

# Standard Code Library

## (for Kunming Regional)

### Contents

#### 1 数学

1.1 Berlekamp-Massey最小递推式 . . . . .	1
1.1.1 优化矩阵快速幂DP . . . . .	1
1.1.2 求矩阵最小多项式 . . . . .	1
1.1.3 求稀疏矩阵的行列式 . . . . .	1
1.1.4 求稀疏矩阵的秩 . . . . .	1
1.1.5 解稀疏方程组 . . . . .	1

#### 2 数论

2.1 Miller-Rabin . . . . .	2
2.2 二次剩余 . . . . .	2

#### 3 图论

3.1 最小直径生成树 . . . . .	3
3.2 欧拉回路 . . . . .	4
3.3 预流推进费用流(可处理负环) $O(nm \log C)$ . . . . .	4
3.4 网络流原理 . . . . .	6
3.4.1 最大流 . . . . .	6
3.4.2 最小割 . . . . .	6
3.4.3 上下界网络流 . . . . .	6
3.4.4 常见建图方法 . . . . .	7
3.4.5 例题 . . . . .	7

3.5 Stoer-Wagner全局最小割 . . . . .	7
---------------------------------	---

#### 4 数据结构

4.1 历史和 . . . . .	8
4.2 二叉堆 . . . . .	9

#### 5 字符串

5.1 KMP . . . . .	9
5.1.1 ex-KMP . . . . .	9
5.2 区间本质不同子串计数 . . . . .	10

#### 6 动态规划

6.1 例题 . . . . .	12
6.1.1 103388A Assigning Prizes 容斥 . . . . .	12

#### 7 计算几何

7.1 最近点对 . . . . .	13
--------------------	----

#### 8 杂项

8.1 STL . . . . .	13
8.1.1 vector . . . . .	13
8.1.2 list . . . . .	13
8.1.3 unordered_set/map . . . . .	14
8.2 德州扑克 . . . . .	14

# 1 数学

## 1.1 Berlekamp-Massey最小递推式

如果要求出一个次数为 $k$ 的递推式, 则输入的数列需要至少有 $2k$ 项. 返回的内容满足 $\sum_{j=0}^{m-1} a_{i-j}c_j = 0$ , 并且 $c_0 = 1$ . 称为最小递推式. 如果不加最后的处理的话, 代码返回的结果会变成 $a_i = \sum_{j=0}^{m-1} c_{j-1}a_{i-j}$ , 有时候这样会方便接着跑递推, 需要的话就删掉最后的处理.

(实际上Berlekamp-Massey是对每个前缀都求出了递推式, 但一般没啥用.)

```
1 vector<int> berlekamp_massey(const vector<int> &a) {
2     vector<int> v, last; // v is the answer, 0-based
3     int k = -1, delta = 0;
4
5     for (int i = 0; i < (int)a.size(); i++) {
6
7         int tmp = 0;
8         for (int j = 0; j < (int)v.size(); j++)
9             tmp = (tmp + (long long)a[i - j - 1] *
10                  ↪ v[j]) % p;
11
12         if (a[i] == tmp)
13             continue;
14
15         if (k < 0) {
16             k = i;
17             delta = (a[i] - tmp + p) % p;
18             v = vector<int>(i + 1);
19
20             continue;
21         }
22
23         vector<int> u = v;
24         int val = (long long)(a[i] - tmp + p) *
25             ↪ qpow(delta, p - 2) % p;
26
27         if (v.size() < last.size() + i - k)
28             v.resize(last.size() + i - k);
29
30         (v[i - k - 1] += val) %= p;
31
32         for (int j = 0; j < (int)last.size(); j++) {
33             v[i - k + j] = (v[i - k + j] - (long
34             ↪ long)val * last[j]) % p;
35             if (v[i - k + j] < 0)
36                 v[i - k + j] += p;
37         }
38
39         if ((int)u.size() - i < (int)last.size() - k)
40             ↪ {
41                 last = u;
42                 k = i;
43                 delta = a[i] - tmp;
44                 if (delta < 0)
45                     delta += p;
46             }
47
48     }
49
50     for (auto &x : v) // 一般是需要最小递推式的, 所以处
51         ↪ 理一下
52         x = (p - x) % p;
53     v.insert(v.begin(), 1);
54
55     return v; //  $\forall i, \sum_{j=0}^m a_{i-j}v_j = 0$ 
56 }
```

如果要求向量序列的递推式, 就把每位乘一个随机权值(或者说是乘一个随机行向量 $v^T$ )变成求数列递推式即可.

如果是矩阵序列的话就随机一个行向量 $u^T$ 和列向量 $v$ , 然后把矩阵变成 $u^T A v$ 的数列就行了.

### 1.1.1 优化矩阵快速幂DP

如果 $f_i$ 是一个向量, 并且转移是一个矩阵, 那显然 $\{f_i\}$ 是一个线性递推序列.

假设 $f_i$ 有 $n$ 维, 先暴力求出 $f_{0 \sim 2n-1}$ , 然后跑Berlekamp-Massey, 最后调用前面的快速齐次线性递推(??页)即可. (快速齐次线性递推的结果是一个序列, 某个给定初值的结果就是点乘, 所以只需要跑一次.)

如果要求 $f_m$ , 并且矩阵有 $k$ 个非零项的话, 复杂度就是 $O(nk + n \log m \log n)$ . (因为暴力求前 $2n - 1$ 个 $f_i$ 需要 $O(nk)$ 时间.)

### 1.1.2 求矩阵最小多项式

矩阵 $A$ 的最小多项式是次数最小的并且 $f(A) = 0$ 的多项式 $f$ .

实际上最小多项式就是 $\{A^i\}$ 的最小递推式, 所以直接用Berlekamp-Massey就好了, 并且显然它的次数不超过 $n$ .

瓶颈在于求出 $A^i$ , 实际上我们只要处理 $A^i v$ 就行了, 每次对向量做递推.

假设 $A$ 有 $k$ 个非零项, 则复杂度为 $O(kn + n^2)$ .

### 1.1.3 求稀疏矩阵的行列式

如果能求出特征多项式, 则常数项乘上 $(-1)^n$ 就是行列式, 但是最小多项式不一定是特征多项式.

把 $A$ 乘上一个随机对角阵 $B$ (实际上就是每行分别乘一个随机数), 则 $AB$ 的最小多项式有很大概率就是特征多项式, 最后再除掉 $\det B$ 就行了.

设 $A$ 有 $k$ 个非零项, 则复杂度为 $O(kn + n^2)$ .

### 1.1.4 求稀疏矩阵的秩

设 $A$ 是一个 $n \times m$ 的矩阵, 首先随机一个 $n \times n$ 的对角阵 $P$ 和一个 $m \times m$ 的对角阵 $Q$ , 然后计算 $QAP A^T Q$ 的最小多项式即可.

实际上不用计算这个矩阵, 因为求最小多项式时要用它乘一个向量, 我们依次把这几个矩阵乘到向量里就行了. 答案就是最小多项式除掉所有 $x$ 因子后剩下的次数.

设 $A$ 有 $k$ 个非零项, 复杂度为 $O(kn + n^2)$ .

### 1.1.5 解稀疏方程组

**问题**  $Ax = b$ , 其中 $A$ 是一个 $n \times n$ 的满秩稀疏矩阵,  $b$ 和 $x$ 是 $1 \times n$ 的列向量,  $A, b$ 已知, 需要解出 $x$ .

**做法** 显然 $x = A^{-1}b$ . 如果我们能求出 $\{A^i b\} (i \geq 0)$ 的最小递推式 $\{r_{0 \sim m-1}\} (m \leq n)$ , 那么就有结论

$$A^{-1}b = -\frac{1}{r_{m-1}} \sum_{i=0}^{m-2} A^i b r_{m-2-i}$$

因为 $A$ 是稀疏矩阵, 直接按定义递推出 $b \sim A^{2n-1}b$ 即可. 设 $A$ 中有 $k$ 个非零项, 则复杂度为 $O(kn + n^2)$ .

```
1 vector<int> solve_sparse_equations(const
2     ↪ vector<tuple<int, int, int> > &A, const
3     ↪ vector<int> &b) {
4     int n = (int)b.size(); // 0-based
5
6     vector<vector<int> > f({b});
7
8     for (int i = 1; i < 2 * n; i++) {
9         vector<int> v(n);
10        auto &u = f.back();
```

```

9
10     for (auto [x, y, z] : A) // [x, y, value]
11         v[x] = (v[x] + (long long)u[y] * z) % p;
12
13     f.push_back(v);
14 }
15
16 vector<int> w(n);
17 mt19937 gen;
18 for (auto &x : w)
19     x = uniform_int_distribution<int>(1, p - 1)
20         ↳ (gen);
21
22 vector<int> a(2 * n);
23 for (int i = 0; i < 2 * n; i++)
24     for (int j = 0; j < n; j++)
25         a[i] = (a[i] + (long long)f[i][j] * w[j])
26             ↳ % p;
27
28 auto c = berlekamp_massey(a);
29 int m = (int)c.size();
30
31 vector<int> ans(n);
32
33 for (int i = 0; i < m - 1; i++)
34     for (int j = 0; j < n; j++)
35         ans[j] = (ans[j] + (long long)c[m - 2 - i]
36             ↳ * f[i][j]) % p;
37
38 int inv = qpow(p - c[m - 1], p - 2);
39
40 for (int i = 0; i < n; i++)
41     ans[i] = (long long)ans[i] * inv % p;
42
43 return ans;
44 }

```

## 2 数论

### 2.1 Miller-Rabin

```

1 // 复杂度可以认为是常数
2
3 // 用一个数检测
4 // 需要调用long long快速幂和O(1)快速乘
5 bool check(long long n, long long b) { // b: base
6     long long a = n - 1;
7     int k = 0;
8
9     while (a % 2 == 0) {
10         a /= 2;
11         k++;
12     }
13
14     long long t = qpow(b, a, n); // 这里的快速幂函数需
15         ↳ 要写O(1)快速乘
16     if (t == 1 || t == n - 1)
17         return true;
18
19     while (k--) {
20         t = mul(t, t, n); // mul是O(1)快速乘函数
21         if (t == n - 1)
22             return true;
23     }
24
25     return false;
26 }

