

```

1  {
2      "IO": {
3          "prefix": "io",
4          "body": [
5              "namespace IO{",
6              "    char ibuf[1<<21],*ip=ibuf,*ip_=ibuf;",
7              "    char obuf[1<<21],*op=obuf,*op_=obuf+(1<<21);",
8              "    inline char gc(){",
9              "        if(ip!=ip_)return *ip++;",
10             "        ip=ibuf;ip_=ip+fread(ibuf,1,1<<21,stdin);",
11             "        return ip==ip_?EOF:*ip++;",
12             "    }",
13             "    inline void pc(char c){",
14             "        if(op==op_)fwrite(obuf,1,1<<21,stdout),op=obuf;",
15             "        *op++=c;",
16             "    }",
17             "    inline int read(){",
18             "        int x=0,ch=gc(),w=1;",
19             "        for(;ch<'0' || ch>'9';ch=gc())if(ch=='-')w=-1;",
20             "        for(;ch>='0' && ch<='9';ch=gc())x=x*10+ch-48;",
21             "        return w*x;",
22             "    }",
23             "    template<class I>",
24             "    inline void write(I x){",
25             "        if(x<0)pc('-'),x=-x;",
26             "        if(x>9)write(x/10);pc(x%10+'0');",
27             "    }",
28             "    class flusher_{",
29             "    public:",
30             "        ~flusher_(){if(op!=obuf)fwrite(obuf,1,op-obuf,stdout);}",
31             "    }IO_flusher;",
32             "}",
33             "using namespace IO;"
34         ],
35         "description": "IO"
36     },
37     "dijkstra": {
38         "prefix": "dij",
39         "body": [
40             "namespace dij{",
41             "    const int N = 1e7 + 10;",
42             "    const int INF = 0x3f3f3f3f;",
43             "    struct Edge",
44             "    {",
45             "        int u, v, w, ne;",
46             "    }e[N << 2];",
47             "    int h[N], cnt, dist[N], vis[N];",
48             "",
49             "    void add(int u, int v, int w)",
50             "    {",
51             "        e[cnt].u = u;",
52             "        e[cnt].v = v;",
53             "        e[cnt].w = w;",
54             "        e[cnt].ne = h[u];",
55             "        h[u] = cnt++;",
56             "    }",
57             "",
58             "    void dijkstra(int s, int d[]){",
59             "        memset(vis, 0, sizeof vis);",
60             "        memset(d, 0x3f, sizeof dist);",
61             "        priority_queue <PII, vector<PII>, greater<PII>> q;",
62             "        d[s] = 0;",
63             "        q.push({0, s});",
64             "        while(q.size())",
65             "        {",
66             "            auto ns = q.top();",
67             "            q.pop();",
68             "            int x = ns.second;",
69             "            if(vis[x]) continue;",
70             "            vis[x] = 1;",
71             "            for(int i = h[x]; ~i ; i = e[i].ne)",

```

```

72         "                {",
73         "                int to = e[i].v;",
74         "                if(d[to] > d[x] + e[i].w)",
75         "                {",
76         "                    d[to] = d[x] + e[i].w;",
77         "                    q.push({d[to], to});",
78         "                }",
79         "            }",
80         "        }",
81         "    }",
82     "}",
83     "using namespace dij;"
84 ],
85     "description": "dijkstra"
86 },
87 "二分-找右": {
88     "prefix": "erfen1",
89     "body": [
90         "int l = 1, r = n;",
91         "while(l < r){",
92         "\tint mid = l + r + 1 >> 1;",
93         "\tif(check(mid)) l = mid;",
94         "\telse r = mid - 1;",
95         "}",
96         "cout << l << endl;"
97     ],
98     "description": "二分-找右"
99 },
100 "二分-找左": {
101     "prefix": "erfen2",
102     "body": [
103         "int l = 1, r = n;",
104         "while(l <= r){",
105         "\tint mid = l + r >> 1;",
106         "\tif(check(mid)) r = mid - 1;",
107         "\telse l = mid + 1;",
108         "}",
109         "if(l >= 1 && l <= n) cout << l << endl;",
110         "else //l不存在"
111     ],
112     "description": "二分-找左"
113 },
114 "二进制枚举": {
115     "prefix": "binary_enumeration",
116     "body": [
117         "void binary_enumeration(int n, int a[]){",
118         "    int sum = 0;",
119         "    for(int i = 0; i < (1 << n); i++){",
120         "        sum = 0;",
121         "        for(int j = 0; j < n; j++){",
122         "            if( i & (1 << j) ){",
123         "                sum += a[j];",
124         "            }",
125         "        }",
126         "        //if() {",
127         "            // 满足什么条件",
128         "        }",
129         "    }",
130     ],
131 ],
132     "description": "二进制枚举"
133 },
134 "求因子和": {
135     "prefix": "yinzi_sum",
136     "body": [
137         "//以线性筛为基础",
138         "",
139         "int jet_num[N]; //用来记录素数的幂是多少的",
140         "",
141         "LL get_yin_zi_sum(LL n)",
142         "{",

