# Standard Code Library

Shanghai Jiao Tong University October, 2015

# Contents

1	数论		3
	1.1	快速数论变换	3
	1.2	多项式求逆	3
	1.3	中国剩余定理	4
	1.4	Miller Rabin	4
	1.5	Pollard Rho	Ę
	1.6	坚固的逆元	Ę
	1.7	直线下整点个数	Į
<b>2</b>	数值		7
	2.1	快速傅立叶变换	7
	2.2	单纯形法求解线性规划	7
	2.3	自适应辛普森	6
3	数据		10
	3.1		10
	3.2	1 0 -1 42/11/10	11
	3.3		15
	3.4		18
	3.5	14.461.424	19
			19
			22
	3.6	Link-Cut-Tree	22
	र्यन्त		۰.
4	图论		25 25
	4.1		
	4.2	MIV (6.6)	25
	4.0		25
	4.3	, 00	27
	4.4	7. T.	28
		0 0 0 1 10	28
		1 1 21 1	29
	4.5	, , , , , , , , , , , , , , , , , , ,	30
	4.6	· ### · ###	30
			30
			31
			32
	4.7		33
			33
		14 444-44-4-1 21 714-44	33
		14 444-44-4-1 21 1442 1448	33
		17 ***** - 1 21 *** * 10	33
	4.8	. 6. 4. 5. 1/10. 6.1/. 10.00	33
		4.8.1 稀疏图	33
			34
	4.9	一般图最大匹配	35
	4.10	无向图全局最小割	37
	4.11	最小树形图	38
	4.12	有根树的同构	39
			39
			41
		4.14.1 弦图的判定	41

CONTENTS 3

	4.15	4.14.2 哈密尔	弦图的[ 顿回路	团数 (ORE	· · · · · · · · · · · · · · · · · · ·	 质的	 图)	 		 	 	 	 	 		 	 					42 43
E	字符	ф																				46
J		中 模式串	DT #id																			
	5.1 5.2					· · · · · ·																46
			模式串四	_,																		40
	5.3	AC 自z	f																			
	5.4	后缀数																				48
	5.5		缀自动机																			48
	5.6		her 算法																			49
	5.7		· · · · ·																			
	5.8	循环串	最小表示	<b>г</b>				 	•	 	 	 	 	 			 		•			50
_	) I &&																					
6	计算		<b></b> 1.																			51
	6.1	二维基																				
		6.1.1	点类 .																			51
		6.1.2																				
		6.1.3	半平面	-				 		 	 	 	 	 			 					52
		6.1.4	最近点为	付 .				 		 	 	 	 	 			 					52
	6.2	三维基	础					 		 	 	 	 	 			 					53
		6.2.1	点类 .					 		 	 	 	 	 			 					53
		6.2.2	凸包 .					 		 	 	 	 	 			 					54
		6.2.3	绕轴旋轴	传 .				 		 	 	 	 	 			 					56
	6.3	多边形						 		 	 	 	 	 			 					57
		6.3.1	判断点征	在多计	形口	カ部 かいかい かいかい かいかい かいかい かいかい かいかい かいかい かい		 		 	 	 	 	 			 					57
		6.3.2	多边形																			57
	6.4	圆																				
		6.4.1	最小覆																			
		6.4.2	最小覆																			
		6.4.3	多边形																			
		0.1.0	2 KIN-	) MH		ЩИЛ		 	•	 	 	 	 	 • •	• •	• •	 		•			01
7	其它																					62
	7.1		用方法					 		 	 	 	 	 			 					62
	• • •	7.1.1	nth ele																			
		7.1.2	next p																			62
	7.2	博弈论																				62
	1.2	7.2.1	四人 巴什博			· · ·																62
		7.2.1 $7.2.2$	威佐夫	-																		62
		7.2.2 $7.2.3$	阶梯博			· · ·		 		 	 	 	 	 -			 			-	-	62
				-																		
	7.0	7.2.4	图上删记	_ / · ·	-																	
	7.3	Java K	eference					 	•	 	 	 	 	 			 		•			63
8	数学	人士																				65
G	8.1		学公式																			
	0.1	市用奴 8.1.1	子公八 求和公司																			65
																						65
		8.1.2	斐波那																			
		8.1.3	错排公司																			65
			莫比乌																			65
			Burnsid																			66
		8.1.6	五边形势																			66
		8.1.7	树的计数																			66
		8.1.8	欧拉公司	式 .				 	•	 	 	 	 	 			 		•			66
		8.1.9	皮克定理																			66
		8.1.10	牛顿恒等	等式				 		 	 	 	 	 			 					67
	8.2	平面几	何公式					 		 	 	 	 	 			 					67
		8.2.1	三角形					 		 	 	 	 	 			 					67
		8.2.2	四边形					 		 	 	 	 	 			 					67
		8.2.3	正加边	形 .				 		 	 	 	 	 			 					68
		8.2.4	圆					 		 	 	 	 	 			 					68
		8.2.5	_ 棱柱 .					 		 	 	 	 	 			 					68
		8.2.6	1241																			68
		8.2.7																				69
		8.2.8																				69
		8.2.9	圆锥.																			
		- · <del>- · ·</del>	• ملالا بنتر						-	 	 	 	 	 			 	- •	-			00