```

```

26
27 // 封装好的函数体
28 // 需要调用check
29 bool Miller_Rabin(long long n) {
30     if (n == 1)
31         return false;
32     if (n == 2)
33         return true;
34     if (n % 2 == 0)
35         return false;
36
37     // int范围内只需要检查 {2, 7, 61}
38     // long long范围内只需要检查 {2, 325, 9375, 28178,
39         ↳ 450775, 9780504, 1795265022}
40
41     for (int i : {2, 325, 9375, 28178, 450775,
42         ↳ 9780504, 1795265022}) {
43         if (i ≥ n)
44             break;
45         if (!check(n, i))
46             return false;
47     }
48
49     return true;
50 }

```

### 2.2 二次剩余

```

1 int p, w;
2
3 struct pi {
4     int a, b; // a + b * sqrt(w)
5
6     pi(int a = 0, int b = 0) : a(a), b(b) {}
7
8     friend pi operator * (const pi &u, const pi &v) {
9         return pi(((long long)u.a * v.a + (long
10             ↳ long)u.b * v.b % p * w) % p,
11             ↳ ((long long)u.a * v.b + (long long)u.b *
12             ↳ v.a) % p);
13     }
14 };
15
16 pi qpow(pi a, int b) {
17     pi ans(1, 0);
18
19     while (b) {
20         if (b & 1)
21             ans = ans * a;
22
23         b >>= 1;
24         a = a * a;
25     }
26
27     return ans;
28 }
29
30 int qpow(int a, int b) {
31     int ans = 1;
32
33     while (b) {
34         if (b & 1)
35             ans = (long long)ans * a % p;
36
37         b >>= 1;
38         a = (long long)a * a % p;
39     }
40 }

```

```

39     return ans;
40 }
41
42 int my_legendre(int a) { // std有同名函数, 最好换个名
    ↪ 字, 不然传了两个数都查不出来
43     return qpow(a, (p - 1) / 2);
44 }
45
46 int quadratic_residual(int b, int mod) {
47     p = mod;
48
49     if (p == 2)
50         return 1;
51
52     if (my_legendre(b) == p - 1)
53         return -1; // 无解
54
55     int a;
56     do {
57         a = rand() % p;
58         w = ((long long)a * a - b) % p;
59         if (w < 0)
60             w += p;
61     } while (my_legendre(w) != p - 1);
62
63     return qpow(pi(a, 1), (p + 1) / 2).a;
64 }

```

## 3 图论

### 3.1 最小直径生成树

首先要找到图的绝对中心(可能在点上, 也可能在某条边上), 然后以绝对中心为起点建最短路树就是最小直径生成树.

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  constexpr int maxn = 505;
6  constexpr long long inf = 0x3f3f3f3f3f3f3f3fll;
7
8  int g[maxn][maxn], id[maxn][maxn], pr[maxn]; // g是邻接
    ↪ 矩阵
9  long long f[maxn][maxn], d[maxn];
10 bool vis[maxn];
11
12 vector<pair<int, int>>
    ↪ minimum_diameter_spanning_tree(int n) { // 1-based
13     for (int i = 1; i ≤ n; i++)
14         for (int j = 1; j ≤ n; j++)
15             g[i][j] *= 2; // 输入的边权都要乘2
16
17     memset(f, 63, sizeof(f));
18
19     for (int i = 1; i ≤ n; i++)
20         f[i][i] = 0;
21
22     for (int i = 1; i ≤ n; i++)
23         for (int j = 1; j ≤ n; j++)
24             if (g[i][j])
25                 f[i][j] = g[i][j];
26
27     for (int k = 1; k ≤ n; k++)
28         for (int i = 1; i ≤ n; i++)
29             for (int j = 1; j ≤ n; j++)
30                 f[i][j] = min(f[i][j], f[i][k] + f[k]
    ↪ [j]);

```

```

31
32     for (int i = 1; i ≤ n; i++) {
33         for (int j = 1; j ≤ n; j++)
34             id[i][j] = j; // 距离i第j近的点
35
36         sort(id[i] + 1, id[i] + n + 1, [&i] (int x,
    ↪ int y) {
37             return f[i][x] < f[i][y];
38         });
39     }
40
41     int o = 0;
42     long long ansv = inf; // vertex
43
44     for (int i = 1; i ≤ n; i++)
45         if (f[i][id[i][n]] * 2 < ansv) {
46             ansv = f[i][id[i][n]] * 2;
47             o = i;
48         }
49
50     int u = 0, v = 0;
51     long long disu = -inf, disv = -inf, anse = inf;
52
53     for (int x = 1; x ≤ n; x++)
54         for (int y = 1; y ≤ n; y++)
55             if (g[x][y]) { // 如果g[x][y] = 0说明没有边
56                 int w = g[x][y];
57
58                 for (int i = n - 1, j = n; i-->
59                     if (f[y][id[x][i]] > f[y][id[x]
    ↪ [j]]) {
60                         long long tmp = f[x][id[x][i]]
    ↪ + f[y][id[x][j]] + w;
61                         if (tmp < anse) {
62                             anse = tmp;
63                             u = x;
64                             v = y;
65
66                             disu = tmp / 2 - f[x]
    ↪ [id[x][i]];
67                             disv = w - disu;
68                         }
69                         j = i;
70                     }
71             }
72
73     printf("%lld\n", min(ansv, anse) / 2); // 直径
74
75     memset(d, 63, sizeof(d));
76
77     if (ansv ≤ anse)
78         d[o] = 0;
79     else {
80         d[u] = disu;
81         d[v] = disv;
82     }
83
84     for (int k = 1; k ≤ n; k++) { // Dijkstra
85         int x = 0;
86         for (int i = 1; i ≤ n; i++)
87             if (!vis[i] && d[i] < d[x])
88                 x = i;
89
90         vis[x] = true;
91         for (int y = 1; y ≤ n; y++)
92             if (g[x][y] && !vis[y]) {
93                 if (d[y] > d[x] + g[x][y]) {
94                     d[y] = d[x] + g[x][y];
95                     pr[y] = x;

```

```

96     }
97     else if (d[y] == d[x] + g[x][y] &&
98             ↪ d[pr[y]] < d[x])
99         pr[y] = x;
100     }
101 }
102 vector<pair<int, int>> vec;
103 for (int i = 1; i ≤ n; i++)
104     if (pr[i])
105         vec.emplace_back(i, pr[i]);
106
107 if (ansv > anse)
108     vec.emplace_back(u, v);
109
110 return vec;
111 }
112
113 int main() {
114
115     int n, m;
116     scanf("%d%d", &n, &m);
117
118     while (m--) {
119         int x, y, z;
120         scanf("%d%d%d", &x, &y, &z);
121
122         g[x][y] = g[y][x] = z; // 无向图
123     }
124
125     auto vec = minimum_diameter_spanning_tree(n);
126     for (auto [x, y] : vec)
127         printf("%d %d\n", x, y);
128
129     return 0;
130 }

```

### 3.2 欧拉回路

$C[x]$ 是记录每条边对应的编号的。

另外为了保证复杂度需要加当前弧优化。

```

1 vector<int> G[maxn], C[maxn], v[maxn];
2 int cur[maxn];
3 bool vis[maxn * 2];
4
5 vector<pair<int, int>> vec;
6
7 int d[maxn];
8
9 void dfs(int x) {
10     bool bad = false;
11
12     while (!bad) {
13         bad = true;
14
15         for (int &i = cur[x]; i < (int)G[x].size(); i++)
16             ↪ ++i
17             if (!vis[C[x][i]]) {
18                 vis[C[x][i]] = true;
19                 vec.emplace_back(x, i);
20                 x = G[x][i];
21                 bad = false;
22             }
23         break;
24     }
25 }

```

### 3.3 预流推进费用流(可处理负环) $O(nm \log C)$

不是很懂什么原理, 待研究.

```

1 // Push-Relabel implementation of the cost-scaling
2   ↪ algorithm
3 // Runs in  $O(\text{max\_flow} * \log(V * \text{max\_edge\_cost})) =$ 
4   ↪  $O(V^3 * \log(V * C))$ 
5 // Really fast in practice, 3e4 edges are fine.
6 // Operates on integers, costs are multiplied by N!!
7
8 #include <bits/stdc++.h>
9 using namespace std;
10
11 // source: unknown
12 template<typename flow_t = int, typename cost_t = int>
13 struct mcSFlow {
14     struct Edge {
15         cost_t c;
16         flow_t f;
17         int to, rev;
18         Edge(int _to, cost_t _c, flow_t _f, int _rev):
19             ↪ c(_c), f(_f), to(_to), rev(_rev) {}
20     };
21
22     static constexpr cost_t INFCOST =
23         ↪ numeric_limits<cost_t>::max() / 2;
24
25     cost_t eps;
26     int N, S, T;
27     vector<vector<Edge>> G;
28     vector<unsigned int> isq, cur;
29     vector<flow_t> ex;
30     vector<cost_t> h;
31
32     mcSFlow(int _N, int _S, int _T): eps(0), N(_N),
33         ↪ S(_S), T(_T), G(_N) {}
34
35     void add_edge(int a, int b, flow_t cap, cost_t
36         ↪ cost) {
37         assert(cap ≥ 0);
38         assert(a ≥ 0 && a < N && b ≥ 0 && b < N);
39
40         if (a == b) {
41             assert(cost ≥ 0);
42             return;
43         }
44
45         cost *= N;
46         eps = max(eps, abs(cost));
47         G[a].emplace_back(b, cost, cap, G[b].size());
48         G[b].emplace_back(a, -cost, 0, G[a].size() -
49             ↪ 1);
50     }
51
52     void add_flow(Edge &e, flow_t f) {
53         Edge &back = G[e.to][e.rev];
54
55         if (!ex[e.to] && f)
56             hs[h[e.to]].push_back(e.to);
57
58         e.f -= f;
59         ex[e.to] += f;
60         back.f += f;
61         ex[back.to] -= f;
62     }
63
64     vector<vector<int>> hs;
65     vector<int> co;

```

