

XCPC Template For Contest

Ayers Zhang

2022 年 10 月 31 日

目录

| | | |
|----------|---------------------------|-----------|
| 1 | 杂项 | 3 |
| 1.1 | 快速读入和快速输出 | 3 |
| 1.2 | | 3 |
| 2 | 图论 | 4 |
| 2.1 | 最短路 | 4 |
| 2.2 | DAG 最长路 | 4 |
| 2.3 | 网络流相关 | 6 |
| 2.3.1 | 最小费用最大流 | 6 |
| 2.3.2 | 带权二分图最大匹配 KM | 7 |
| 3 | 数学 | 10 |
| 3.1 | 数论 | 10 |
| 3.2 | 组合数学 | 10 |
| 3.2.1 | 球盒问题模型 | 10 |
| 4 | 数据结构 | 11 |
| 4.1 | ST 表解决静态 RMQ 问题 | 11 |

1 杂项

1.1 快速读入和快速输出

```
1  template <typename T>
2  void read(T &x) {
3      x = 0; char c = 0; int f = 1;
4      for (; !isdigit(c); c = getchar()) if (c == '-') f = -1;
5      for (; isdigit(c); c = getchar()) x = x * 10 + (c & 15);
6      x *= f;
7  }
8
9  template <typename T>
10 void write(T x) {
11     if (x < 0) putchar('-'), x = -x;
12     if (x > 9) write(x / 10);
13     putchar(x % 10 + '0');
14 }
```

1.2

2 图论

2.1 最短路

```
1 struct Node {
2     int u;
3     i64 dis;
4     Node() {}
5     Node(int a, i64 b) : u(a), dis(b) {}
6     bool operator<(const Node &rhs) const {
7         return dis > rhs.dis;
8     }
9 };
10 std::priority_queue<Node> q;
11 i64 dis[N];
12 bool vis[N];
13 void dijkstra(int s) {
14     while (!q.empty()) q.pop();
15     for (int i = 1; i <= n; i++) {
16         dis[i] = inf, vis[i] = 0;
17     }
18     q.push(Node(s, 0));
19     dis[s] = 0;
20     while (!q.empty()) {
21         int u = q.top().u;
22         q.pop();
23         if (vis[u]) continue;
24         vis[u] = 1;
25         for (int e = head[u]; e; e = edge[e].nxt) {
26             int v = edge[e].v;
27             if (dis[v] > dis[u] + edge[e].w) {
28                 dis[v] = dis[u] + edge[e].w;
29                 q.push(Node(v, dis[v]));
30             }
31         }
32     }
33 }
```

2.2 DAG 最长路

由于带环图理论上不存在最长路，所以只考虑 DAG 中的最长路，采用拓扑排序进行求解。

```
1 void findLongestPath() {
2     std::queue<int> q;
3     for (int i = 0; i < n; i++) {
4         if (ind[i] == 0) {
5             q.push(i);
6         }
7     }
8     dis[0] = inf;
9     while (!q.empty()) {
10         int u = q.front();
11         q.pop();
12         for (int e = head[u]; e; e = ed[e].nxt) {
```

```
13     int v = ed[e].to;
14     ind[v]--;
15     dis[v] = std::max(dis[v], dis[u] + ed[e].w);
16     if (ind[v] == 0) {
17         q.push(v);
18     }
19 }
20 }
21 }
```

2.3 网络流相关

2.3.1 最小费用最大流

Dinic 写法，在普通情况下时间复杂度为 $O(V^2E)$ ，二分图的时间复杂度为 $O(VE)$ 。

每条边单位流量会花费价值，在跑出最大流的情况下要求费用最小

```
1  const int N = 805, M = 320005;
2
3  struct Edge {
4      int v;
5      i64 w, c;
6      int nxt;
7
8      Edge() {}
9      Edge(int a, i64 ws, i64 cs, int b) : v(a), w(ws), c(cs), nxt(b) {}
10 } edge[M];
11
12 int edgeCnt = 1;
13 int head[N], arc[N];
14 bool vis[N];
15
16 int S, T;
17
18 i64 dis[N];
19 i64 ans, ret;
20
21 void addEdge(int u, int v, i64 w, i64 c) {
22     edge[++edgeCnt] = Edge(v, w, c, head[u]);
23     head[u] = edgeCnt;
24 }
25
26 void add(int u, int v, i64 w, i64 c) {
27     addEdge(u, v, w, c);
28     addEdge(v, u, 0, -c);
29 }
30
31 bool spfa() {
32     for (int i = S; i <= T; i++) {
33         dis[i] = inf;
34         arc[i] = head[i];
35     }
36     std::queue<int> q;
37     dis[S] = 0;
38     vis[S] = 1;
39     q.push(S);
40     while (!q.empty()) {
41         int u = q.front();
42         q.pop();
43         vis[u] = 0;
44         for (int e = head[u]; e; e = edge[e].nxt) {
45             int v = edge[e].v;
46             if (edge[e].w && dis[v] > dis[u] + edge[e].c) {
47                 dis[v] = dis[u] + edge[e].c;
48                 if (!vis[v]) {
49                     q.push(v);
```

```
50     vis[v] = 1;
51     }
52     }
53     }
54     }
55     return dis[T] != inf;
56 }
57
58 i64 dfs(int u, i64 flow) {
59     if (u == T) {
60         return flow;
61     }
62     vis[u] = 1;
63     i64 res = 0;
64     for (int &e = arc[u]; e && res < flow; e = edge[e].nxt) {
65         int v = edge[e].v;
66         if (!vis[v] && edge[e].w && dis[v] == dis[u] + edge[e].c) {
67             i64 x = dfs(v, std::min(edge[e].w, flow - res));
68             if (x) {
69                 edge[e].w -= x;
70                 edge[e ^ 1].w += x;
71                 res += x;
72                 ret += x * edge[e].c;
73             }
74         }
75     }
76     vis[u] = 0;
77     return res;
78 }
79
80 void solve() {
81     // 注意要事先设定好源点和汇点，源点为第一个点，汇点为最后一个点
82     // 其中的 ans 就是最大流，ret 为最小费用
83     memset(vis, 0, sizeof vis);
84     ans = 0;
85     ret = 0;
86     while (spfa()) {
87         i64 x;
88         while ((x = dfs(S, inf))) {
89             ans += x;
90         }
91     }
92 }
```