4 CONTENTS

	8.2.10	圆台		 		 									 				6
	8.2.11	球		 		 									 				6
	8.2.12	球台		 		 									 				7
	8.2.13	球扇形.		 		 									 				7
8.3	立体几	何公式 .		 		 									 				7
	8.3.1	球面三角	公式 .	 		 									 				7
	8.3.2	四面体体	积公式	 		 									 				7

## Chapter 1

# 数论算法

## 1.1 快速数论变换

使用条件及注意事项: mod 必须要是一个形如  $a2^b + 1$  的数, prt 表示 mod 的原根。

```
const int mod = 998244353;
1
     const int prt = 3;
     int prepare(int n) {
3
         int len = 1;
         for (; len <= 2 * n; len <<= 1);
5
         for (int i = 0; i <= len; i++) {
6
             e[0][i] = fpm(prt, (mod - 1) / len * i, mod);
7
             e[1][i] = fpm(prt, (mod - 1) / len * (len - i), mod);
9
10
         return len;
11
     void DFT(int *a, int n, int f) {
12
         for (int i = 0, j = 0; i < n; i++) {
13
             if (i > j) std::swap(a[i], a[j]);
14
15
             for (int t = n >> 1; (j \hat{} = t) < t; t >>= 1);
16
         for (int i = 2; i <= n; i <<= 1)
17
             for (int j = 0; j < n; j += i)
18
                  for (int k = 0; k < (i >> 1); k++) {
19
20
                      int A = a[j + k];
                      int B = (long long)a[j + k + (i >> 1)] * e[f][n / i * k] % mod;
21
                      a[j + k] = (A + B) \% mod;
22
                      a[j + k + (i >> 1)] = (A - B + mod) \% mod;
23
24
         if (f == 1) {
25
             long long rev = fpm(n, mod - 2, mod);
26
             for (int i = 0; i < n; i++) {
27
                 a[i] = (long long)a[i] * rev % mod;
28
29
30
         }
     }
```

## 1.2 多项式求逆

使用条件及注意事项: 求一个多项式在模意义下的逆元。

```
void getInv(int *a, int *b, int n) {
    static int tmp[MAXN];
    std::fill(b, b + n, 0);
    b[0] = fpm(a[0], mod - 2, mod);
    for (int c = 1; c <= n; c <<= 1) {
        for (int i = 0; i < c; i++) tmp[i] = a[i];
        std::fill(b + c, b + (c << 1), 0);
}</pre>
```