```

59 flow_t max_flow() {
60     ex.assign(N, 0);
61     h.assign(N, 0);
62     hs.resize(2 * N);
63     co.assign(2 * N, 0);
64     cur.assign(N, 0);
65     h[S] = N;
66     ex[T] = 1;
67     co[0] = N - 1;
68
69     for (auto &e : G[S])
70         add_flow(e, e.f);
71
72     if (hs[0].size())
73         for (int hi = 0; hi ≥ 0; ) {
74             int u = hs[hi].back();
75             hs[hi].pop_back();
76
77             while (ex[u] > 0) { // discharge u
78                 if (cur[u] == G[u].size()) {
79                     h[u] = 1e9;
80
81                     for (unsigned int i = 0; i <
82                         ↪ G[u].size(); ++i) {
83                         auto &e = G[u][i];
84
85                         if (e.f && h[u] > h[e.to]
86                             ↪ + 1) {
87                             h[u] = h[e.to] + 1,
88                             ↪ cur[u] = i;
89                         }
90
91                         if (++co[h[u]], !--co[hi] &&
92                             ↪ hi < N)
93                             for (int i = 0; i < N; +
94                                 ↪ +i)
95                                 if (hi < h[i] && h[i]
96                                     ↪ < N) {
97                                     --co[h[i]];
98                                     h[i] = N + 1;
99                                 }
100
101                             hi = h[u];
102                         }
103                     else if (G[u][cur[u]].f && h[u] ==
104                         ↪ h[G[u][cur[u]].to] + 1)
105                         add_flow(G[u][cur[u]],
106                             ↪ min(ex[u], G[u]
107                             ↪ [cur[u]].f));
108                     else
109                         ++cur[u];
110                 }
111
112                 while (hi ≥ 0 && hs[hi].empty())
113                     --hi;
114             }
115
116             return -ex[S];
117         }
118
119 void push(Edge &e, flow_t amt) {
120     if (e.f < amt)
121         amt = e.f;
122
123     e.f -= amt;
124     ex[e.to] += amt;
125     G[e.to][e.rev].f += amt;
126
127     ex[G[e.to][e.rev].to] -= amt;
128 }
129
130 void relabel(int vertex) {
131     cost_t newHeight = -INFCOST;
132
133     for (unsigned int i = 0; i < G[vertex].size();
134         ↪ ++i) {
135         Edge const &e = G[vertex][i];
136
137         if (e.f && newHeight < h[e.to] - e.c) {
138             newHeight = h[e.to] - e.c;
139             cur[vertex] = i;
140         }
141     }
142
143     h[vertex] = newHeight - eps;
144 }
145
146 static constexpr int scale = 2;
147
148 pair<flow_t, cost_t> minCostMaxFlow() {
149     cost_t retCost = 0;
150
151     for (int i = 0; i < N; ++i)
152         for (Edge &e : G[i])
153             retCost += e.c * (e.f);
154
155     //find max-flow
156     flow_t retFlow = max_flow();
157     h.assign(N, 0);
158     ex.assign(N, 0);
159     isq.assign(N, 0);
160     cur.assign(N, 0);
161     queue<int> q;
162
163     for (; eps; eps >= scale) {
164         //refine
165         fill(cur.begin(), cur.end(), 0);
166
167         for (int i = 0; i < N; ++i)
168             for (auto &e : G[i])
169                 if (h[i] + e.c - h[e.to] < 0 &&
170                     ↪ e.f)
171                     push(e, e.f);
172
173         for (int i = 0; i < N; ++i) {
174             if (ex[i] > 0) {
175                 q.push(i);
176                 isq[i] = 1;
177             }
178         }
179
180         // make flow feasible
181         while (!q.empty()) {
182             int u = q.front();
183             q.pop();
184             isq[u] = 0;
185
186             while (ex[u] > 0) {
187                 if (cur[u] == G[u].size())
188                     relabel(u);
189
190                 for (unsigned int &i = cur[u],
191                     ↪ max_i = G[u].size(); i <
192                     ↪ max_i; ++i) {
193                     Edge &e = G[u][i];

```

```

183         if (h[u] + e.c - h[e.to] < 0)
184             ↪ {
185                 push(e, ex[u]);
186
187                 if (ex[e.to] > 0 &&
188                     ↪ isq[e.to] == 0) {
189                     q.push(e.to);
190                     isq[e.to] = 1;
191                 }
192             }
193         }
194     }
195 }
196
197 if (eps > 1 && eps >> scale == 0)
198     eps = 1 << scale;
199
200 }
201
202 for (int i = 0; i < N; ++i)
203     for (Edge &e : G[i])
204         retCost -= e.c * (e.f);
205
206 return make_pair(retFlow, retCost / 2 / N);
207 }
208
209 flow_t getFlow(Edge const &e) {
210     return G[e.to][e.rev].f;
211 }
212 };
213
214 int main() {
215
216     int n, m;
217     scanf("%d%d", &n, &m);
218
219     mcmf<long long, long long> mcmf(n, 0, n - 1);
220
221     while (m--) {
222         int x, y, z, w;
223         scanf("%d%d%d%d", &x, &y, &z, &w);
224
225         mcmf.add_edge(x - 1, y - 1, z, w);
226     }
227
228     auto [flow, cost] = mcmf.minCostMaxFlow();
229
230     printf("%lld %lld\n", flow, cost);
231
232     return 0;
233 }

```

## 3.4 网络流原理

### 3.4.1 最大流

#### • 判断一条边是否必定满流

在残量网络中跑一遍Tarjan，如果某条满流边的两端处于同一SCC中则说明它不一定满流。（因为可以找出包含反向边的环，增广之后就不满流了。）

### 3.4.2 最小割

首先牢记最小割的定义：选权值和尽量小的一些边，使得删除这些边之后 $s$ 无法到达 $t$ 。

#### • 最小割输出一种方案

在残量网络上从 $S$ 开始floodfill，源点可达的记为 $S$ 集，不可达的记为 $T$ ，如果一条边的起点在 $S$ 集而终点在 $T$ 集，就将其加入最小割中。

#### • 最小割的可行边与必须边

— 可行边：满流，且残量网络上不存在 $u$ 到 $v$ 的路径，也就是 $u$ 和 $v$ 不在同一SCC中。（实际上也就是最大流必定满流的边。）

— 必须边：满流，且残量网络上 $S$ 可达 $u$ ， $v$ 可达 $T$ 。

#### • 字典序最小的最小割

直接按字典序从小到大的顺序依次判断每条边能否在最小割中即可。

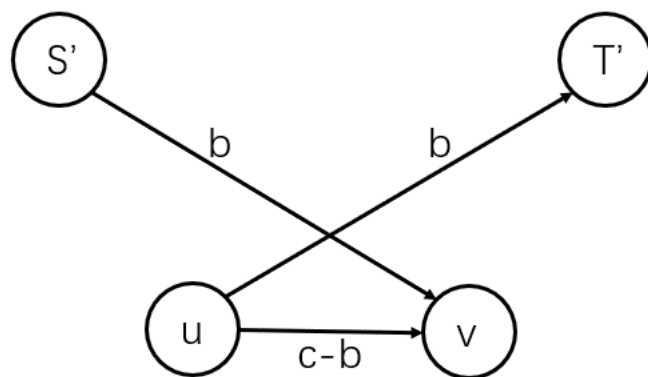
如果一条边是可行边，我们就需要把它删掉，同时进行退流， $u \rightarrow s$ 和 $t \rightarrow v$ 都退掉等同于这条边容量的流量。

退流用Dinic实现即可。

### 3.4.3 上下界网络流

#### 无源汇上下界可行流

新建源汇 $S'$ ， $T'$ ，然后如图所示转化每一条边。



在新图跑一遍最大流之后检查一遍辅助边，如果有辅助边没满流则无解，否则把每条边的流量加上 $b$ 就是一组可行方案。

#### 有源汇上下界最大流

如果不需要判断是否有解的话可以直接按照和上面一样的方法转化。因为附加边实际上算了两次流量，所以最终答案应该减掉所有下界之和。

（另外这里如果要压缩附加边的话，不能像无源汇的情况一样对每个点只开一个变量统计溢出的流量，正确的做法是进出流量各统计一下，每个点连两条附加边。）

如果需要判有解的话会出一点问题。这时候就需要转化成无源汇的情况，验证有解之后撤掉 $T'$ 到 $S'$ 的那条附加边再从 $S'$ 到 $T'$ 跑一遍最大流。

```

1 int ex[maxn], id[maxn];
2
3 int main() {
4
5     memset(last, -1, sizeof(last));
6
7     int n, m, src, sink;
8     scanf("%d%d%d%d", &n, &m, &src, &sink);
9     s = n + 1;
10    t = n + 2;
11
12    while (m--) {
13        int x, y, b, c;
14        scanf("%d%d%d%d", &x, &y, &b, &c);
15
16        addedge(x, y, c - b);
17
18        ex[y] += b;
19        ex[x] -= b;
20    }
21

```



```

22     for (int i = 1; i ≤ n; i++) {
23         id[i] = cnte;
24
25         if (ex[i] ≥ 0)
26             addedge(s, i, ex[i]);
27         else
28             addedge(i, t, -ex[i]);
29     }
30
31     addedge(sink, src, (~0u) >> 1);
32
33     Dinic();
34
35     if (any_of(id + 1, id + n + 1, [] (int i) {return
36         ↪ (bool)e[i].cap;}))
37         printf("please go home to sleep\n");
38     else {
39         int flow = e[cnte - 1].cap;
40         e[cnte - 1].cap = e[cnte - 2].cap = 0;
41         s = src;
42         t = sink;
43
44         printf("%d\n", flow + Dinic());
45     }
46
47     return 0;

```

#### 有源汇上下界最小流

按照上面的方法转换后先跑一遍最大流，然后撤掉超级源汇和附加边，反过来跑一次最大流退流，最大流减去退掉的流量就是最小流。

#### 3.4.4 常见建图方法

##### • 最大流/费用流

流量不是很多的时候可以理解成很多条路径，并且每条边可以经过的次数有限。

##### • 最小割

常用的模型是最大权闭合子图。当然它并不是万能的，因为限制条件可以带权值。

##### 1. 如果某些点全部在 $S$ 集或者 $T$ 集则获得一个正的收益

把这个条件建成一个点，向要求的点连 $\infty$ 边，然后 $s$ 向它连 $\infty$ 边。（如果是 $T$ 集就都反过来）

那么如果它在 $S$ 集就一定满足它要求的点都在 $S$ 集，反之如果是 $T$ 集亦然。

##### 2. 如果两个点不在同一集合中则需要付出代价

建双向边，那显然如果它们不在同一集合中就需要割掉中间的边，付出对应的代价。

##### 3. 二分图，如果相邻的两个点在同一集合则需要付出代价

染色后给一半的点反转源汇，就转换成上面的问题了。

#### 3.4.5 例题

##### • 费用流

##### 1. 序列上选和尽量大的数，但连续 $k$ 个数中最多选 $p$ 个。

费用流建图，先建一条 $n + 1$ 个点的无限容量的链表示不选，然后每个点往后面 $k$ 个位置连边，答案是流量为 $p$ 的最大费用流。因为条件等价于选 $p$ 次并且每次选的所有数间隔都至少是 $k$ 。

##### 2. 还要求连续 $k$ 个数中最少选 $q$ 个。

任选一个位置把图前后切开就会发现通过截面的流量总和恰为 $p$ 。注意到如果走了最开始的链就代表不选，因此要限制至少有 $q$ 的流量不走链，那么只需要把链的容量改成 $p - q$ 就行了。

### 3.5 Stoer-Wagner全局最小割

