

# template

- template
  - 写在前面
    - 基础模版
    - vimrc
  - 数据结构
    - zkw 线段树
    - 珂朵莉树
  - 数学
    - 快速幂
    - 高斯消元
  - 图论
    - 倍增
    - 网络流
      - 最大流
      - 费用流

## 写在前面

### 基础模版

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define OPFI(x) freopen(#x".in", "r", stdin);\
5                      freopen(#x".out", "w", stdout)
6  #define REP(i, a, b) for(int i=(a); i<=(b); ++i)
7  #define REPd(i, a, b) for(int i=(a); i>=(b); --i)
8  inline ll rd(){
9      ll r=0, k=1; char c;
10     while(!isdigit(c=getchar())) if(c=='-') k=-k;
11     while(isdigit(c)) r=r*10+c-'0', c=getchar();
12     return r*k;
13 }
14 int main(){
15     return 0;
```

16 }

## vimrc

```
1  syntax on
2  set ts=4
3  set expandtab
4  set autoindent
5  set cindent
6  set shiftwidth=4
7  set nu
8  set softtabstop=4
9  set smartindent
10 set showmatch
11 set ruler
12 set mouse=a
13 inoremap <F1> <esc>:w<CR>
14 inoremap <F5> <esc>:below term<CR>
15 nmap <F1> :w<CR>
16 nmap <F5> :below term<CR>
17 colo habamax
18 set title
19 set shell=powershell
20 set wim=list
21 set backspace=indent,eol,start
22 set nocompatible
```

## 数据结构

### zkw 线段树

### 单点修 区间查

```
1  ll s[N<<2], a[N];
2  int M;
3
4  ll f(ll x, ll y){
5      return x+y; // 改这
6  }
7
8  void build(){
```

```

9      for(M=1; M<=n+1; M<=1);
10     REP(i, 1, n) s[i+M]=a[i];
11     REPd(i, M-1, 1) s[i]=f(s[2*i], s[2*i+1]);
12 }
13
14 ll qrangle(int l, int r, ll init){ // 根据 f 传 init
15     ll res=init;
16     for(l=l+M-1, r=r+M+1; l^r^1; l>>=1, r>>=1){
17         if(~l&1) res=f(res, s[l^1]);
18         if(r&1) res=f(res, s[r^1]);
19     }
20     return res;
21 }
22
23 void edit(int x, ll v){
24     for(s[x+=M]=v, x>>=1; x; x>>=1){
25         s[x]=f(s[2*x], s[2*x+1]);
26     }
27 }
28
29 ll qpoint(int x){
30     return s[x+M];
31 }

```

珂朵莉树

```

1  struct node{
2      int l, r;
3      mutable int v;
4      bool operator<(const node& rhs) const { return l<rhs.l;
5  }
6  };
7
8  set<node> odt;
9
10 typedef set<node>::iterator iter;
11
12 iter split(ll p){
13     iter tmp=odt.lower_bound((node){p, 0, 0});
14     if(tmp!=odt.end()&&tmp->l==p) return tmp;
15     --tmp;
16     int tl=tmp->l, tr=tmp->r, tv=tmp->v;
17     odt.erase(tmp);
18     odt.insert((node){tl, p-1, tv});

```

```

17     return odt.insert((node){p, tr, tv}).first;
18 }
19
20 // 【修改 & 查询】注意 split 顺序
21 // iter itr=split(r+1), itl=split(l);

```

## 数学

### 快速幂

```

1  const ll MOD=998244353; // 改模数
2
3  ll qpow(ll a, ll x){
4      ll res=1;
5      a%=MOD;
6      while(x){
7          if(x&1) res=res*a%MOD;
8          a=a*a%MOD, x>>=1;
9      }
10     return res;
11 }
12
13 ll inv(ll x){ return qpow(x, MOD-2); } // 模数为质数时

```

### 高斯消元

```

1  const int N=110;
2  ll n;
3  double a[N][N], b[N];
4  void work(){
5      n=rd();
6      REP(i, 1, n){
7          REP(j, 1, n) a[i][j]=rd();
8          b[i]=rd();
9      }
10     REP(i, 1, n){
11         int t=i;
12         REP(j, i+1, n) if(abs(a[j][i])>1e-7&&(abs(a[t]
13 [i])>abs(a[j][i])||abs(a[t][i])<1e-7)) t=j;
14         REP(j, i, n) swap(a[t][j], a[i][j]);
15         if(abs(a[i][i])<1e-7){

```

```

15         puts("No Solution");
16         return 0;
17     }
18     swap(b[t], b[i]);
19     double e=a[i][i];
20     REP(j, i, n) a[i][j]/=e;
21     b[i]/=e;
22     REP(j, i+1, n){
23         double d=a[j][i];
24         REP(k, i, n) a[j][k]-=d*a[i][k];
25         b[j]-=d*b[i];
26     }
27 }
28 REPd(i, n, 1) REP(j, 1, i-1) b[j]-=a[j][i]*b[i], a[j]
[i]=0;
29 // REP(i, 1, n) printf("%.2f\n", b[i]);
30 // b[1...n] 保存 Ax=b 的解
31 }

```

## 图论

### 倍增