```

```

143         "    LL ans = 1;",
144         "    for(int i = 1; (LL) prime[i] * prime[i] <= n; i++){",
145         "        if(!(n % prime[i])){",
146         "            LL jet = 1, sum = 1;",
147         "            while(!(n % prime[i])){",
148         "                jet_num[prime[i]]++;",
149         "                jet *= prime[i];",
150         "                sum += jet;",
151         "                n /= prime[i];",
152         "            }",
153         "            ans *= sum;",
154         "        }",
155         "    }",
156         "    if(n > 1) {",
157         "        jet_num[n]++;",
158         "        ans *= (n + 1);",
159         "    }",
160         "    return ans;",
161     "}",
162     ""
163 ],
164 "description": "求因子和"
165 },
166 "求因子个数": {
167     "prefix": "yinzi_num",
168     "body": [
169         "//以线性筛为基础",
170         "LL get_yin_zi_num(LL n)",
171         "{",
172         "    LL ans = 1;",
173         "    for(int i = 1; (LL) prime[i] * prime[i] <= n; i++){",
174         "        if(!(n % prime[i])){",
175         "            LL cnt = 0;",
176         "            while(!(n % prime[i])){",
177         "                cnt ++;",
178         "                n /= prime[i];",
179         "            }",
180         "            ans *= (1 + cnt);",
181         "        }",
182         "    }",
183         "    if(n > 1) ans *= 2;",
184         "    return ans;",
185         "}",
186         ""
187     ],
188     "description": "求因子个数"
189 },
190 "素因子分解": {
191     "prefix": "prime_factorization",
192     "body": [
193         "vector < int > v;",
194         "for(int j = 2; j <= a / j; j++){",
195         "    while(a % j == 0){",
196         "        a /= j;",
197         "        v.push_back(j);",
198         "    }",
199         "}",
200         "if(a > 1) v.push_back(a);",
201         ""
202     ],
203     "description": "素因子分解"
204 },
205 "01背包": {
206     "prefix": "bag_01",
207     "body": [
208         "for(int i = 1; i <= n; i++){",
209         "    for(int j = v; j >= c[i]; j--){",
210         "        dp[j] = max(dp[j], dp[j - c[i]] + w[i]);",
211         "    }",
212         "}"
213     ],
214     "description": "01背包"
215 }

```

```

214         "description": "01背包"
215     },
216     "二分图匹配": {
217         "prefix": "erfen_graph_matching",
218         "body": [
219             "bool vis[N];",
220             "int match[N];",
221             "bool dfs(int u)",
222             "{",
223             "    for(int i = h[u]; ~i; i = e[i].ne){",
224             "        int v = e[i].v;",
225             "        if(!vis[v]){",
226             "            vis[v] = true;",
227             "            if(!match[v] || dfs(match[v])){",
228             "                match[v] = u;",
229             "                return true;",
230             "            }",
231             "        }",
232             "    }",
233             "    return false;",
234             "}"
235         ],
236         "description": "二分图匹配"
237     },
238     "链式前向星": {
239         "prefix": "graph",
240         "body": [
241             "const int N = 1e6 + 10;",
242             "struct Edge",
243             "{",
244             "    int v, ne;",
245             "}e[N << 2];",
246             "int h[N];",
247             "int cnt;",
248             "void add(int u, int v)",
249             "{",
250             "    e[cnt].v = v;",
251             "    e[cnt].ne = h[u];",
252             "    h[u] = cnt++;",
253             "}",
254             "void init(){",
255             "    memset(h, -1, sizeof(h));",
256             "    cnt = 0;",
257             "}"
258         ],
259         "description": "链式前向星"
260     },
261     "矩阵快速幂": {
262         "prefix": "q_Matrix",
263         "body": [
264             "namespace Q_Matrix{",
265             "    const int MAX = 10;",
266             "    typedef struct{",
267             "        LL m[MAX][MAX];",
268             "    }Matrix;",
269             "",
270             "    Matrix P;//构造出的矩阵",
271             "",
272             "    LL k,mod,a[MAX];",
273             "    Matrix matrixmul(Matrix a,Matrix b) //矩阵乘法",
274             "{",
275             "    int i,j,k;",
276             "    Matrix c;",
277             "    for (i = 0 ; i < MAX; i++)",
278             "        for (j = 0; j < MAX;j++)",
279             "        {",
280             "            c.m[i][j] = 0;",
281             "            for (k = 0; k < MAX; k++)",
282             "                c.m[i][j] =(c.m[i][j]+(a.m[i][k]*b.m[k][j]))%mod;",
283             "            c.m[i][j] %=mod;",
284             "        }",

```

```

285         "        return c;",
286     "    }",
287 "    ",
288     "    Matrix quickpow(Matrix m , LL n)",
289     "{",
290     "        Matrix b;//单位矩阵在这构造也可以",
291     "        for(int i=0;i<MAX;i++)",
292     "            for(int j=0;j<MAX;j++)",
293     "            {",
294     "                if(i==j)b.m[i][j]=1;",
295     "                else b.m[i][j]=0;",
296     "            }",
297     "        while (n >= 1)",
298     "        {",
299     "            if (n & 1)",
300     "                b = matrixmul(b,m);",
301     "            n = n >> 1;",
302     "            m = matrixmul(m,m);",
303     "        }",
304     "        return b;",
305     "    }",
306 "}",
307 "using namespace Q_Matrix;",
308 ""
309 ],
310 "description": "矩阵快速幂"
311 },
312 "树状数组": {
313     "prefix": "tree_array",
314     "body": [
315         "const int N = 1e6 + 10;",
316         "int tree[N], n;",
317         "void add(LL x,LL d)",
318         "{",
319         "    while(x <= n){",
320         "        tree[x] += d;",
321         "        x += lowbit(x);//查询x的后继们",
322         "    }",
323     "}",
324     "LL sum(LL x)//前缀和",
325     "{",
326     "    LL sum = 0;",
327     "    while(x > 0){",
328     "        sum += tree[x];",
329     "        x -= lowbit(x);//查询x的前驱们",
330     "    }",
331     "    return sum;",
332     "}",
333     ""
334 ],
335 "description": "树状数组"
336 },
337 "最小生成树": {
338     "prefix": "kruskal",
339     "body": [
340         "namespace kruskal{",
341         "    int n, m, u, v;",
342         "    LL w, ans, cnt;",
343         "    const int N = 1e6 + 10;",
344         "    int fa[N];",
345         "    struct sa{",
346         "        int u,v;",
347         "        LL w;",
348         "    }e[N];",
349     "",
350     "    bool cmp(struct sa x,struct sa y){",
351     "        return x.w < y.w;",
352     "    }",
353     "",
354     "    void init(){",
355     "        // n点, m条边",