```

1  const int N = 601;
2  int fa[N], siz[N], edge[N][N];
3
4  int find(int x) {
5      return fa[x] == x ? x : fa[x] = find(fa[x]);
6  }
7
8  int dist[N], vis[N], bin[N];
9  int n, m;
10
11 int contract(int& s, int& t) { // Find s,t
12     memset(dist, 0, sizeof(dist));
13     memset(vis, false, sizeof(vis));
14
15     int i, j, k, mincut, maxc;
16
17     for (i = 1; i ≤ n; i++) {
18         k = -1;
19         maxc = -1;
20
21         for (j = 1; j ≤ n; j++)
22             if (!bin[j] && !vis[j] && dist[j] > maxc)
23                 ↪ {
24                     k = j;
25                     maxc = dist[j];
26                 }
27
28         if (k == -1)
29             return mincut;
30
31         s = t;
32         t = k;
33         mincut = maxc;
34         vis[k] = true;
35
36         for (j = 1; j ≤ n; j++)
37             if (!bin[j] && !vis[j]) dist[j] += edge[k]
38                 ↪ [j];
39     }
40
41     return mincut;
42 }
43
44 const int inf = 0x3f3f3f3f;
45
46 int Stoer_Wagner() {
47     int mincut, i, j, s, t, ans;
48     for (mincut = inf, i = 1; i < n; i++) {
49         ans = contract(s, t);
50         bin[t] = true;
51
52         if (mincut > ans)
53             mincut = ans;
54         if (mincut == 0)
55             return 0;
56
57         for (j = 1; j ≤ n; j++)
58             if (!bin[j])
59                 edge[s][j] = (edge[j][s] += edge[j]
60                     ↪ [t]);
61     }
62
63     return mincut;
64 }
65
66 int main() {
67     cin >> n >> m;

```



```

66     if (m < n - 1) {
67         cout << 0;
68         return 0;
69     }
70
71     for (int i = 1; i ≤ n; ++i)
72         fa[i] = i, siz[i] = 1;
73
74     for (int i = 1, u, v, w; i ≤ m; ++i) {
75         cin >> u >> v >> w;
76
77         int fu = find(u), fv = find(v);
78         if (fu != fv) {
79             if (siz[fu] > siz[fv]) swap(fu, fv);
80             fa[fu] = fv, siz[fv] += siz[fu];
81         }
82
83         edge[u][v] += w, edge[v][u] += w;
84     }
85
86     int fr = find(1);
87
88     if (siz[fr] != n) {
89         cout << 0;
90         return 0;
91     }
92
93     cout << Stoer_Wagner();
94
95     return 0;
96 }

```

## 4 数据结构

### 4.1 历史和

EC-Final2020 G, 原题是询问某个区间有多少子区间, 满足子区间中数的种类数为奇数.

离线之后转化成枚举右端点并用线段树维护左端点, 然后就是一个支持区间反转(0/1互换)和询问历史和的线段树.

“既然标记会复合 就说明在两个标记中间没有经过任何 pushup 操作

也就是说一个这两个标记对应着 相同的 0 的数量 以及 相同的 1 的数量

那么标记对于答案的影响只能是  $a * 0 + b * 1$

我们维护 a b 即可”

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  constexpr int maxn = (1 << 20) + 5;
6
7  int cnt[maxn][2], mul[maxn][2];
8  bool rev[maxn];
9  long long sum[maxn];
10
11 int now;
12
13 void build(int l, int r, int o) {
14     cnt[o][0] = r - l + 1;
15
16     if (l == r)
17         return;
18
19     int mid = (l + r) / 2;
20

```

```

21     build(l, mid, o * 2);
22     build(mid + 1, r, o * 2 + 1);
23 }
24
25 void apply(int o, bool flip, long long w0, long long
    ↪ w1) {
26     sum[o] += w0 * cnt[o][0] + w1 * cnt[o][1];
27
28     if (flip)
29         swap(cnt[o][0], cnt[o][1]);
30
31     if (rev[o])
32         swap(w0, w1);
33
34     mul[o][0] += w0;
35     mul[o][1] += w1;
36     rev[o] ^= flip;
37 }
38
39 void pushdown(int o) {
40     if (!mul[o][0] && !mul[o][1] && !rev[o])
41         return;
42
43     apply(o * 2, rev[o], mul[o][0], mul[o][1]);
44     apply(o * 2 + 1, rev[o], mul[o][0], mul[o][1]);
45
46     mul[o][0] = mul[o][1] = 0;
47     rev[o] = false;
48 }
49
50 void update(int o) {
51     cnt[o][0] = cnt[o * 2][0] + cnt[o * 2 + 1][0];
52     cnt[o][1] = cnt[o * 2][1] + cnt[o * 2 + 1][1];
53
54     sum[o] = sum[o * 2] + sum[o * 2 + 1];
55 }
56
57 int s, t;
58
59 void modify(int l, int r, int o) {
60     if (s ≤ l && t ≥ r) {
61         apply(o, true, 0, 0);
62         return;
63     }
64
65     int mid = (l + r) / 2;
66     pushdown(o);
67
68     if (s ≤ mid)
69         modify(l, mid, o * 2);
70     if (t > mid)
71         modify(mid + 1, r, o * 2 + 1);
72
73     update(o);
74 }
75
76 long long query(int l, int r, int o) {
77     if (s ≤ l && t ≥ r)
78         return sum[o];
79
80     int mid = (l + r) / 2;
81     pushdown(o);
82
83     long long ans = 0;
84     if (s ≤ mid)
85         ans += query(l, mid, o * 2);
86     if (t > mid)
87         ans += query(mid + 1, r, o * 2 + 1);
88
89     return ans;

```

```

90 }
91
92 vector<pair<int, int> > vec[maxn]; // pos, id
93
94 long long ans[maxn];
95 int a[maxn], last[maxn];
96
97 int main() {
98
99     int n;
100     scanf("%d", &n);
101
102     build(1, n, 1);
103
104     for (int i = 1; i ≤ n; i++)
105         scanf("%d", &a[i]);
106
107     int m;
108     scanf("%d", &m);
109
110     for (int i = 1; i ≤ m; i++) {
111         int l, r;
112         scanf("%d%d", &l, &r);
113
114         vec[r].emplace_back(l, i);
115     }
116
117     for (int i = 1; i ≤ n; i++) {
118         s = last[a[i]] + 1;
119         t = now = i;
120
121         modify(1, n, 1);
122         apply(1, false, 0, 1);
123
124         for (auto [l, k] : vec[i]) {
125             s = l;
126             ans[k] = query(1, n, 1);
127         }
128
129         last[a[i]] = i;
130     }
131
132     for (int i = 1; i ≤ m; i++)
133         printf("%lld\n", ans[i]);
134
135     return 0;
136 }

```

```

20     return a[1];
21 }
22
23 int pop() {
24     int res = a[1];
25     a[1] = a[size--];
26
27     for (int x = 1, son; ; x = son) {
28         if (x * 2 == size)
29             son = x * 2;
30         else if (x * 2 > size)
31             break;
32         else if (a[x * 2] < a[x * 2 + 1])
33             son = x * 2;
34         else
35             son = x * 2 + 1;
36
37         if (a[son] < a[x])
38             swap(a[x], a[son]);
39         else
40             break;
41     }
42
43     return res;
44 }
45 };

```

## 5 字符串

### 5.1 KMP

```

1 void kmp(const char *s, int *fail, int n) { // 1-based
2     int j = fail[0] = 0;
3
4     for (int i = 1; i ≤ n; i++) {
5         while (j && s[i] != s[j])
6             j = fail[j - 1];
7
8         fail[i] = (j += s[i] == s[j]);
9     }
10 }

```

### 4.2 二叉堆

```

1 struct my_binary_heap {
2     static constexpr int maxn = 100005;
3
4     int a[maxn], size;
5
6     my_binary_heap() : size(0) {}
7
8     void push(int val) {
9         a[++size] = val;
10
11         for (int x = size; x > 1; x /= 2) {
12             if (a[x] < a[x / 2])
13                 swap(a[x], a[x / 2]);
14             else
15                 break;
16         }
17     }
18
19     int &top() {

```

#### 5.1.1 ex-KMP

```

1 void exkmp(const char *s, int *a, int n) {
2     int l = 0, r = 0;
3     a[0] = n;
4
5     for (int i = 1; i ≤ n; i++) {
6         a[i] = i > r ? 0 : min(r - i + 1, a[i - l]);
7
8         while (i + a[i] < n && s[a[i]] == s[i + a[i]])
9             a[i]++;
10
11         if (i + a[i] - 1 > r) {
12             l = i;
13             r = i + a[i] - 1;
14         }
15     }
16 }

```

## 5.2 区间本质不同子串计数(后缀自动机+LCT+线段树)

问题: 给定一个字符串 $s$ , 多次询问 $[l, r]$ 区间的本质不同的子串个数, 可能强制在线.

做法: 考虑建出后缀自动机, 然后枚举右端点, 用线段树维护每个左端点的答案.

显然只有 $\text{right}$ 集合在 $[l, r]$ 中的串才有可能有贡献, 所以我们可以只考虑每个串最大的 $\text{right}$ .

每次右端点+1时找到它对应的结点 $u$ , 则 $u$ 到根节点路径上的每个点, 它的 $\text{right}$ 集合都会被 $r$ 更新.