2.3.2 带权二分图最大匹配 KM

```
1  const int N = 805;
2
3  int n, m;
4  int matchx[N], matchy[N];
5  int pre[N];
6  bool visx[N], visy[N];
7  i64 slack[N];
8  i64 lx[N], ly[N];
9  std::queue<int> q;
```

```
10 i64 g[N][N];
11
12 bool check(int u) {
13     visy[u] = 1;
14     if (matchy[u] != -1) {
15         q.push(matchy[u]);
16         visx[matchy[u]] = 1;
17         return 0;
18     }
19     while (u != -1) {
20         matchy[u] = pre[u];
21         std::swap(u, matchx[pre[u]]);
22     }
23     return 1;
24 }
25
26 void add(int u, int v, i64 w) {
27     g[u][v] = std::max(0ll, w);
28 }
29
30 void bfs(int i) {
31     while (!q.empty()) {
32         q.pop();
33     }
34     q.push(i);
35     visx[i] = true;
36     while (true) {
37         while (!q.empty()) {
38             int u = q.front();
39             q.pop();
40             for (int v = 1; v <= n; v++) {
41                 if (!visy[v]) {
42                     i64 delta = lx[u] + ly[v] - g[u][v];
43                     if (slack[v] >= delta) {
44                         pre[v] = u;
45                         if (delta) {
46                             slack[v] = delta;
47                         } else if (check(v)) { // delta=0 代表有机会加入相等子图 找增广路
48                             // 找到就return 重建交错树
49                             return;
50                         }
51                     }
52                 }
53             }
54         }
55         // 没有增广路 修改顶标
56         i64 a = inf;
57         for (int j = 1; j <= n; j++) {
58             if (!visy[j]) {
59                 a = std::min(a, slack[j]);
60             }
61         }
62         for (int j = 1; j <= n; j++) {
63             if (visx[j]) { // S
64                 lx[j] -= a;
65             }
66         }
67     }
68 }
```



```
66     if (visy[j]) { // T
67         ly[j] += a;
68     } else { // T'
69         slack[j] -= a;
70     }
71 }
72 for (int j = 1; j <= n; j++) {
73     if (!visy[j] && slack[j] == 0 && check(j)) {
74         return;
75     }
76 }
77 }
78 }
79
80 i64 res;
81
82 void solve() {
83     n = std::max(n, m);
84     for (int i = 1; i <= n; i++) {
85         matchx[i] = matchy[i] = -1;
86         lx[i] = -inf;
87         ly[i] = slack[i] = 0;
88         pre[i] = 0;
89         visx[i] = visy[i] = 0;
90     }
91     res = 0;
92     for (int i = 1; i <= n; i++) {
93         for (int j = 1; j <= n; j++) {
94             lx[i] = std::max(lx[i], g[i][j]);
95         }
96     }
97     for (int i = 1; i <= n; i++) {
98         std::fill(slack + 1, slack + 1 + n, inf);
99         std::fill(visx + 1, visx + 1 + n, false);
100         std::fill(visy + 1, visy + 1 + n, false);
101         bfs(i);
102     }
103     for (int i = 1; i <= n; i++) {
104         if (g[i][matchx[i]] > 0) {
105             res += g[i][matchx[i]];
106         }
107     }
108 }
```

3 数学

3.1 数论

3.2 组合数学

3.2.1 球盒问题模型

| n 个球 | r 个盒子 | 是否允许有空盒 | 方案数 |
|--------|---------|---------|------------------------|
| 不相同 | 不相同 | 允许 | r^n |
| 相同 | 不相同 | 不允许 | $\binom{n-1}{r-1}$ |
| 相同 | 不相同 | 允许 | $\binom{n+r-1}{r-1}$ |
| 不相同 | 不相同 | 不允许 | $r! \times S(n, r)$ |
| 不相同 | 相同 | 不允许 | $S(n, r)$ |
| 不相同 | 相同 | 允许 | $\sum_{k=1}^r S(n, k)$ |

4 数据结构

4.1 ST 表解决静态 RMQ 问题

```
1  template <typename T>
2  class sparseTable {
3  private:
4      std::vector< std::vector<T> > st;
5
6      static T func(const T &lhs, const T &rhs) {
7          return std::max(lhs, rhs);
8      }
9
10     std::function<T(const T &, const T &)> judge;
11
12 public:
13     sparseTable(const std::vector<T> &v, std::function<T(const T &, const T &)> _func = func) {
14         judge = _func;
15         int n = v.size(), k = ceil(log2(n)) + 1;
16         st.assign(n, std::vector<T>(k, 0));
17         for (int i = 0; i < n; i++) {
18             st[i][0] = v[i];
19         }
20         for (int j = 1; j < k; j++) {
21             for (int i = 0; i + (1 << j) - 1 < n; i++) {
22                 st[i][j] = judge(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
23             }
24         }
25     }
26
27     T query(int l, int r) {
28         int k = log2(r - l + 1);
29         return judge(st[l][k], st[r - (1 << k) + 1][k]);
30     }
31 };
```