```

1 void dfs(int x, int fa){
2     pa[x][0]=fa; dep[x]=dep[fa]+1;
3     REP(i, 1, SP) pa[x][i]=pa[pa[x][i-1]][i-1];
4     for(int& v:g[x]) if(v!=fa){
5         dfs(v, x);
6     }
7 }
8
9 int lca(int x, int y){
10     if (dep[x]<dep[y]) swap(x, y);
11     int t=dep[x]-dep[y];
12     REP(i, 0, SP) if(t&(1<<i)) x=pa[x][i];
13     REPd(i, SP-1, -1){
14         int xx=pa[x][i], yy=pa[y][i];
15         if (xx!=yy) x=xx, y=yy;
16     }
17     return x==y?x:pa[x][0];
18 }

```

## 网络流

不是我写的，但是看着还好

最大流

其中 11 是我改的，不敢保证有没有漏改，但是过了洛谷模版题

```
1  constexpr ll INF=LLONG_MAX/2;
2
3  struct E {
4      int to; ll cp;
5      E(int to, ll cp): to(to), cp(cp) {}
6  };
7
8  struct Dinic {
9      static const int M = 1E5 * 5;
10     int m, s, t;
11     vector<E> edges;
12     vector<int> G[M];
13     int d[M];
14     int cur[M];
15
16     void init(int n, int s, int t) {
17         this->s = s; this->t = t;
18         for (int i = 0; i <= n; i++) G[i].clear();
19         edges.clear(); m = 0;
20     }
21
22     void addedge(int u, int v, ll cap) {
23         edges.emplace_back(v, cap);
24         edges.emplace_back(u, 0);
25         G[u].push_back(m++);
26         G[v].push_back(m++);
27     }
28
29     bool BFS() {
30         memset(d, 0, sizeof d);
31         queue<int> Q;
32         Q.push(s); d[s] = 1;
33         while (!Q.empty()) {
34             int x = Q.front(); Q.pop();
```

```

35         for (int& i: G[x]) {
36             E &e = edges[i];
37             if (!d[e.to] && e.cp > 0) {
38                 d[e.to] = d[x] + 1;
39                 Q.push(e.to);
40             }
41         }
42     }
43     return d[t];
44 }
45
46 ll DFS(int u, ll cp) {
47     if (u == t || !cp) return cp;
48     ll tmp = cp, f;
49     for (int& i = cur[u]; i < G[u].size(); i++) {
50         E& e = edges[G[u][i]];
51         if (d[u] + 1 == d[e.to]) {
52             f = DFS(e.to, min(cp, e.cp));
53             e.cp -= f;
54             edges[G[u][i] ^ 1].cp += f;
55             cp -= f;
56             if (!cp) break;
57         }
58     }
59     return tmp - cp;
60 }
61
62 ll go() {
63     ll flow = 0;
64     while (BFS()) {
65         memset(cur, 0, sizeof cur);
66         flow += DFS(s, INF);
67     }
68     return flow;
69 }
70 } DC;

```

费用流

```

1  struct E {
2      int from, to, cp, v;
3      E() {}

```

```

4      E(int f, int t, int cp, int v) : from(f), to(t), cp(cp),
      v(v) {}
5  };
6
7  struct MCMF {
8      int n, m, s, t;
9      vector<E> edges;
10     vector<int> G[M];
11     bool inq[M];
12     int d[M], p[M], a[M];
13
14     void init(int _n, int _s, int _t) {
15         n = _n; s = _s; t = _t;
16         FOR (i, 0, n + 1) G[i].clear();
17         edges.clear(); m = 0;
18     }
19
20     void addedge(int from, int to, int cap, int cost) {
21         edges.emplace_back(from, to, cap, cost);
22         edges.emplace_back(to, from, 0, -cost);
23         G[from].push_back(m++);
24         G[to].push_back(m++);
25     }
26
27     bool BellmanFord(int &flow, int &cost) {
28         FOR (i, 0, n + 1) d[i] = INF;
29         memset(inq, 0, sizeof inq);
30         d[s] = 0, a[s] = INF, inq[s] = true;
31         queue<int> Q; Q.push(s);
32         while (!Q.empty()) {
33             int u = Q.front(); Q.pop();
34             inq[u] = false;
35             for (int& idx: G[u]) {
36                 E &e = edges[idx];
37                 if (e.cp && d[e.to] > d[u] + e.v) {
38                     d[e.to] = d[u] + e.v;
39                     p[e.to] = idx;
40                     a[e.to] = min(a[u], e.cp);
41                     if (!inq[e.to]) {
42                         Q.push(e.to);
43                         inq[e.to] = true;
44                     }
45                 }
46             }

```



```
47         }
48         if (d[t] == INF) return false;
49         flow += a[t];
50         cost += a[t] * d[t];
51         int u = t;
52         while (u != s) {
53             edges[p[u]].cp -= a[t];
54             edges[p[u] ^ 1].cp += a[t];
55             u = edges[p[u]].from;
56         }
57         return true;
58     }
59
60     int go() {
61         int flow = 0, cost = 0;
62         while (BellmanFord(flow, cost));
63         return cost;
64     }
65 } MM;
```