对于某个特定的左端点 $l$ , 我们需要保证本质不同的子串左端点不能越过它; 因此对于一个结点 $p$ , 我们知道它对应的子串长度 $(\text{val}_{\text{par}_p}, \text{val}_p)$ 之后, 在 $p$ 的 $\text{right}$ 集合最大值减去对应长度, 这样对应的 $l$ 内全部+1即可; 这样询问时就只需要查询 $r$ 对应的线段树中 $[l, r]$ 的区间和. (当然旧的 $\text{right}$ 对应的区间也要减掉)

实际上可以发现更新时都是把路径分成若干个整段更新 $\text{right}$ 集合, 因此可以用LCT维护这个过程.

时间复杂度 $O(n \log^2 n)$ , 空间 $O(n)$ , 当然如果强制在线的话, 就把线段树改成主席树, 空间复杂度就和时间复杂度同阶了.

```

1 int tim; // tim实际上就是当前的右端点
2
3 node *access(node *x) {
4     node *y = null;
5
6     while (x != null) {
7         splay(x);
8
9         x → ch[1] = null;
10        x → refresh();
11
12        if (x → val) // val记录的是上次访问时间, 也就
            ↳ 是right集合最大值
13            update(x → val - val[x → r] + 1, x →
                ↳ val - val[par[x → l]], -1);
14
15        x → val = tim;
16        x → lazy = true;
17
18        update(x → val - val[x → r] + 1, x → val -
            ↳ val[par[x → l]], 1);
19
20        x → ch[1] = y;
21
22        (y = x) → refresh();
23
24        x = x → p;
25    }
26
27    return y;
28 }
29
30 // 以下是main函数中的用法
31 for (int i = 1; i ≤ n; i++) {
32     tim++;
33     access(null + id[i]);
34
35     if (i ≥ m) // 例题询问长度是固定的, 如果不固定的话
        ↳ 就按照右端点离线即可
36         ans[i - m + 1] = query(i - m + 1, i);
37 }

```

还有一份完整的代码, 因为写起来确实细节挺多的. 这份代码支持在尾部加一个字符或者询问区间有多少子串至少出现了两次, 并且强制在线.

```

1 #include <bits/stdc++.h>
2

```

```

3 using namespace std;
4
5 constexpr int maxn = 200005, maxm = maxn * 17 * 15;
6
7 int mx[maxm][2], lc[maxm], rc[maxm], seg_cnt;
8 int root[maxn];
9
10 int s, t, d;
11
12 void modify_seg(int l, int r, int &o) {
13     int u = o;
14     o = ++seg_cnt;
15
16     mx[o][0] = max(mx[u][0], t);
17     mx[o][1] = max(mx[u][1], d);
18
19     if (l == r)
20         return;
21
22     lc[o] = lc[u];
23     rc[o] = rc[u];
24
25     int mid = (l + r) / 2;
26     if (s ≤ mid)
27         modify_seg(l, mid, lc[o]);
28     else
29         modify_seg(mid + 1, r, rc[o]);
30 }
31
32 int query_seg(int l, int r, int o, int k) {
33     if (s ≤ l && t ≥ r)
34         return mx[o][k];
35
36     int mid = (l + r) / 2, ans = 0;
37
38     if (s ≤ mid)
39         ans = max(ans, query_seg(l, mid, lc[o], k));
40     if (t > mid)
41         ans = max(ans, query_seg(mid + 1, r, rc[o],
            ↳ k));
42
43     return ans;
44 }
45
46 int N;
47
48 void modify(int pos, int u, int v, int &rt) {
49     s = pos;
50     t = u;
51     d = v;
52
53     modify_seg(1, N, rt);
54 }
55
56 int query(int l, int r, int rt) {
57     s = l;
58     t = r;
59     int ans = query_seg(1, N, rt, 0);
60
61     s = 1;
62     t = l;
63     return max(ans, query_seg(1, N, rt, 1) - l);
64 }
65
66 struct node {
67     int size, l, r, id, tim;
68     node *ch[2], *p;
69     bool tag;
70
71     node() = default;

```

```

72
73 void apply(int v) {
74     tim = v;
75     tag = true;
76 }
77
78 void pushdown() {
79     if (tag) {
80         ch[0] → tim = ch[1] → tim = tim;
81         ch[0] → tag = ch[1] → tag = true;
82
83         tag = false;
84     }
85 }
86
87 void update() {
88     size = ch[0] → size + ch[1] → size + 1;
89     l = (ch[0] → l ? ch[0] → l : id);
90     r = (ch[1] → r ? ch[1] → r : id);
91 }
92 } null[maxn];
93
94 inline bool isroot(node *x) {
95     return x != x → p → ch[0] && x != x → p →
96         ↪ ch[1];
97 }
98
99 inline bool dir(node *x) {
100     return x == x → p → ch[1];
101 }
102
103 void init(node *x, int i) {
104     *x = node();
105     x → ch[0] = x → ch[1] = x → p = null;
106     x → size = 1;
107     x → id = x → l = x → r = i;
108 }
109
110 void rot(node *x, int d) {
111     node *y = x → ch[d ^ 1];
112
113     y → p = x → p;
114     if (!isroot(x))
115         x → p → ch[dir(x)] = y;
116
117     if ((x → ch[d ^ 1] = y → ch[d]) != null)
118         y → ch[d] → p = x;
119     (y → ch[d] = x) → p = y;
120
121     x → update();
122     y → update();
123 }
124
125 void splay(node *x) {
126     x → pushdown();
127
128     while (!isroot(x)) {
129         if (!isroot(x → p))
130             x → p → p → pushdown();
131         x → p → pushdown();
132         x → pushdown();
133
134         if (isroot(x → p)) {
135             rot(x → p, dir(x) ^ 1);
136             break;
137         }
138
139         if (dir(x) == dir(x → p))
140             rot(x → p → p, dir(x → p) ^ 1);

```

```

141         rot(x → p, dir(x) ^ 1);
142
143         rot(x → p, dir(x) ^ 1);
144     }
145 }
146
147 void splay(node *x, node *rt) {
148     x → pushdown();
149
150     while (x → p != rt) {
151         if (x → p → p != rt)
152             x → p → p → pushdown();
153         x → p → pushdown();
154         x → pushdown();
155
156         if (x → p → p == rt) {
157             rot(x → p, dir(x) ^ 1);
158             break;
159         }
160
161         if (dir(x) == dir(x → p))
162             rot(x → p → p, dir(x → p) ^ 1);
163         else
164             rot(x → p, dir(x) ^ 1);
165
166         rot(x → p, dir(x) ^ 1);
167     }
168 }
169
170 int val[maxn], par[maxn], go[maxn][26], sam_cnt,
171     ↪ sam_last;
172
173 node *access(node *x, int r) {
174     root[r] = root[r - 1];
175
176     node *y = null;
177
178     while (x != null) {
179         splay(x);
180
181         x → pushdown();
182
183         x → ch[1] = null;
184         x → update();
185
186         if (x → tim && val[x → r]) { // last time
187             ↪ visited
188             int right = x → tim, left = right - val[x
189                 ↪ → r] + 1;
190             modify(left, val[x → r], right + 1,
191                 ↪ root[r]);
192         }
193
194         x → apply(r);
195         x → pushdown();
196
197         x → ch[1] = y;
198         (y = x) → update();
199
200         x = x → p;
201     }
202
203     return y;
204 }
205
206 void new_leaf(node *x, node *par) {
207     x → p = par;
208 }

```

```

206 void new_node(node *x, node *y, node *par) {
207     splay(y);
208
209     if (isroot(y) && y → p == par) {
210         assert(y → ch[0] == null);
211
212         y → ch[0] = x;
213         x → p = y;
214         y → update();
215     }
216     else {
217         splay(par, y);
218
219         assert(y → ch[0] == par);
220         assert(par → ch[1] == null);
221         par → ch[1] = x;
222         x → p = par;
223
224         par → update();
225         y → update();
226     }
227
228     x → tim = y → tim;
229 }
230
231 void extend(int c) {
232     int p = sam_last, np = ++sam_cnt;
233     val[np] = val[p] + 1;
234
235     init(null + np, np);
236
237     while (p && !go[p][c]) {
238         go[p][c] = np;
239         p = par[p];
240     }
241
242     if (!p) {
243         par[np] = 1;
244         new_leaf(null + np, null + par[np]);
245     }
246     else {
247         int q = go[p][c];
248
249         if (val[q] == val[p] + 1) {
250             par[np] = q;
251             new_leaf(null + np, null + par[np]);
252         }
253         else {
254             int nq = ++sam_cnt;
255             val[nq] = val[p] + 1;
256             memcpy(go[nq], go[q], sizeof(go[q]));
257
258             init(null + nq, nq);
259
260             new_node(null + nq, null + q, null +
                ↪ par[q]);
261             new_leaf(null + np, null + nq);
262
263             par[nq] = par[q];
264             par[np] = par[q] = nq;
265
266             while (p && go[p][c] == q) {
267                 go[p][c] = nq;
268                 p = par[p];
269             }
270         }
271     }
272
273     sam_last = np;

```

```

274 }
275
276 char str[maxn];
277
278 int main() {
279
280     init(null, 0);
281
282     sam_last = sam_cnt = 1;
283     init(null + 1, 1);
284
285     int n, m;
286     scanf("%s%d", str + 1, &m);
287     n = strlen(str + 1);
288     N = n + m;
289
290     for (int i = 1; i ≤ n; i++) {
291         extend(str[i] - 'a');
292         access(null + sam_last, i);
293     }
294
295     int tmp = 0;
296
297     while (m--) {
298         int op;
299         scanf("%d", &op);
300
301         if (op == 1) {
302             scanf(" %c", &str[++n]);
303
304             str[n] = (str[n] - 'a' + tmp) % 26 + 'a';
305
306             extend(str[n] - 'a');
307             access(null + sam_last, n);
308         }
309         else {
310             int l, r;
311             scanf("%d%d", &l, &r);
312
313             l = (l - 1 + tmp) % n + 1;
314             r = (r - 1 + tmp) % n + 1;
315
316             printf("%d\n", tmp = query(l, r,
                ↪ root[r]));
317         }
318     }
319
320     return 0;
321 }

```

## 6 动态规划

### 6.1 例题

#### 6.1.1 103388A Assigning Prizes 容斥

**题意** 给定一个长为  $n$  的序列  $a_i$ , 要求构造非严格递减序列  $b_i$ , 满足  $a_i \leq b_i \leq R$ , 求方案数.  $n \leq 5 \times 10^3, R, a_i \leq 10^9$ .