```

```

356         "        cin >> n >> m;",
357         "        for(int i = 1; i <= m; i++) cin >> e[i].u >> e[i].v >> e[i].w;",
358         "        sort(e + 1, e + 1 + m, cmp);",
359         "        for(int i = 1; i <= n; i++) fa[i] = i;",
360         "    }",
361     "",
362     "    int find(int x){",
363         "        return fa[x] == x ? x : fa[x] = find(fa[x]);",
364     "}",
365     "",
366     "",
367     "    void solve(){",
368         "        for(int i = 1; i <= m; i++){",
369             "            if(cnt == n - 1) break;",
370             "            w = e[i].w;",
371             "            u = find(e[i].u);",
372             "            v = find(e[i].v);",
373             "            if(u != v){",
374                 "                fa[u] = v;",
375                 "                ans += w;",
376                 "                cnt ++;",
377             "            }",
378             "            if(cnt == n - 1) break;",
379         "        }",
380     "}",
381 "}",
382 "using namespace kruskal;"
383 ],
384 "description": "最小生成树"
385 },
386 "并查集": {
387     "prefix": "DSU",
388     "body": [
389         "namespace union_set{",
390             "    const int N = 1e6 + 10;",
391             "    int fa[N], n;",
392             "    void init ()",
393             "{",
394                 "        for(int i = 1; i <= n; i++) fa[i] = i;",
395             "    }",
396             "    int find(int x){",
397                 "        return fa[x] == x ? x : fa[x] = find(fa[x]);",
398             "    }",
399             "    void join(int a,int b)",
400             "{",
401                 "        int a1 = find(a),b1 = find(b);",
402                 "        if(a1 != b1) fa[a1] = b1;",
403             "    }",
404         "}",
405         "using namespace union_set;"
406     ],
407     "description": "并查集"
408 },
409 "素数筛": {
410     "prefix": "prime_sieve",
411     "body": [
412         "namespace prime_sieve",
413         "{",
414             "    const int N = 5e4 + 5;",
415             "    int cnt, prime[N];",
416             "    bool flag[N];",
417             "    inline void init()",
418             "{",
419                 "        memset(flag, 1, sizeof(flag));",
420                 "        flag[0] = flag[1] = 0;",
421                 "        for (int i = 2; i <= N; i++)",
422                 "{",
423                     "            if (flag[i])",
424                     "{",
425                         "                prime[++cnt] = i;",
426                     "            }",

```

```

427         "                for (int j = 1; j <= cnt && 1LL * prime[j] * i <= N; j++)",
428         "                {",
429         "                    flag[prime[j] * i] = 0;",
430         "                    if (i % prime[j] == 0)",
431         "                        break;",
432         "                }",
433         "            }",
434         "        }",
435         "    }",
436         "using namespace prime_sieve;"
437     ],
438     "description": "素数筛"
439 },
440 "快速幂": {
441     "prefix": "q_pow",
442     "body": [
443         "LL qm (LL a, LL b, LL c){",
444         "    LL ret = 1 % c;",
445         "    while(b){",
446         "        if(b & 1)",
447         "            ret = ret * a % c;",
448         "        a = a * a % c;",
449         "        b = b >> 1;",
450         "    }",
451         "    return ret;",
452     "}",
453     ],
454     "description": "快速幂"
455 },
456 "按权相加法": {
457     "prefix": "jinzhi",
458     "body": [
459         "int base_conversion(string str, int k)",
460         "{",
461         "    int ans = 0;",
462         "    for(int i = 0; i < str.size(); i++){",
463         "        ans = ans * k + (str[i] - '0');",
464         "    }",
465         "    return ans;",
466     "}",
467     ],
468     ],
469     "description": "按权相加法"
470 },
471 "spfa": {
472     "prefix": "spfa",
473     "body": [
474         "namespace spfa{",
475         "    const int N = 1e7 + 10;",
476         "    const int inf = 0x3f3f3f3f;",
477         "    struct Edge",
478         "    {",
479         "        int u, v, w, ne;",
480         "    }e[N << 2];",
481     "    ",
482     "    int h[N], number[N];",
483     "    int cnt;",
484     "    ",
485     "    void init(){",
486     "        memset(h, -1, sizeof(h));",
487     "        memset(number, 0, sizeof(number));",
488     "        cnt = 0;",
489     "    }",
490     "    ",
491     "    void add(int u, int v, int w)",
492     "    {",
493     "        e[cnt].u = u;",
494     "        e[cnt].v = v;",
495     "        e[cnt].w = w;",
496     "        e[cnt].ne = h[u];",
497     "        h[u] = cnt++;",

```

```

498     }",
499     "    int dis[N], vis[N], n;",
500     "",
501     "    int spfa(int s, int d[]){",
502     "        queue < int > q;",
503     "        memset(vis, 0, sizeof(vis));",
504     "        for(int i = 0; i <= n; i++) d[i] = inf; //最短路",
505     "        d[s] = 0;",
506     "        vis[s] = 1;",
507     "        q.push(s);",
508     "        while(!q.empty()){",
509     "            int u = q.front();",
510     "            q.pop();",
511     "            vis[u] = 0;",
512     "            for(int i = h[u]; ~i; i = e[i].ne){",
513     "                int v = e[i].v, w = e[i].w;",
514     "                if(d[v] > d[u] + w){ //最短路",
515     "                    d[v] = d[u] + w;",
516     "                    if( !vis[v] ){",
517     "                        vis[v] = 1;",
518     "                        q.push(v);",
519     "                        number[v] ++;",
520     "                        if(number[v] == n) return 0;",
521     "                    }",
522     "                }",
523     "            }",
524     "        }",
525     "        return 1;",
526     "    }",
527     "}",
528     "using namespace spfa;"
529 ],
530 "description": "spfa"
531 },
532 "素数筛+欧拉函数": {
533     "prefix": "prime_sieve_phi",
534     "body": [
535         "namespace prime_sieve_phi{",
536         "    const int N = 5e4 + 5;",
537         "    int cnt, ans, prime[N], pre[N], phi[N];",
538         "    bool flag[N];",
539         "    inline void init()",
540         "    {",
541         "        memset(flag, 1, sizeof(flag));",
542         "        flag[1] = 0;",
543         "        cnt = 0;",
544         "        phi[1] = 1;",
545         "        for(int i = 2; i <= N; i++)",
546         "        {",
547         "            if(flag[i])",
548         "            {",
549         "                prime[++cnt] = i;",
550         "                pre[i] = cnt;",
551         "                phi[i] = i - 1;",
552         "            }",
553         "            for(int j = 1; j <= cnt && 1LL * prime[j] * i <= N; j++)",
554         "            {",
555         "                flag[prime[j] * i] = 0;",
556         "                if(i % prime[j] == 0) {",
557         "                    phi[i * prime[j]] = phi[i] * prime[j];",
558         "                    break;",
559         "                }",
560         "                phi[i * prime[j]] = phi[i] * (prime[j] - 1);",
561         "            }",
562         "        }",
563         "    }",
564     "}",
565     "using namespace prime_sieve_phi;"
566 ],
567 "description": "素数筛+欧拉函数"
568 },

