南京工業大學
NANJING TECH UNIVERSITY

ICPC Template Manual

作者: 贺梦杰

June 13, 2019

Contents

Chapter 1

基础

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
8      int len = str.length(); //模式串长度
9      while (i < len)
10     {
11         if (str[i] == str[j])
12         {
13             ptr[i] = j + 1;
14             j++;
15             i++; //确定前缀后缀相同的长度
16         }
17         else
18         {
19             //不同时
20             if (j != 0)
21                 j = ptr[j - 1]; //j回到前一个字符的next值位置
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);
34     //获得next数组
35     int i = 0;
36     int j = 0;
37     int len = s.length();
38     while (i < len)
39     {
40         if (s[i] == p[j])
41         {
```

```
42              i++;
43              j++; //匹配
44              if (j >= p.length())
45                  return i - j;
46          }
47          else
48          {
49              //字符不相同回到前一个字符的next值位置
50              if (j != 0)
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 线段树

5.1.1 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
}
```

5.1.2 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*

5.1.3 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);
```

```
21          Update(mid + 1, right, rs(rt), s, t, val);
22      }
23      PushUp(rt);
24  }
```

5.1.4 区间查询

```
1   void Query(int left, int right, int s, int t, int rt)
2   {
3       if (left >= s && right <= t)
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)
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      {
17          sum += Query(left, mid, ls(rt), s, t, val);
18          sum += Query(mid + 1, right, rs(rt), s, t, val);
19      }
20      return sum;
21  }
```

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

5.2 树状数组

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

5.2.1 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*

5.2.2 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*

5.2.3 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*

5.3 二维树状数组

5.3.1 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;
```

```
18              y+=lowbit(y);
19          }
20          x+=lowbit(x);
21      }
22  }
23  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*

5.3.2 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;
14      long long init_y=y;
15      long long init_x=x;
16      while(x>=1)
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)
36      {
37          y=init_y;
38          while(y<=m)
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

```
#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;
```

```
40          for (auto ele : G[top.v])
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;
5       bool operator<(Edge e) const
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;
30      while (!q.empty())
```

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

```
#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])
    {
```

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

```
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)
```

```
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;
11         if (t == cur)
12             break;
13     }
14 }
```

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

Chapter 7

数学

Chapter 8

计算几何

Chapter 9

其他