter 8 计算几何

Chapter 9 其他