

# ACM/ICPC 模板

XXXXXX

2023 年 2 月 15 日

# 目录

|          |                         |          |
|----------|-------------------------|----------|
| <b>1</b> | <b>图论</b>               | <b>1</b> |
| 1.1      | Dijkstra . . . . .      | 1        |
| 1.1.1    | 堆优化的 Dijkstra . . . . . | 1        |
| 1.1.2    | 朴素的 Dijkstra . . . . .  | 1        |
| <b>2</b> | <b>字符串</b>              | <b>2</b> |
| 2.1      | KMP . . . . .           | 2        |
| 2.2      | 后缀数组 . . . . .          | 2        |

# 1 图论

## 1.1 Dijkstra

### 1.1.1 堆优化的 Dijkstra

```
1 void dij(int s) {
2     memset(dis, 0x3f, sizeof dis);
3     dis[s] = 0; //初始化dis数组
4     priority_queue<pair<int, int>> q; //一个优先队列
5     q.push({0, s}); //c++中的pair<int, int>默认先按第一维比较, 所以把dis值放在第一维
6     while(!q.empty()) {
7         int x = q.top().second; q.pop(); //取堆顶
8         if(flag[x]) continue; //判断是否已经在S集合中, 如果已经在S集合中, 那么直接continue
9         flag[x] = true; //标记x在S集合中
10        for(int i = fr[x], y; i; i = nxt[i]) { //遍历所有与x相连的边
11            y = to[i];
12            if(dis[y] > dis[x] + w[i]) { //判断是否可以松弛。
13                dis[y] = dis[x] + w[i];
14                if(!flag[y]) q.push({-dis[y], y}); //c++中的优先队列默认为大根堆, 而我们需要小根堆, 为了方便可以直接取相反数
15            }
16        }
17    }
18 }
```

### 1.1.2 朴素的 Dijkstra

```
1 void dij(int s) {
2     memset(dis, 0x3f, sizeof dis);
3     dis[s] = 0; //初始化dis数组
4     for(int k = 1, t; k <= n; ++k) { //一共n轮, 每次找不在S集合中的点中dis值最小的点, 将其放入S集合中并用它去松弛其他点
5         t = -1;
6         for(int i = 1; i <= n; ++i)
7             if(!flag[i] && (t == -1 || dis[i] < dis[t])) t = i; //如果 (不在S中 && (t还没有赋值 || i的dis值比t更小)), 那么t = i
8         flag[t] = true; //标记t在S集合中
9         for(int i = 1; i <= n; ++i) //松弛操作
10            dis[i] = min(dis[i], dis[t] + a[t][i]);
11    }
12 }
```

## 2 字符串

### 2.1 KMP

```

1 // 未经验证
2 int nxt[MAXN];
3 void getNext(char s[]) { // 模式串next数组
4     int n = strlen(s + 1);
5     for(int i = 0; i <= n; ++i) nxt[i] = 0;
6     for(int i = 2, j = 0; i <= n; ++j) {
7         while(j && s[i] != s[j + 1]) j = nxt[j];
8         if(s[i] == s[j + 1]) ++j;
9         nxt[i] = j;
10    }
11 }
12 int kmp(char s[], char t[]) { //返回第一次匹配的地方, 否则返回-1
13     int n = strlen(s + 1), m = strlen(t + 1);
14     for(int i = 1, j = 0; i <= m; ++j) {
15         while(j && t[i] != s[j + 1]) j = nxt[j];
16         if(t[i] == s[j + 1]) ++j;
17         if(j == n) return i - m + 1;
18     }
19     return -1;
20 }

```

### 2.2 后缀数组

```

1 struct SA {
2     static const int MAXN = 1e5 + 50;
3     static char tmpStr[MAXN];
4     static int n, sa[MAXN], rk[MAXN], cnt[MAXN], x[MAXN], y[MAXN];
5     static int height[MAXN], st[MAXN][35]; // 不需要可以删除, 注意必须连同下面初始化一起删除
6     static void work(string s) {
7         strcpy(tmpStr + 1, s.c_str());
8         work(tmpStr);
9     }
10    static void work(char s[]) {
11        int m = 255; n = strlen(s + 1);
12        for(int i = 1; i <= n; ++i) ++cnt[x[i] = s[i]];
13        for(int i = 2; i <= m; ++i) cnt[i] += cnt[i - 1];
14        for(int i = n; i >= 1; --i) sa[cnt[x[i]]--] = i;
15        for(int k = 1, p = 0; k <= n; k <= 1, m = p, p = 0) {
16            for(int i = n - k + 1; i <= n; ++i) y[++p] = i;
17            for(int i = 1; i <= n; ++i) if(sa[i] > k) y[++p] = sa[i] - k;
18            for(int i = 1; i <= m; ++i) cnt[i] = 0;
19            for(int i = 1; i <= n; ++i) ++cnt[x[i]];
20            for(int i = 2; i <= m; ++i) cnt[i] += cnt[i - 1];
21            for(int i = n; i >= 1; --i) sa[cnt[x[y[i]]]--] = y[i], y[i] = 0;
22            swap(x, y), x[sa[1]] = p = 1;
23            for(int i = 2; i <= n; ++i)
24                x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1] + k]) ? p : ++p;
25            if(p == n) break;
26        }
27        for(int i = 1; i <= n; ++i) rk[sa[i]] = i;
28        // height数组: 引理 height[rk[i]] >= height[rk[i - 1]] - 1
29        for(int i = 1, k = 0; i <= n; ++i) {
30            if(rk[i] == 1) continue;
31            if(k) --k;
32            int j = sa[rk[i] - 1];
33            while(j + k <= n && i + k <= n && s[i + k] == s[j + k]) ++k;
34            height[rk[i]] = k;

```

```
35     }
36     // lcp数组的ST表: lcp(sa[i], sa[j]) = min(height[i.....j])
37     for(int i = 1; i <= n; ++i) st[i][0] = height[i];
38     for(int i = n; i >= 1; --i)
39         for(int j = 1; i + (1 << j) - 1 <= n; ++j)
40             st[i][j] = min(st[i][j - 1], st[i + (1 << j) - 1][j - 1]);
41 }
42 static int lcp(int i, int j) { // ST表求LCP
43     int a = rk[i], b = rk[j];
44     if(a == b) return 1;
45     if(a > b) swap(a, b);
46     int len = log2(b - a);
47     return min(st[a + 1][len], st[b - (1 << len) + 1][len]);
48 }
49 };
50 char SA::tmpStr[MAXN];
51 int SA::n, SA::sa[MAXN], SA::rk[MAXN], SA::cnt[MAXN], SA::x[MAXN], SA::y[MAXN];
52 int SA::height[MAXN], SA::st[MAXN][35]; // 这里
```