**做法**  $a_i$  的范围太大了, 不能简单地记录上一位的值.

考虑使用容斥. 方便起见把  $a_i$  直接变成  $R - a_i + 1$ , 条件就变成了  $b_i \leq a_i$  且  $b_i \geq b_{i-1}$ .

这里有两个限制条件, 可以固定  $b_i \leq a_i$  是必须满足的条件, 只对  $b_i \geq b_{i-1}$  使用容斥, 枚举哪些位置是比上一位小的(违反限制), 其他位置随意.

枚举后的形态一定是有若干个区间是严格递减的, 其他位置随意. 考虑如果一个区间  $[l, r]$  是严格递减的, 显然所有的数都  $< a_l$ , 所

以这段区间的方案数就是  $\binom{a_i}{r-l+1}$ . 另外实际上  $b_l$  是没有违反限制的, 所以这里对系数的贡献是  $(-1)^{r-l}$ .

考虑令  $dp_i$  表示只考虑前  $i$  个位置的答案, 转移时自然就是枚举一个  $j$ , 然后计算  $dp_{j-1}$  乘上区间  $[j, i]$  严格递减的方案数. 另外还有一种情况是  $b_i$  没有违反限制, 这时显然直接在  $dp_{i-1}$  的基础上乘上一个  $a_i$  就好了. (转移时还要注意, 由于枚举的是严格递减区间, 自然就不能枚举只有一个数的区间.)

```

1 constexpr int maxn = 5005, p = (int)1e9 + 7;
2
3 int inv[maxn];
4 int a[maxn], f[maxn][maxn], dp[maxn];
5
6 int main() {
7
8     int n, m;
9     scanf("%d%d", &n, &m);
10
11     inv[1] = 1;
12     for (int i = 2; i ≤ n; i++)
13         inv[i] = (long long)(p - p / i) * inv[p % i] %
14             ↪ p;
15
16     for (int i = 1; i ≤ n; i++) {
17         scanf("%d", &a[i]);
18         a[i] = m - a[i] + 1;
19     }
20
21     if (any_of(a + 1, a + n + 1, [] (int x) {return x
22         ↪ ≤ 0;})) {
23         printf("0\n");
24         return 0;
25     }
26
27     for (int i = n - 1; i; i--)
28         a[i] = min(a[i], a[i + 1]);
29
30     // b_i ≥ b_{i-1} && b_i ≤ a_i
31     // 我们可以假设 b_i ≤ a_i 是必定被满足的, 然后对 b_i
32     ↪ 非严格递增的条件进行容斥, 枚举某一段是严格递减的
33     // 如果 [j, i] 严格递减, 显然它们都 ≤ a_j, 所以这个区
34     ↪ 间的方案数是 {a_j \choose i-j+1}
35     // 如果 i 是合法的, 直接一个个转移即可, 因为这一部分的
36     ↪ 转移和区间长度没有关系
37
38     for (int i = 1; i ≤ n; i++) {
39         f[i][0] = 1;
40
41         for (int j = 1; j ≤ n - i + 1 && j ≤ a[i];
42             ↪ j++)
43             f[i][j] = (long long)f[i][j - 1] * (a[i] -
44                 ↪ j + 1) % p * inv[j] % p;
45     }
46
47     dp[0] = 1;
48
49     for (int i = 1; i ≤ n; i++) {
50         dp[i] = (long long)dp[i - 1] * a[i] % p;
51
52         for (int j = 1; j < i; j++) {
53             int tmp = (long long)dp[j - 1] * f[j][i -
54                 ↪ j + 1] % p;
55
56             if ((i - j) % 2)
57                 tmp = p - tmp;
58
59             dp[i] = (dp[i] + tmp) % p;
60         }
61     }
62 }

```

```

55 printf("%d\n", dp[n]);
56
57 return 0;
58 }

```

## 7 计算几何

### 7.1 最近点对

首先分治的做法是众所周知的.

有期望  $O(n)$  的随机增量法: 首先将所有点随机打乱, 然后每次增加一个点, 更新答案.

假设当前最近点对距离为  $s$ , 则把平面划分成  $s \times s$  的方格, 用哈希表存储每个方格有哪些点.

加入一个新点时只需要枚举自身和周围共计 9 个方格中的点, 显然枚举到的点最多 16 个. 如果加入之后答案变小了, 就  $O(n)$  暴力重构.

前  $i$  个点中  $i$  是最近点对中的点的概率至多为  $\frac{2}{i}$ , 所以每个点的期望贡献都是  $O(1)$ , 总的复杂度就是期望  $O(n)$ .

如果对每个点都要求出距离最近的点的话, 也有随机化的  $O(n)$  做法:

一个真的随机算法:

A simple randomized sieve algorithm for the closest-pair problem (<https://www.grantcp.com/>)

1. 循环直到删完所有点:

- 随机选一个点, 计算它到所有点的最短距离  $d$ .
- 将所有点划分到  $l = d/3$  的网格里, 比如  $(\lfloor \frac{x}{l} \rfloor, \lfloor \frac{y}{l} \rfloor)$ .
- 将九宫格内孤立的点删除, 这意味着这些点的最近点对距离不小于  $\frac{2\sqrt{2}}{3}d$ .

2. 取最后一个  $d$ , 将所有点划分到  $(\lfloor \frac{x}{d} \rfloor, \lfloor \frac{y}{d} \rfloor)$  的网格里, 暴力计算九宫格内的

第一部分每次期望会删掉至少一半的点, 因为有  $\geq 1/2$  概率随到一个最近点距离. 此第一部分的复杂度是  $O(n)$  的.

第二部分分析类似分治做法, 周围只有常数个点.

所以总复杂度是  $O(n)$  的.

👍 42    🗨 6 条评论

## 8 杂项

### 8.1 STL

#### 8.1.1 vector

- `vector(int nSize)`: 创建一个 vector, 元素个数为 `nSize`
- `vector(int nSize, const T &value)`: 创建一个 vector, 元素个数为 `nSize`, 且值均为 `value`
- `vector(begin, end)`: 复制 `[begin, end)` 区间内另一个数组的元素到 vector 中
- `void assign(int n, const T &x)`: 设置向量中前 `n` 个元素的值为 `x`
- `void assign(const_iterator first, const_iterator last)`: 向量中 `[first, last)` 中元素设置成当前向量元素
- `void emplace_back(Args&&... args)`: 自动构造并 `push_back` 一个元素, 例如对一个存储 `pair` 的 vector 可以 `v.emplace_back(x, y)`

#### 8.1.2 list

- `assign()` 给 list 赋值
- `back()` 返回最后一个元素
- `begin()` 返回指向第一个元素的迭代器



- `clear()` 删除所有元素
- `empty()` 如果list是空的则返回true
- `end()` 返回末尾的迭代器
- `erase()` 删除一个元素
- `front()` 返回第一个元素
- `insert()` 插入一个元素到list中
- `max_size()` 返回list能容纳的最大元素数量
- `merge()` 合并两个list
- `pop_back()` 删除最后一个元素
- `pop_front()` 删除第一个元素
- `push_back()` 在list的末尾添加一个元素
- `push_front()` 在list的头部添加一个元素
- `rbegin()` 返回指向第一个元素的逆向迭代器
- `remove()` 从list删除元素
- `remove_if()` 按指定条件删除元素
- `rend()` 指向list末尾的逆向迭代器
- `resize()` 改变list的大小
- `reverse()` 把list的元素倒转
- `size()` 返回list中的元素个数
- `sort()` 给list排序
- `splice()` 合并两个list
- `swap()` 交换两个list
- `unique()` 删除list中重复的元

### 8.1.3 unordered\_set/map

- `unordered_map<int, int, hash>`: 自定义哈希函数，其中hash是一个带重载括号的类。

## 8.2 德州扑克

一般来说德州扑克里Ace都是最大的，所以把Ace的点数规定为14会好写许多。

附一个高低奥马哈的参考代码，除了有四张底牌和需要比低之外和德州区别不大。

```

1 struct Card {
2     int suit, value; // Ace is treated as 14
3
4     Card(string s) {
5         char a = s[0];
6
7         if (isdigit(a))
8             value = a - '0';
9         else if (a == 'T')
10            value = 10;
11        else if (a == 'A')
12            value = 14;
13        else if (a == 'J')
14            value = 11;
15        else if (a == 'Q')
16            value = 12;
17        else if (a == 'K')
18            value = 13;
19        else
20            value = -1; // error
21
22        char b = s[1];
23        suit = b; // Club, Diamond, Heart, Spade
24    }
25
26    friend bool operator < (const Card &a, const Card
27        ↪ &b) {
28        return a.value < b.value;
29    }
30
31    friend bool operator == (const Card &a, const Card
32        ↪ &b) {

```

```

31         return a.value == b.value;
32     }
33 };
34
35 constexpr int Highcard = 1, Pair = 2, TwoPairs = 3,
36     ↪ ThreeofaKind = 4, Straight = 5,
37     Flush = 6, FullHouse = 7, FourofaKind = 8,
38     ↪ StraightFlush = 9;
39
40 struct Hand {
41     vector<Card> v;
42     int type;
43
44     Hand() : type(0) {}
45
46     Hand(const Hand &o) : v(o.v), type(o.type) {}
47
48     Hand(const vector<Card> &v) : v(v), type(0) {}
49
50     void init_high() {
51         sort(v.begin(), v.end()); // 升序排序
52
53         bool straight = false;
54         if (v.back().value == 14) {
55             if (v[0].value == 2 && v[1].value == 3 &&
56                 ↪ v[2].value == 4 && v[3].value == 5) {
57                 straight = true;
58                 rotate(v.begin(), v.begin() + 1,
59                     ↪ v.end());
60             }
61         }
62
63         if (!straight) {
64             bool ok = true;
65             for (int i = 1; i < 5; i++)
66                 ok &= (v[i].value == v[i - 1].value +
67                     ↪ 1);
68
69             if (ok)
70                 straight = true;
71         }
72
73         bool flush = all_of(v.begin(), v.end(), [&]
74             ↪ (const Card &a) {return a.suit ==
75             ↪ v.front().suit;});
76
77         if (flush && straight) { // 同花顺
78             type = StraightFlush;
79             reverse(v.begin(), v.end());
80             return;
81         }
82
83         vector<int> c;
84         c.assign(15, 0);
85
86         for (auto &o : v)
87             c[o.value]++;
88
89         vector<int> kind[5];
90
91         for (int i = 2; i ≤ 14; i++)
92             if (c[i] > 1)
93                 kind[c[i]].push_back(i);
94
95         if (!kind[4].empty()) { // 四条
96             type = FourofaKind;
97
98             for (int i = 0; i < 4; i++)
99                 if (v[i].value != kind[4].front()) {
100                     swap(v[i], v.back());