CHAPTER 1. 数论算法

```
std::fill(tmp + c, tmp + (c << 1), 0);
9
             DFT(tmp, c \ll 1, 0);
             DFT(b, c << 1, 0);
10
             for (int i = 0; i < (c << 1); i++) {
11
                 b[i] = (long long)(2 - (long long)tmp[i] * b[i] % mod + mod) * b[i] % mod;
13
             DFT(b, c << 1, 1);
14
             std::fill(b + c, b + (c << 1), 0);
15
16
         }
     }
17
```

## 1.3 中国剩余定理

6

使用条件及注意事项:模数可以不互质。

```
bool solve(int n, std::pair<long long, long long> input[],
1
                        std::pair<long long, long long> &output) {
         output = std::make_pair(1, 1);
3
         for (int i = 0; i < n; ++i) {
4
             long long number, useless;
5
             euclid(output.second, input[i].second, number, useless);
6
             long long divisor = std::__gcd(output.second, input[i].second);
             if ((input[i].first - output.first) % divisor) {
8
                 return false;
10
             number *= (input[i].first - output.first) / divisor;
11
12
             fix(number, input[i].second);
             output.first += output.second * number;
13
             output.second *= input[i].second / divisor;
14
             fix(output.first, output.second);
15
         }
16
17
         return true;
     }
18
```

### 1.4 Miller Rabin

```
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
1
2
     bool check(const long long &prime, const long long &base) {
3
         long long number = prime - 1;
         for (; ~number & 1; number >>= 1);
5
         long long result = power_mod(base, number, prime);
6
         for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1) {
             result = multiply_mod(result, result, prime);
         return result == prime - 1 || (number & 1) == 1;
10
     }
11
12
     bool miller_rabin(const long long &number) {
13
         if (number < 2) {
14
             return false;
15
16
         if (number < 4) {
17
             return true;
18
19
20
         if (~number & 1) {
             return false;
21
         }
22
         for (int i = 0; i < 12 && BASE[i] < number; ++i) {</pre>
23
             if (!check(number, BASE[i])) {
^{24}
```

1.5. POLLARD RHO 7

### 1.5 Pollard Rho

```
long long pollard_rho(const long long &number, const long long &seed) {
1
         long long x = rand() \% (number - 1) + 1, y = x;
2
         for (int head = 1, tail = 2; ; ) {
3
             x = multiply_mod(x, x, number);
             x = add_mod(x, seed, number);
5
             if (x == y) {
6
7
                  return number;
             long long answer = std::__gcd(abs(x - y), number);
9
             if (answer > 1 && answer < number) {</pre>
10
                  return answer;
11
12
             if (++head == tail) {
13
14
                  y = x;
15
                  tail <<= 1;
             }
16
         }
17
     }
18
19
20
     void factorize(const long long &number, std::vector<long long> &divisor) {
         if (number > 1) {
21
             if (miller_rabin(number)) {
22
23
                  divisor.push_back(number);
24
                  long long factor = number;
25
                  for (; factor >= number;
26
                         factor = pollard_rho(number, rand() % (number - 1) + 1));
27
                  factorize(number / factor, divisor);
28
                  factorize(factor, divisor);
29
             }
30
         }
31
     }
32
```

## 1.6 坚固的逆元

```
long long inverse(const long long &x, const long long &mod) {
   if (x == 1) {
      return 1;
   } else {
      return (mod - mod / x) * inverse(mod % x, mod) % mod;
   }
}
```

## 1.7 直线下整点个数

CHAPTER 1. 数论算法

```
6     if (a >= m) {
7         return n * (a / m) + solve(n, a % m, b, m);
8     }
9     if (b >= m) {
10         return (n - 1) * n / 2 * (b / m) + solve(n, a, b % m, m);
11     }
12     return solve((a + b * n) / m, (a + b * n) % m, m, b);
13  }
```