```


[illegible]

```

640         "\t\t}",
641         "\t}",
642         "}",
643         "using namespace prime_sieve_sum_yinzi;"
644     ],
645     "description": "线性筛+因子个数,因d也为积性函数,故也可以线性筛出来"
646 },
647 "线性筛+莫比乌斯+欧拉+前缀和": {
648     "prefix": "prime_sieve_phi_mu",
649     "body": [
650         "namespace prime_sieve_phi_mu{",
651         "    const int N = 5e4 + 5;",
652         "    int cnt, ans, prime[N], pre[N], phi[N], mu[N];",
653         "    bool flag[N];",
654         "    LL sum_phi[N], sum_mu[N];",
655         "    inline void init()",
656         "    {",
657         "        memset(flag, 1, sizeof(flag));",
658         "        flag[1] = 0;",
659         "        cnt = 0;",
660         "        phi[1] = 1;",
661         "        mu[1] = 1;",
662         "        for(int i = 2; i <= N; i++){",
663         "            if(flag[i]){",
664         "                prime[++cnt] = i;",
665         "                pre[i] = cnt;",
666         "                phi[i] = i - 1;",
667         "                mu[i] = -1;",
668         "            }",
669         "            for(int j = 1; j <= cnt && 1LL * prime[j] * i <= N; j++){",
670         "                flag[prime[j] * i] = 0;",
671         "                if(i % prime[j] == 0) {",
672         "                    phi[i * prime[j]] = phi[i] * prime[j];",
673         "                    mu[prime[j] * i] = 0;",
674         "                    break;",
675         "                }",
676         "                phi[i * prime[j]] = phi[i] * (prime[j] - 1);",
677         "                mu[prime[j] * i] = -mu[i];",
678         "            }",
679         "        }",
680         "        for(int i = 1; i <= N; i++){",
681         "            sum_phi[i] = sum_phi[i - 1] + phi[i];",
682         "            sum_mu[i] = sum_mu[i - 1] + mu[i];",
683         "        }",
684         "    }",
685     "}",
686     "using namespace prime_sieve_phi_mu;"
687 ],
688     "description": "线性筛+莫比乌斯+欧拉+前缀和"
689 },
690 "杜教筛(欧拉+莫比乌斯)": {
691     "prefix": "djs",
692     "body": [
693         "unordered_map < LL, LL > phi_w;",
694         "unordered_map < LL, LL > mu_w;",
695         "",
696         "LL djs_phi(LL x){",
697         "    if(x <= N) return sum_phi[x];",
698         "    if(phi_w[x]) return phi_w[x];",
699         "    LL ans = x * (x + 1) / 2;",
700         "    for(LL l = 2, r; l <= x; l = r + 1){",
701         "        r = x / (x / l);",
702         "        ans -= (r - l + 1) * (djs_phi(x / l));",
703         "    }",
704         "    return phi_w[x] = ans;",
705     "}",
706     "",
707     "LL djs_mu(LL x){",
708         "    if(x <= N) return sum_mu[x];",
709         "    if(mu_w[x]) return mu_w[x];",
710         "    LL ans = 1;",

```

```

711         "    for(LL l = 2, r; l <= x; l = r + 1){",
712         "        r = x / (x / l);",
713         "        ans -= (r - l + 1) * (djs_mu(x / l));",
714         "    }",
715         "    return mu_w[x] = ans;",
716     }"
717 ],
718     "description": "杜教筛（欧拉+莫比乌斯）"
719 },
720 "模运算": {
721     "prefix": "mod",
722     "body": [
723         "const LL mod = 1e9 + 7;",
724         "LL mul(LL x, LL y){return 1LL * x * y % mod;}",
725         "LL dec(LL x, LL y){return x >= y ? x - y : x + mod - y;}",
726         "LL add(LL x, LL y){return x + y >= mod ? x + y - mod : x + y;}",
727         "LL pmod(LL x) {return (x + mod) % mod;}",
728     ]
729 },
730     "description": "模运算"
731 },
732 "逆元费马小定理": {
733     "prefix": "inv_fm",
734     "body": [
735         "LL inv_fm(LL n, LL p) { return qm(n, p - 2, p); }"
736     ],
737     "description": "逆元费马小定理"
738 },
739 "埃氏筛": {
740     "prefix": "prime_aishi_sieve",
741     "body": [
742         "namespace prime_aishi_sieve{",
743         "    const int N = 1E6 + 10;",
744         "    bool vis[N];",
745         "    int prime[N], cnt;",
746         "",
747         "    void aishi_sieve(){",
748         "        for(int i = 0; i <= N; i++) vis[i] = 1;",
749         "        vis[0] = vis[1] = 0;",
750         "        for(int i = 2; i <= N; i++){",
751         "            if(vis[i]){",
752         "                prime[++cnt] = i;",
753         "                for(int j = i + i; j <= N; j += i){",
754         "                    vis[j] = 0;",
755         "                }",
756         "            }",
757         "        }",
758         "    }",
759         "}",
760         "using namespace prime_aishi_sieve;"
761     ],
762     "description": "埃氏筛"
763 },
764 "线段树加法": {
765     "prefix": "xds_add",
766     "body": [
767         "namespace xds_add{",
768         "    const int N = 1e6 + 10;",
769         "    LL a[N << 2], tr[N << 2], add_tag[N << 2], k;",
770         "    int n, x, y;",
771         "",
772         "    inline void pushup(int i)",
773         "    {",
774         "        tr[i] = tr[ls] + tr[rs];",
775         "    }",
776         "",
777         "    void bulid(int i, int l, int r)",
778         "    {",
779         "        add_tag[i] = 0;",
780         "        if(l == r){",
781         "            tr[i] = a[l];",