```



```

94         break;
95     }
96
97     return;
98 }
99
100 if (!kind[3].empty() && !kind[2].empty()) {
101     type = FullHouse;
102
103     sort(v.begin(), v.end(), [&] (const Card
104         ↪ &a, const Card &b) {
105         bool ta = (a.value ==
106             ↪ kind[3].front()), tb = (b.value ==
107             ↪ kind[3].front());
108
109         return ta > tb;
110     });
111
112     return;
113 }
114
115 if (flush) {
116     type = Flush;
117     sort(v.begin(), v.end());
118     reverse(v.begin(), v.end());
119
120     return;
121 }
122
123 if (straight) {
124     type = Straight;
125     reverse(v.begin(), v.end());
126     return;
127 }
128
129 if (!kind[3].empty()) {
130     type = ThreeofaKind;
131
132     sort(v.begin(), v.end(), [&] (const Card
133         ↪ &a, const Card &b) {
134         bool ta = (a.value ==
135             ↪ kind[3].front()), tb = (b.value ==
136             ↪ kind[3].front());
137
138         return ta > tb;
139     });
140
141     if (v[3] < v[4])
142         swap(v[3], v[4]);
143
144     return;
145 }
146
147 if ((int)kind[2].size() == 2) {
148     type = TwoPairs;
149
150     sort(v.begin(), v.end(), [&] (const Card
151         ↪ &a, const Card &b) {
152         bool ta = (c[a.value] == 2), tb =
153             ↪ (c[b.value] == 2);
154
155         if (ta != tb)
156             return ta > tb;
157
158         return a.value > b.value;
159     });
160
161     return;
162 }

```

```

156     if ((int)kind[2].size() == 1) {
157         type = Pair;
158
159         sort(v.begin(), v.end(), [&] (const Card
160             ↪ &a, const Card &b) {
161             bool ta = (c[a.value] == 2), tb =
162                 ↪ (c[b.value] == 2);
163
164             if (ta != tb)
165                 return ta > tb;
166
167             return a.value > b.value;
168         });
169
170         return;
171     }
172
173     type = Highcard;
174
175     sort(v.begin(), v.end());
176     reverse(v.begin(), v.end());
177 }
178
179 void init_low() {
180     for (auto &o : v)
181         if (o.value == 14)
182             o.value = 1;
183
184     sort(v.begin(), v.end());
185     reverse(v.begin(), v.end());
186 }
187
188 friend int cmp_high(const Hand &a, const Hand &b)
189     ↪ {
190     if (a.type != b.type)
191         return a.type < b.type ? -1 : 1;
192
193     if (a.v != b.v)
194         return a.v < b.v ? -1 : 1;
195
196     return 0;
197 }
198
199 friend bool small_high(const Hand &a, const Hand
200     ↪ &b) {
201     return cmp_high(a, b) < 0;
202 }
203
204 friend int cmp_low(const Hand &a, const Hand &b) {
205     for (int i = 0; i < 5; i++)
206         if (a.v[i].value != b.v[i].value)
207             return a.v[i] < b.v[i] ? 1 : -1;
208
209     return 0;
210 }
211
212 friend bool small_low(const Hand &a, const Hand
213     ↪ &b) {
214     return cmp_low(a, b) < 0;
215 }
216
217 bool operator ! () const {
218     return v.empty();
219 }
220
221 string str() const {
222     stringstream ss;
223
224     for (auto &o : v)

```

```

220         ss << o.value << ' ';
221
222     return ss.str();
223 }
224 };
225
226 Hand get_max_high(vector<Card> u, vector<Card> v) { //
    ↪ private, public
227     Hand ans;
228
229     for (int i = 0; i < 4; i++)
230         for (int j = i + 1; j < 4; j++)
231             for (int k = 0; k < 5; k++)
232                 for (int p = k + 1; p < 5; p++)
233                     for (int q = p + 1; q < 5; q++) {
234                         Hand tmp({u[i], u[j], v[k],
    ↪ v[p], v[q]});
235
236                         tmp.init_high();
237
238                         if (!ans || cmp_high(tmp, ans)
    ↪ > 0)
239                             ans = tmp;
240                     }
241
242     return ans;
243 }
244
245 Hand get_max_low(vector<Card> tu, vector<Card> tv) {
246     vector<Card> u, v;
247
248     for (auto o : tu)
249         if (o.value == 14 || o.value ≤ 8)
250             u.push_back(o);
251
252     for (auto o : tv)
253         if (o.value == 14 || o.value ≤ 8)
254             v.push_back(o);
255
256     Hand ans;
257
258     for (int i = 0; i < (int)u.size(); i++)
259         for (int j = i + 1; j < (int)u.size(); j++)
260             for (int k = 0; k < (int)v.size(); k++)
261                 for (int p = k + 1; p < (int)v.size();
    ↪ p++)
262                     for (int q = p + 1; q <
    ↪ (int)v.size(); q++) {
263                         vector<Card> vec = {u[i],
    ↪ u[j], v[k], v[p], v[q]};
264
265                         bool bad = false;
266
267                         for (int a = 0; a < 5; a++)
268                             for (int b = a + 1; b < 5;
    ↪ b++)
269                                 if (vec[a].value ==
    ↪ vec[b].value)
270                                     bad = true;
271
272                         if (bad)
273                             continue;
274
275                         Hand tmp(vec);
276
277                         tmp.init_low();
278
279                         if (!ans || cmp_low(tmp, ans)
    ↪ > 0)
280                             ans = tmp;

```

```

281     }
282
283     return ans;
284 }
285
286 int main() {
287
288     ios::sync_with_stdio(false);
289
290     int T;
291     cin >> T;
292
293     while (T--) {
294         int p;
295         cin >> p;
296
297         vector<Card> alice, bob, pub;
298
299         for (int i = 0; i < 4; i++) {
300             string s;
301             cin >> s;
302             alice.push_back(Card(s));
303         }
304
305         for (int i = 0; i < 4; i++) {
306             string s;
307             cin >> s;
308             bob.push_back(Card(s));
309         }
310
311         for (int i = 0; i < 5; i++) {
312             string s;
313             cin >> s;
314             pub.push_back(Card(s));
315         }
316
317         Hand alice_high = get_max_high(alice, pub),
    ↪ bob_high = get_max_high(bob, pub);
318         Hand alice_low = get_max_low(alice, pub),
    ↪ bob_low = get_max_low(bob, pub);
319
320         int dh = cmp_high(alice_high, bob_high);
321         int ans[2] = {0};
322
323         if (!alice_low && !bob_low) {
324             if (!dh) {
325                 ans[0] = p - p / 2;
326                 ans[1] = p / 2;
327             }
328             else
329                 ans[dh == -1] = p;
330         }
331         else if (!alice_low || !bob_low) {
332             ans[!alice_low] += p / 2;
333
334             if (!dh) {
335                 ans[0] += p - p / 2 - (p - p / 2) / 2;
336                 ans[1] += (p - p / 2) / 2;
337             }
338             else
339                 ans[dh == -1] += p - p / 2;
340         }
341         else {
342             int dl = cmp_low(alice_low, bob_low);
343
344             if (!dl) {
345                 ans[0] += p / 2 - p / 2 / 2;
346                 ans[1] += p / 2 / 2;
347             }

```

```
348 |         else
349 |             ans[dl == -1] += p / 2;
350 |
351 |         if (!dh) {
352 |             ans[0] += p - p / 2 - (p - p / 2) / 2;
353 |             ans[1] += (p - p / 2) / 2;
354 |         }
355 |     else
```

```
356 |         ans[dh == -1] += p - p / 2;
357 |     }
358 |
359 |     cout << ans[0] << ' ' << ans[1] << '\n';
360 | }
361 |
362 | return 0;
363 | }
```



# Visual Studio Code

## Keyboard shortcuts for Windows

### General

Ctrl+Shift+P, F1	Show Command Palette
Ctrl+P	Quick Open, Go to File...
Ctrl+Shift+N	New window/Instance
Ctrl+Shift+W	Close window/Instance
Ctrl+,	User Settings
Ctrl+K Ctrl+S	Keyboard Shortcuts

### Basic editing

Ctrl+X	Cut line (empty selection)
Ctrl+C	Copy line (empty selection)
Alt+ ↑ / ↓	Move line up/down
Shift+Alt + ↑ / ↓	Copy line up/down
Ctrl+Shift+K	Delete line
Ctrl+Enter	Insert line below
Ctrl+Shift+Enter	Insert line above
Ctrl+Shift+\<	Jump to matching bracket
Ctrl+ ) / [	Indent/outdent line
Home / End	Go to beginning/end of line
Ctrl+Home	Go to beginning of file
Ctrl+End	Go to end of file
Ctrl+↑ / ↓	Scroll line up/down
Alt+PgUp / PgDn	Scroll page up/down
Ctrl+Shift+I	Fold (collapse) region
Ctrl+Shift+J	Unfold (uncollapse) region
Ctrl+K Ctrl+[	Fold (collapse) all subregions
Ctrl+K Ctrl+]	Unfold (uncollapse) all subregions
Ctrl+K Ctrl+0	Fold (collapse) all regions
Ctrl+K Ctrl+J	Unfold (uncollapse) all regions
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+/ Shift+Alt+A	Toggle line comment Toggle block comment
Alt+Z	Toggle word wrap