## Chapter 2

# 数值算法

## 2.1 快速傅立叶变换

```
int prepare(int n) {
2
         int len = 1;
         for (; len <= 2 * n; len <<= 1);
3
         for (int i = 0; i < len; i++) {
             e[0][i] = Complex(cos(2 * pi * i / len), sin(2 * pi * i / len));
5
             e[1][i] = Complex(cos(2 * pi * i / len), -sin(2 * pi * i / len));
6
7
8
         return len;
9
10
     void DFT(Complex *a, int n, int f) {
11
         for (int i = 0, j = 0; i < n; i++) {
             if (i > j) std::swap(a[i], a[j]);
13
             for (int t = n >> 1; (j \hat{} = t) < t; t >>= 1);
14
15
         for (int i = 2; i <= n; i <<= 1)
16
             for (int j = 0; j < n; j += i)
17
                  for (int k = 0; k < (i >> 1); k++) {
18
                      Complex A = a[j + k];
19
                      Complex B = e[f][n / i * k] * a[j + k + (i >> 1)];
20
                      a[j + k] = A + B;
21
                      a[j + k + (i >> 1)] = A - B;
22
23
         if (f == 1) {
             for (int i = 0; i < n; i++)
25
                 a[i].a /= n;
26
         }
27
     }
28
```

## 2.2 单纯形法求解线性规划

使用条件及注意事项: 返回结果为  $max\{c_{1\times m}\cdot x_{m\times 1}\mid x_{m\times 1}\geq 0_{m\times 1}, a_{n\times m}\cdot x_{m\times 1}\leq b_{n\times 1}\}$ 

```
std::vector<double> solve(const std::vector<std::vector<double> > &a,
1
                                const std::vector<double> &b, const std::vector<double> &c) {
2
         int n = (int)a.size(), m = (int)a[0].size() + 1;
         std::vector<std::vector<double> > value(n + 2, std::vector<double>(m + 1));
4
         std::vector<int> index(n + m);
5
6
         int r = n, s = m - 1;
         for (int i = 0; i < n + m; ++i) {
             index[i] = i;
8
         }
9
         for (int i = 0; i < n; ++i) {
10
             for (int j = 0; j < m - 1; ++j) {
11
```

CHAPTER 2. 数值算法

```
value[i][j] = -a[i][j];
12
13
             value[i][m - 1] = 1;
14
             value[i][m] = b[i];
15
             if (value[r][m] > value[i][m]) {
16
17
                  r = i;
             }
18
         }
19
20
         for (int j = 0; j < m - 1; ++j) {
              value[n][j] = c[j];
21
22
         value[n + 1][m - 1] = -1;
23
         for (double number; ; ) {
24
              if (r < n) {
25
                  std::swap(index[s], index[r + m]);
26
                  value[r][s] = 1 / value[r][s];
27
                  for (int j = 0; j \le m; ++j) {
28
                      if (j != s) {
29
                           value[r][j] *= -value[r][s];
30
31
                      }
                  }
32
                  for (int i = 0; i <= n + 1; ++i) {
33
                      if (i != r) {
34
                           for (int j = 0; j \le m; ++j) {
35
36
                               if (j != s) {
                                   value[i][j] += value[r][j] * value[i][s];
37
38
39
                           value[i][s] *= value[r][s];
40
                      }
41
                  }
42
             }
43
             r = s = -1;
44
             for (int j = 0; j < m; ++j) {
45
                  if (s < 0 || index[s] > index[j]) {
46
47
                      if (value[n + 1][j] > eps | | value[n + 1][j] > -eps && value[n][j] > eps) {
                           s = j;
48
                      }
49
                  }
50
             }
51
             if (s < 0) {
52
                  break;
53
54
             for (int i = 0; i < n; ++i) {
55
                  if (value[i][s] < -eps) {
56
                      if (r < 0)
57
                      || (number = value[r][m] / value[r][s] - value[i][m] / value[i][s]) < -eps
58
                       | |  number < eps && index[r + m] > index[i + m]) {
59
                           r = i;
60
61
                      }
                  }
62
             }
63
             if (r < 0) {
64
                        Solution is unbounded.
65
                  return std::vector<double>();
             }
67
68
         if (value[n + 1][m] < -eps) {
69
                   No solution.
70
             return std::vector<double>();
71
         }
72
73
         std::vector<double> answer(m - 1);
74
         for (int i = m; i < n + m; ++i) {
```