```

```

782         "        return;\"",
783     "    }\"",
784     "        int mid = (l + r) >> 1;\"",
785     "        bulid(ls, l, mid);\"",
786     "        bulid(rs, mid + 1, r);\"",
787     "        pushup(i);\"",
788     "    }\"",
789     "\"\",
790     "    inline void ADD(int i, int l, int r, LL k)\"",
791     "    {\"",
792     "        add_tag[i] = (add_tag[i] + k);\"",
793     "        tr[i] = (tr[i] + (r - l + 1) * k);\"",
794     "    }\"",
795     "\"\",
796     "    inline void pushdown(int i, int l, int r, int mid)\"",
797     "    {\"",
798     "        if( (!add_tag[i]) ) return;\"",
799     "        ADD(ls, l, mid, add_tag[i]);\"",
800     "        ADD(rs, mid + 1, r, add_tag[i]);\"",
801     "        add_tag[i] = 0;\"",
802     "    }\"",
803     "\"\",
804     "    inline void update_ADD (int i, int l, int r, int x, int y, LL k)\"",
805     "    {\"",
806     "        if(l > y || r < x) return;\"",
807     "        if(l >= x && r <= y) return ADD(i, l, r, k);\"",
808     "        int mid = (l + r) >> 1;\"",
809     "        pushdown(i, l, r, mid);\"",
810     "        update_ADD(ls, l, mid, x, y, k);\"",
811     "        update_ADD(rs, mid + 1, r, x, y, k);\"",
812     "        pushup(i);\"",
813     "    }\"",
814     "\"\",
815     "    LL query(int i, int l, int r, int x, int y)\"",
816     "    {\"",
817     "        LL res = 0;\"",
818     "        if(l > y || r < x) return 0;\"",
819     "        if(l >= x && r <= y) return tr[i];\"",
820     "        int mid = (l + r) >> 1;\"",
821     "        pushdown(i, l, r, mid);\"",
822     "        if(x <= mid) res = res + query(ls, l, mid, x, y);\"",
823     "        if(y > mid) res = res + query(rs, mid + 1, r, x, y);\"",
824     "        return res;\"",
825     "    }\"",
826     "}\"\",
827     "using namespace xds_add;\"
828 ],
829 "description": "线段树加法"
830 },
831 "线段树+最大值": {
832     "prefix": "xds_max",
833     "body": [
834         "namespace xds_max{\"",
835         "    const int N = 1e6 + 10;\"",
836         "    LL a[N << 2], tr[N << 2];\"",
837         "    int n;\"",
838         "\"\",
839         "    inline void pushup(int i)\"",
840         "    {\"",
841         "        tr[i] = max(tr[ls], tr[rs]);\"",
842         "    }\"",
843         "\"\",
844         "    void bulid(int i, int l, int r)\"",
845         "    {\"",
846         "        if(l == r){\"",
847         "            tr[i] = a[l];\"",
848         "            return;\"",
849         "        }\"",
850         "        int mid = (l + r) >> 1;\"",
851         "        bulid(ls, l, mid);\"",
852         "        bulid(rs, mid + 1, r);\"",

```

```

853         "        pushup(i);",
854     "    }",
855 "    ",
856     "    inline void update (int i, int l, int r, int x, LL y)",
857     "{",
858         "        if(l > x || r < x) return;",
859         "        if(l == x && l == r) {tr[i] = y; return;}",
860         "        int mid = (l + r) >> 1;",
861         "        update(ls, l, mid, x, y);",
862         "        update(rs, mid + 1, r, x, y);",
863         "        pushup(i);",
864     "    }",
865 "    ",
866     "    LL query(int i, int l, int r, int x, int y)",
867     "{",
868         "        LL res = 0;",
869         "        if(l > y || r < x) return 0;",
870         "        if(l >= x && r <= y) return tr[i];",
871         "        int mid = (l + r) >> 1;",
872         "        if(x <= mid) res = max(res, query(ls, l, mid, x, y));",
873         "        if(y > mid) res = max(res, query(rs, mid + 1, r, x, y));",
874         "        return res;",
875     "    }",
876 "}",
877 "using namespace xds_max;"
878 ],
879 "description": "线段树+最大值"
880 },
881 "线段树+最小值": {
882     "prefix": "xds_min",
883     "body": [
884         "namespace xds_min{",
885         "    const int N = 1e6 + 10;",
886         "    LL a[N << 2], tr[N << 2];",
887         "    int n;",
888     "    ",
889     "    inline void pushup(int i)",
890     "    {",
891         "        tr[i] = min(tr[ls], tr[rs]);",
892     "    }",
893 "    ",
894     "    void bulid(int i, int l, int r)",
895     "    {",
896         "        if(l == r){",
897             "            tr[i] = a[l];",
898             "            return;",
899         "        }",
900         "        int mid = (l + r) >> 1;",
901         "        bulid(ls, l, mid);",
902         "        bulid(rs, mid + 1, r);",
903         "        pushup(i);",
904     "    }",
905 "    ",
906     "    inline void update (int i, int l, int r, int x, LL y)",
907     "    {",
908         "        if(l > x || r < x) return;",
909         "        if(l == x && l == r) {tr[i] = y; return;}",
910         "        int mid = (l + r) >> 1;",
911         "        update(ls, l, mid, x, y);",
912         "        update(rs, mid + 1, r, x, y);",
913         "        pushup(i);",
914     "    }",
915 "    ",
916     "    LL query(int i, int l, int r, int x, int y)",
917     "    {",
918         "        LL res = INF;",
919         "        if(l > y || r < x) return 0;",
920         "        if(l >= x && r <= y) return tr[i];",
921         "        int mid = (l + r) >> 1;",
922         "        if(x <= mid) res = min(res, query(ls, l, mid, x, y));",
923         "        if(y > mid) res = min(res, query(rs, mid + 1, r, x, y));",