### Navigation

Ctrl+T	Show all Symbols
Ctrl+G	Go to Line...
Ctrl+P	Go to File...
Ctrl+Shift+O	Go to Symbol...
Ctrl+Shift+M	Show Problems panel
F8	Go to next error or warning
Shift+F8	Go to previous error or warning
Ctrl+Shift+Tab	Navigate editor group history
Alt+ ← / →	Go back / forward

Ctrl+M

Toggle Tab moves focus

### Search and replace

Ctrl+F	Find
Ctrl+H	Replace
F3 / Shift+F3	Find next/previous
Alt+Enter	Select all occurrences of Find match
Ctrl+D	Add selection to next Find match
Ctrl+K Ctrl+D	Move last selection to next Find match
Alt+C / R / W	Toggle case-sensitive / regex / whole word

### Multi-cursor and selection

Alt+Click	Insert cursor
Ctrl+Alt+ ↑ / ↓	Insert cursor above / below
Ctrl+U	Undo last cursor operation
Shift+Alt+I	Insert cursor at end of each line selected
Ctrl+L	Select current line
Ctrl+Shift+L	Select all occurrences of current selection
Ctrl+F2	Select all occurrences of current word
Shift+Alt+→	Expand selection
Shift+Alt+←	Shrink selection
Shift+Alt+ (drag mouse)	Column (box) selection
Ctrl+Shift+Alt + (arrow key)	Column (box) selection
Ctrl+Shift+Alt + PgUp/PgDn	Column (box) selection page up/down

### Rich languages editing

Ctrl+Space, Ctrl+I	Trigger suggestion
Ctrl+Shift+Space	Trigger parameter hints
Shift+Alt+F	Format document
Ctrl+K Ctrl+F	Format selection
F12	Go to Definition
Alt+F12	Peek Definition
Ctrl+K F12	Open Definition to the side
Ctrl+. Shift+F12	Quick Fix Show References
F2	Rename Symbol
Ctrl+K Ctrl+X	Trim trailing whitespace
Ctrl+K M	Change file language

### Editor management

Ctrl+F4, Ctrl+W	Close editor
Ctrl+K F	Close folder
Ctrl+\ Ctrl+ 1 / 2 / 3	Split editor Focus into 1 <sup>st</sup> , 2 <sup>nd</sup> or 3 <sup>rd</sup> editor group
Ctrl+K Ctrl+ ←/→	Focus into previous/next editor group
Ctrl+Shift+PgUp / PgDn	Move editor left/right
Ctrl+K ← / →	Move active editor group

### File management

Ctrl+N	New File
Ctrl+O	Open File...
Ctrl+S	Save
Ctrl+Shift+S	Save As...
Ctrl+K S	Save All
Ctrl+F4	Close
Ctrl+K Ctrl+W	Close All
Ctrl+Shift+T	Reopen closed editor
Ctrl+K Enter	Keep preview mode editor open
Ctrl+Tab	Open next
Ctrl+Shift+Tab	Open previous
Ctrl+K P	Copy path of active file
Ctrl+K R	Reveal active file in Explorer
Ctrl+K O	Show active file in new window/instance

### Display

F11	Toggle full screen
Shift+Alt+O	Toggle editor layout (horizontal/vertical)
Ctrl+ = / -	Zoom in/out
Ctrl+B	Toggle Sidebar visibility
Ctrl+Shift+E	Show Explorer / Toggle focus
Ctrl+Shift+F	Show Search
Ctrl+Shift+G	Show Source Control
Ctrl+Shift+D	Show Debug
Ctrl+Shift+X	Show Extensions
Ctrl+Shift+H	Replace in files
Ctrl+Shift+J	Toggle Search details
Ctrl+Shift+U	Show Output panel
Ctrl+Shift+V	Open Markdown preview
Ctrl+K V	Open Markdown preview to the side
Ctrl+K Z	Zen Mode (Esc Esc to exit)

### Debug

F9	Toggle breakpoint
F5	Start/Continue
Shift+F5	Stop
F11 / Shift+F11	Step into/out
F10	Step over
Ctrl+K Ctrl+I	Show hover

### Integrated terminal

Ctrl+`	Show integrated terminal
Ctrl+Shift+`	Create new terminal
Ctrl+C	Copy selection
Ctrl+V	Paste into active terminal
Ctrl+↑ / ↓	Scroll up/down
Shift+PgUp / PgDn	Scroll page up/down
Ctrl+Home / End	Scroll to top/bottom

Other operating systems' keyboard shortcuts and additional assigned shortcuts available at [aka.ms/vscodekeybindings](https://aka.ms/vscodekeybindings)



# Visual Studio Code

## Keyboard shortcuts for Linux

### General

Ctrl+Shift+P, F1	Show Command Palette
Ctrl+P	Quick Open, Go to File...
Ctrl+Shift+N	New window/Instance
Ctrl+W	Close window/Instance
Ctrl+,	User Settings
Ctrl+K Ctrl+S	Keyboard Shortcuts

### Basic editing

Ctrl+X	Cut line (empty selection)
Ctrl+C	Copy line (empty selection)
Alt+ 1 / 1	Move line down/up
Ctrl+Shift+K	Delete line
Ctrl+Enter / Ctrl+Shift+Enter	Insert line below/ above
Ctrl+Shift+\	Jump to matching bracket
Ctrl+)] / Ctrl+[	Indent/Outdent line
Home / End	Go to beginning/end of line
Ctrl+ Home / End	Go to beginning/end of file
Ctrl+ 1 / 1	Scroll line up/down
Alt+ PgUp / PgDn	Scroll page up/down
Ctrl+Shift+ [ / ]	Fold/unfold region
Ctrl+K Ctrl+ [ / ]	Fold/unfold all subregions
Ctrl+K Ctrl+0 /	Fold/Unfold all regions
Ctrl+K Ctrl+J	
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+ /	Toggle line comment
Ctrl+Shift+A	Toggle block comment
Alt+Z	Toggle word wrap

### Rich languages editing

Ctrl+Space, Ctrl+I	Trigger suggestion
Ctrl+Shift+Space	Trigger parameter hints
Ctrl+Shift+I	Format document
Ctrl+K Ctrl+F	Format selection
F12	Go to Definition
Ctrl+Shift+F10	Peek Definition
Ctrl+K F12	Open Definition to the side
Ctrl+,	Quick Fix
Shift+F12	Show References
F2	Rename Symbol
Ctrl+K Ctrl+X	Trim trailing whitespace
Ctrl+K M	Change file language

### Multi-cursor and selection

Alt+Click	Insert cursor*
Shift+Alt+ 1 / 1	Insert cursor above/below
Ctrl+U	Undo last cursor operation
Shift+Alt+I	Insert cursor at end of each line selected
Ctrl+L	Select current line
Ctrl+Shift+L	Select all occurrences of current selection
Ctrl+F2	Select all occurrences of current word
Shift+Alt + →	Expand selection
Shift+Alt + ←	Shrink selection
Shift+Alt + drag mouse	Column (box) selection

### Display

F11	Toggle full screen
Shift+Alt+0	Toggle editor layout (horizontal/vertical)
Ctrl+ = / -	Zoom in/out
Ctrl+B	Toggle Sidebar visibility
Ctrl+Shift+E	Show Explorer / Toggle focus
Ctrl+Shift+F	Show Search
Ctrl+Shift+G	Show Source Control
Ctrl+Shift+D	Show Debug
Ctrl+Shift+X	Show Extensions
Ctrl+Shift+H	Replace in files
Ctrl+Shift+J	Toggle Search details
Ctrl+Shift+C	Open new command prompt/terminal
Ctrl+K Ctrl+H	Show Output panel
Ctrl+Shift+V	Open Markdown preview
Ctrl+K V	Open Markdown preview to the side
Ctrl+K Z	Zen Mode (Esc Esc to exit)

### Search and replace

Ctrl+F	Find
Ctrl+H	Replace
F3 / Shift+F3	Find next/previous
Alt+Enter	Select all occurrences of Find match
Ctrl+D	Add selection to next Find match
Ctrl+K Ctrl+D	Move last selection to next Find match

### Navigation

Ctrl+T	Show all Symbols
Ctrl+G	Go to Line...
Ctrl+P	Go to File...
Ctrl+Shift+O	Go to Symbol...
Ctrl+Shift+M	Show Problems panel
F8	Go to next error or warning
Shift+F8	Go to previous error or warning
Ctrl+Shift+Tab	Navigate editor group history
Ctrl+Alt+-	Go back
Ctrl+Shift+-	Go forward
Ctrl+M	Toggle Tab moves focus

### Editor management

Ctrl+W	Close editor
Ctrl+K F	Close folder
Ctrl+\	Split editor
Ctrl+ 1 / 2 / 3	Focus into 1 <sup>st</sup> , 2 <sup>nd</sup> , 3 <sup>rd</sup> editor group
Ctrl+K Ctrl + ←	Focus into previous editor group
Ctrl+K Ctrl + →	Focus into next editor group
Ctrl+Shift+PgUp	Move editor left
Ctrl+Shift+PgDn	Move editor right
Ctrl+K ←	Move active editor group left/up
Ctrl+K →	Move active editor group right/down

### File management

Ctrl+N	New File
Ctrl+O	Open File...
Ctrl+S	Save
Ctrl+Shift+S	Save As...
Ctrl+W	Close
Ctrl+K Ctrl+W	Close All
Ctrl+Shift+T	Reopen closed editor
Ctrl+K Enter	Keep preview mode editor open
Ctrl+Tab	Open next
Ctrl+Shift+Tab	Open previous
Ctrl+K P	Copy path of active file
Ctrl+K R	Reveal active file in Explorer
Ctrl+K O	Show active file in new window/Instance

### Debug

F9	Toggle breakpoint
F5	Start / Continue
F11 / Shift+F11	Step into/out
F10	Step over
Shift+F5	Stop
Ctrl+K Ctrl+I	Show hover

### Integrated terminal

Ctrl+`	Show integrated terminal
Ctrl+Shift+`	Create new terminal
Ctrl+Shift+C	Copy selection
Ctrl+Shift+V	Paste into active terminal
Ctrl+Shift+ 1 / 1	Scroll up/down
Shift+ PgUp / PgDn	Scroll page up/down
Shift+ Home / End	Scroll to top/bottom

\* The Alt+Click gesture may not work on some Linux distributions. You can change the modifier key for the Insert cursor command to Ctrl+Click with the “editor.multiCursorModifier” setting.