2.3. 自适应辛普森 11

## 2.3 自适应辛普森

```
double area(const double &left, const double &right) {
2
         double mid = (left + right) / 2;
         return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
3
     }
4
5
     double simpson(const double &left, const double &right,
6
7
                    const double &eps, const double &area_sum) {
         double mid = (left + right) / 2;
8
         double area_left = area(left, mid);
9
         double area_right = area(mid, right);
10
         double area_total = area_left + area_right;
11
12
         if (std::abs(area_total - area_sum) < 15 * eps) {</pre>
             return area_total + (area_total - area_sum) / 15;
13
14
         return simpson(left, mid, eps / 2, area_left)
15
              + simpson(mid, right, eps / 2, area_right);
16
17
     }
18
     double simpson(const double &left, const double &right, const double &eps) {
19
         return simpson(left, right, eps, area(left, right));
20
     }
21
```

## Chapter 3

# 数据结构

## 3.1 Splay 普通操作版

使用条件及注意事项:

- 1. 插入 x 数
- 2. 删除 x 数 (若有多个相同的数,因只删除一个)
- 3. 查询 x 数的排名 (若有多个相同的数, 因输出最小的排名)
- 4. 查询排名为 x 的数
- 5. 求 x 的前驱 (前驱定义为小于 x, 且最大的数)
- 6. 求 x 的后继 (后继定义为大于 x, 且最小的数)

```
int pred(int x) {
1
         splay(x, -1);
2
         for (x = c[x][0]; c[x][1]; x = c[x][1]);
         return x;
4
     }
5
     int succ(int x) {
6
         splay(x, -1);
         for (x = c[x][1]; c[x][0]; x = c[x][0]);
8
9
         return x;
10
     void remove(int x) {
11
         if (b[x] > 1) \{b[x] --; splay(x, -1); return;\}
12
         splay(x, -1);
13
         if (!c[x][0] \&\& !c[x][1]) rt = 0;
         else if (c[x][0] \&\& !c[x][1]) f[rt = c[x][0]] = -1;
15
         else if (!c[x][0] \&\& c[x][1]) f[rt = c[x][1]] = -1;
16
         else{
17
             int t = pred(x); f[rt = c[x][0]] = -1;
             c[t][1] = c[x][1]; f[c[x][1]] = t;
19
             splay(c[x][1], -1);
20
21
         c[x][0] = c[x][1] = f[x] = d[x] = s[x] = b[x] = 0;
22
23
     int find(int z) {
24
         int x=rt;
25
         while (d[x]!=z)
26
             if (c[x][d[x]<z]) x=c[x][d[x]<z];
27
             else break;
28
29
         return x;
30
     void insert(int z) {
31
         if (!rt) {
32
             f[rt = ++size] = -1;
33
             d[size] = z; b[size] = 1;
34
```