```

```

924         "        return res;",
925         "    }",
926     "}",
927     "using namespace xds_min;"
928 ],
929     "description": "线段树+最小值"
930 },
931     "KMP": {
932         "prefix": "KMP",
933         "body": [
934             "namespace Kmp{",
935             "    const int N = 1E7 + 10;",
936             "    string t;",
937             "    int Next[N];",
938             "    void get_next(string t){",
939                 "        int i = 0, j = -1;",
940                 "        int n = t.length();",
941                 "        Next[0] = -1;",
942                 "        while (i < n){",
943                     "            if (j == -1 || t[i] == t[j]){",
944                         "                i++, j++;",
945                         "                Next[i] = j;",
946             "            }",
947             "            else",
948                 "                j = Next[j];",
949             "        }",
950         "    }",
951         "    bool kmp(string s, string ss){",
952             "        int i = 0, j = 0;",
953             "        int slen = s.length(), sslen = ss.length();",
954             "        get_next(ss);",
955             "        while (i < slen && j < sslen){",
956                 "            if (j == -1 || s[i] == ss[j]){",
957                     "                i++, j++; //i是主串下标, j是模式串下标",
958             "            }",
959             "            else",
960                 "                j = Next[j]; //如果不匹配了, 就移动模式串",
961             "        }",
962             "        if (j == sslen)",
963                 "            return 1;",
964             "        else",
965                 "            return 0;",
966         "    }",
967     "}",
968     "using namespace Kmp;"
969 ],
970     "description": "KMP"
971 },
972     "快速幂+快速乘": {
973         "prefix": "q_pow_mm",
974         "body": [
975             "namespace q_pow_mm{",
976             "    LL mm(LL a, LL b, LL m){",
977                 "        LL ret = 0;",
978                 "        while(b){",
979                     "            if(b & 1) ",
980                         "                ret = (ret + a) % m;",
981                     "            a = (a * 2) % m;",
982                     "            b >>= 1;",
983                 "        }",
984                 "        return ret;",
985             "    }",
986             "    LL q (LL a, LL b){",
987                 "        LL ret = 1;",
988                 "        while(b){",
989                     "            if(b & 1)",
990                 "                ret = ret * a;",
991                 "                a = a * a;",
992                 "                b = b >> 1;",
993             "        }",
994             "        return ret;",

```

```

995         "    }",
996         "    LL qm (LL a, LL b, LL c){",
997         "        a = a % c;",
998         "        LL ret = 1 % c;",
999         "        while(b){",
1000         "            if(b & 1)",
1001         "                ret = mm(ret, a, c) % c;",
1002         "            a = mm(a, a, c) % c;",
1003         "            b = b >> 1;",
1004         "        }",
1005         "        return ret;",
1006     "    }",
1007 "}",
1008 "using namespace q_pow_mm;"
1009 ],
1010 "description": "快速幂+快速乘"
1011 },
1012 "欧几里得+拓展": {
1013     "prefix": "gcd+exgcd",
1014     "body": [
1015         "LL gcd(LL a, LL b) {return b == 0 ? a : gcd(b, a % b);}",
1016         "",
1017         "LL exgcd(LL a, LL b, LL &x, LL &y){",
1018         "    if(!b){",
1019         "        x = 1;",
1020         "        y = 0;",
1021         "        return a;",
1022         "    }",
1023         "    LL r = exgcd(b, a % b, x, y);",
1024         "    LL tmp = y;",
1025         "    y = x - (a / b) * y;",
1026         "    x = tmp;",
1027         "    return r;",
1028     "]"
1029 ],
1030 "description": "欧几里得+拓展"
1031 },
1032 "逆元exgcd": {
1033     "prefix": "inv_exgcd",
1034     "body": [
1035         "LL exgcd(LL a, LL b, LL &x, LL &y){",
1036         "    if(!b){",
1037         "        x = 1;",
1038         "        y = 0;",
1039         "        return a;",
1040         "    }",
1041         "    LL r = exgcd(b, a % b, x, y);",
1042         "    LL tmp = y;",
1043         "    y = x - (a / b) * y;",
1044         "    x = tmp;",
1045         "    return r;",
1046     "}",
1047     "",
1048     "LL inv_exgcd(LL n, LL p){",
1049     "    LL d, x, y;",
1050     "    d = exgcd(n, p, x, y);",
1051     "    if(d == 1)",
1052     "        return (x % p + p) % p;",
1053     "    else",
1054     "        return -1;",
1055     "}"
1056 ],
1057 "description": "逆元exgcd"
1058 },
1059 "逆元欧拉定理": {
1060     "prefix": "inv_euler",
1061     "body": [
1062         "LL euler(LL n){",
1063         "    LL ans = n;",
1064         "    for(int i = 2; i * i <= n; i ++){",
1065         "        if(!(n % i)){",

```

```

1066         "            ans = ans / i * (i - 1);",
1067         "            while(!(n % i)) n /= i;",
1068         "        }",
1069         "    }",
1070         "    if(n > 1) ans = ans / n * (n - 1);",
1071         "    return ans;",
1072     "}",
1073     "",
1074     "LL inv_euler(LL n, LL p) {return qm(n, euler(p) - 1, p);}"
1075 ],
1076 "description": "逆元欧拉定理"
1077 },
1078 "ls与rs宏定义": {
1079     "prefix": "ls",
1080     "body": [
1081         "#define ls                i << 1",
1082         "#define rs                i << 1 | 1"
1083     ],
1084     "description": "ls与rs宏定义"
1085 },
1086 "hashString": {
1087     "prefix": "hashString",
1088     "body": [
1089         "const int base = 131, N = 1e5 + 10;",
1090         "ULL h[N], p[N] = {1};",
1091         "",
1092         "ULL get(int l, int r){",
1093         "    return h[r] - h[l - 1] * p[r - l + 1];",
1094         "}"
1095     ],
1096     "description": "hashString"
1097 },
1098 "杨辉三角": {
1099     "prefix": "yanghuisanjiang",
1100     "body": [
1101         "void init(){",
1102         "    a[0][0] = 1;",
1103         "    for(int i = 0; i <= N; ++ i){",
1104         "        a[i][0] = a[i][i] = 1;",
1105         "        for(int j = 1; j <= i / 2; ++ j){",
1106         "            a[i][j] = a[i][i - j] = add(a[i - 1][j - 1], a[i - 1][j]);",
1107         "        }",
1108         "    }",
1109     "}",
1110     ""
1111 ],
1112 "description": "杨辉三角"
1113 },
1114 "Lucas求大组合数": {
1115     "prefix": "lucas",
1116     "body": [
1117         "LL inv_fm(LL n, LL p) {",
1118         "    return qm(n, p - 2, p);",
1119         "}",
1120         "",
1121         "LL C(LL n, LL m, LL p){",
1122         "    return m > n ? 0 : mul(fac[n], mul(inv_fm(fac[m], p), inv_fm(fac[n - m], p)));",
1123         "}",
1124         "",
1125         "LL lucas(LL n, LL m, LL p){",
1126         "    if(n < m) return 0;",
1127         "    if(n == m || m == 0) return 1;",
1128         "    else{",
1129         "        return mul(C(n % p, m % p, p), lucas(n / p, m / p, p));",
1130         "    }",
1131     "}",
1132     ""
1133 ],
1134 "description": "Lucas求大组合数"
1135 },

```



```

1136 "逆元阶乘+组合数": {
1137     "prefix": "inv_fac",
1138     "body": [
1139         "const LL N = 1e6 + 10;",
1140         "LL fac[N], finv[N];",
1141         "",
1142         "void init(){",
1143         "// 需保证 mod > N, 不然不能这么做",
1144         "    fac[0] = 1;",
1145         "    for(int i = 1; i <= N; ++ i){",
1146         "        fac[i] = mul(fac[i - 1], i); ",
1147         "    }",
1148         "    finv[N] = qm(fac[N], mod - 2, mod);",
1149         "    for(int i = N - 1; i >= 1; -- i){",
1150         "        finv[i] = mul(finv[i + 1], i + 1);",
1151         "    }",
1152         "}",
1153         "LL C(LL n, LL m, LL p){",
1154         "    return m > n ? 0 : mul(fac[n], mul(finv[m], finv[n - m]));",
1155         "}",
1156         ""
1157     ],
1158     "description": "逆元阶乘+组合数"
1159 },
1160 "逆元线性打表": {
1161     "prefix": "inv_xian",
1162     "body": [
1163         "void init()",
1164         "{",
1165         "    inv[1] = 1;",
1166         "    for (int i = 2; i <= N; i ++)",
1167         "        inv[i] = mul(dec(mod, mod / i), inv[mod % i]); ",
1168         "}",
1169         ""
1170     ],
1171     "description": "逆元线性打表"
1172 },
1173 "KM": {
1174     "prefix": "KM",
1175     "body": [
1176         "namespace KM{",
1177         "    const int N = 1e4 + 10;",
1178         "    int n, ex_L[N], ex_R[N], match[N], slack[N], w[N][N];",
1179         "    bool vis_L[N], vis_R[N];",
1180         "    const int INF = 0x3f3f3f3f;",
1181         "    ",
1182         "    bool dfs(int u){",
1183         "        vis_L[u] = true;",
1184         "        for(int v = 1; v <= n; ++ v){",
1185         "            if(vis_R[v]) continue;",
1186         "            int gap = ex_L[u] + ex_R[v] - w[u][v];",
1187         "            if(!gap){",
1188         "                vis_R[v] = true;",
1189         "                if(!match[v] || dfs(match[v])){",
1190         "                    match[v] = u;",
1191         "                    return true;",
1192         "                }",
1193         "            }",
1194         "            else{",
1195         "                slack[v] = min(slack[v], gap);",
1196         "            }",
1197         "        }",
1198         "        return false;",
1199         "    }",
1200         "",
1201         "    int km(){",
1202         "        memset(match, 0, sizeof match);",
1203         "        memset(ex_R, 0, sizeof ex_R);",
1204         "        for(int i = 1; i <= n; ++ i){",
1205         "            ex_L[i] = w[i][1];",
1206         "            for(int j = 1; j <= n; ++ j){",

```

```

1207         ex_L[i] = max(ex_L[i], w[i][j]);",
1208     "        }",
1209     "    }",
1210     "    for(int i = 1; i <= n; ++ i){",
1211     "        memset(slack, 0x3f, sizeof slack);",
1212     "        while(true){",
1213     "            memset(vis_L, false, sizeof vis_L);",
1214     "            memset(vis_R, false, sizeof vis_R);",
1215     "            if(dfs(i)) break;",
1216     "            int d = INF;",
1217     "            for(int j = 1; j <= n; ++ j) ",
1218     "                if (!vis_R[j]) d = min(d, slack[j]);",
1219     "            for(int j = 1; j <= n; ++ j){",
1220     "                if(vis_L[j]) ex_L[j] -= d;",
1221     "                if(vis_R[j]) ex_R[j] += d;",
1222     "                else slack[j] -= d;",
1223     "            }",
1224     "        }",
1225     "    }",
1226     "    int ans = 0;",
1227     "    for(int i = 1; i <= n; ++ i){",
1228     "        if(match[i]) ans += w[match[i]][i];",
1229     "    }",
1230     "    return ans;",
1231     " }",
1232 "}",
1233 "using namespace KM;"
1234 ],
1235 "description": "KM(不用重新定义n)"
1236 },
1237 "字典树": {
1238     "prefix": "Trie",
1239     "body": [
1240         "namespace Trie{",
1241         "    const int N = 1e5 + 10;",
1242         "    int son[N][26], cnt[N], idx;",
1243         "    char s[N];",
1244         "    void init(){",
1245         "        memset(son, 0, sizeof son);",
1246         "        memset(cnt, 0, sizeof cnt);",
1247         "        idx = 0;",
1248         "    }",
1249         "    void insert(char *s){",
1250         "        int p = 0;",
1251         "        for(int i = 0; s[i]; ++ i){",
1252         "            int u = s[i] - 'a';",
1253         "            if(!son[p][u]) son[p][u] = ++ idx;",
1254         "            p = son[p][u];",
1255         "        }",
1256         "        cnt[p] ++;",
1257         "    }",
1258         "    int query(char *s){",
1259         "        int p = 0;",
1260         "        for(int i = 0; s[i]; ++ i){",
1261         "            int u = s[i] - 'a';",
1262         "            if(!son[p][u]) return 0;",
1263         "            p = son[p][u];",
1264         "        }",
1265         "        return cnt[p];",
1266         "    }",
1267     "}",
1268     "using namespace Trie;"
1269 ],
1270 "description": "字典树"
1271 },
1272 "01字典树": {
1273     "prefix": "Trie01",
1274     "body": [
1275         "namespace Trie01{",
1276         "    const int N = 1e5 + 10;",
1277         "    int son[N * 32][2], val[N * 32], idx;",

```

```

1278     "    void init(){",
1279     "        idx = 0;",
1280     "        son[0][0] = son[0][1] = 0;",
1281     "    }",
1282     "    void insert(int x){",
1283     "        int p = 0;",
1284     "        for(int i = 31; i >= 0; -- i){",
1285     "            int u = (x >> i) & 1;",
1286     "            if(!son[p][u]){",
1287     "                son[p][u] = ++ idx;",
1288     "                p = son[p][u];",
1289     "                son[p][0] = son[p][1] = 0;",
1290     "            }",
1291     "            else p = son[p][u];",
1292     "        }",
1293     "        val[p] = x;",
1294     "    }",
1295     "    int query(int x){",
1296     "        int p = 0;",
1297     "        for(int i = 31; i >= 0; -- i){",
1298     "            int u = (x >> i) & 1;",
1299     "            if(son[p][u ^ 1]) p = son[p][u ^ 1];",
1300     "            else p = son[p][u];",
1301     "        }",
1302     "        return val[p];",
1303     "    }",
1304     "}",
1305     "using namespace Trie01;"
1306 ],
1307     "description": "01字典树"
1308 },
1309 "树链剖分": {
1310     "prefix": "TreeChain",
1311     "body": [
1312         "namespace TreeChain{",
1313         "    int w[N];",
1314         "    int pre[N], sizx[N], son[N], deep[N];",
1315         "    int dfn[N], top[N], a[N];",
1316         "    int cnx; // dfs2 pool",
1317         "",
1318         "    void dfs1(int u, int fa)",
1319         "    {",
1320         "        pre[u] = fa;",
1321         "        deep[u] = deep[fa] + 1;",
1322         "        sizx[u] = 1;",
1323         "        int maxson = -1;",
1324         "        for (int i = h[u]; ~i; i = e[i].ne)",
1325         "        {",
1326         "            int v = e[i].v;",
1327         "            if (v != fa)",
1328         "            {",
1329         "                dfs1(v, u);",
1330         "                sizx[u] += sizx[v];",
1331         "                if (maxson < sizx[v])",
1332         "                {",
1333         "                    maxson = sizx[v];",
1334         "                    son[u] = v;",
1335         "                }",
1336         "            }",
1337         "        }",
1338         "    }",
1339     "",
1340     "    void dfs2(int u, int t)",
1341     "    {",
1342     "        top[u] = t;",
1343     "        dfn[u] = ++cnx;",
1344     "        a[cnx] = w[u];",
1345     "        if (!son[u])",
1346     "            return;",
1347     "        dfs2(son[u], t);",
1348     "        for (int i = h[u]; ~i; i = e[i].ne)",

```

```

1349         {"",
1350         "         int v = e[i].v;",
1351         "         if (v != pre[u] && v != son[u])",
1352         "         {",
1353         "             dfs2(v, v);",
1354         "         }",
1355         "     }",
1356     },
1357     "",
1358     "void mtre(int x, int y, int z)",
1359     "{",
1360     "    while (top[x] != top[y])",
1361     "    {",
1362     "        if (deep[top[x]] < deep[top[y]])",
1363     "        {",
1364     "            swap(x, y);",
1365     "        }",
1366     "        modify(1, dfn[top[x]], dfn[x], z);",
1367     "        x = pre[top[x]]; ",
1368     "    }",
1369     "    if (deep[x] > deep[y])",
1370     "    {",
1371     "        swap(x, y);",
1372     "    }",
1373     "    modify(1, dfn[x], dfn[y], z);",
1374     "    }",
1375     "}",
1376     "using namespace TreeChain;"
1377 ],
1378 "description": "树链剖分"
1379 }
1380 }
```