3.2. SPLAY 区间操作版 13

```
35
             splay(size, -1);
             return;
36
         }
37
         int x = find(z);
38
         if (d[x] == z) {
39
             b[x]++;
40
             splay(x, -1);
41
42
             return;
43
         }
         c[x][d[x] < z] = ++size; f[size] = x;
44
         d[size] = z; b[size] = s[size] = 1;
45
         splay(size, -1);
46
47
     int select(int z) {
48
         int x = rt;
49
         while (z < s[c[x][0]] + 1 \mid | z > s[c[x][0]] + b[x])
50
             if (z > s[c[x][0]] + b[x]) {
51
                 z = s[c[x][0]] + b[x];
52
                 x = c[x][1];
53
54
               else x = c[x][0];
55
         return x;
56
     }
57
     int main() {
58
59
         scanf("%d",&n);
         for (int i = 1; i <= n; i++) {
60
             int opt, x;
61
             scanf("%d%d", &opt, &x);
62
             if (opt == 1) insert(x);
63
             else if (opt == 2) remove(find(x)); //删除 x 数 (若有多个相同的数, 因只删除一个)
64
             else if (opt == 3) { // 查询 x 数的排名 (若有多个相同的数,因输出最小的排名)
65
                  insert(x);
66
                  printf("\frac{n}{d}", s[c[find(x)][0]] + 1);
67
                 remove(find(x));
68
             }
69
             else if (opt == 4) printf("%d\n",d[select(x)]);
70
             else if (opt == 5) {
71
                  insert(x);
72
                  printf("%d\n", d[pred(find(x))]);
73
74
                 remove(find(x));
75
             else if (opt == 6) {
76
                 insert(x);
77
                 printf("%d\n", d[succ(find(x))]);
78
                 remove(find(x));
79
             }
80
81
82
         return 0;
83
```

## 3.2 Splay 区间操作版

使用条件及注意事项:

这是为 NOI2005 维修数列的代码,仅供区间操作用的 splay 参考。

```
const int INF = 1000000000;
const int Maxspace = 500000;
struct SplayNode{
   int ls, rs, zs, ms;
   SplayNode() {
      ms = 0;
      ls = rs = zs = -INF;
}
```

CHAPTER 3. 数据结构

```
8
         SplayNode(int d) {
9
             ms = zs = 1s = rs = d;
10
11
         SplayNode operator +(const SplayNode &p)const {
             SplayNode ret;
13
             ret.ls = max(ls, ms + p.ls);
14
15
             ret.rs = max(rs + p.ms, p.rs);
16
             ret.zs = max(rs + p.ls, max(zs, p.zs));
             ret.ms = ms + p.ms;
17
             return ret;
18
         }
19
     }t[MAXN], d[MAXN];
20
     int n, m, rt, top, a[MAXN], f[MAXN], c[MAXN][2], g[MAXN], h[MAXN], z[MAXN];
21
     bool r[MAXN], b[MAXN];
22
23
     void makesame(int x, int s) {
         if (!x) return;
24
         b[x] = true;
25
         d[x] = SplayNode(g[x] = s);
26
27
         t[x].zs = t[x].ms = g[x] * h[x];
         t[x].ls = t[x].rs = max(g[x], g[x] * h[x]);
28
29
     void makerev(int x) {
30
31
         if (!x) return;
32
         r[x] = 1;
         swap(c[x][0], c[x][1]);
33
         swap(t[x].ls, t[x].rs);
34
35
     void pushdown(int x) {
36
         if (!x) return;
37
         if (r[x]) {
38
             makerev(c[x][0]);
39
             makerev(c[x][1]);
40
             r[x]=0;
41
         }
42
         if (b[x]) {
43
             makesame(c[x][0],g[x]);
44
             makesame(c[x][1],g[x]);
45
             b[x]=g[x]=0;
46
         }
47
     }
48
     void updata(int x) {
49
         if (!x) return;
50
         h[x]=h[c[x][0]]+h[c[x][1]]+1;
51
         t[x]=t[c[x][0]]+d[x]+t[c[x][1]];
52
     }
53
     void rotate(int x,int k) {
54
         pushdown(x);pushdown(c[x][k]);
55
         int y = c[x][k]; c[x][k] = c[y][k^1]; c[y][k^1] = x;
56
57
         if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y; else rt = y;
         f[y] = f[x]; f[x] = y; f[c[x][k]] = x;
58
59
         updata(x); updata(y);
60
     void splay(int x, int s) {
61
         while (f[x] != s) {
62
             if (f[f[x]]!=s) {
63
                  pushdown(f[f[x]]);
64
                  rotate(f[f[x]], (c[f[f[x]]][1] == f[x]) \hat{r} r[f[f[x]]]);
65
             }
66
             pushdown(f[x]);
67
             rotate(f[x], (c[f[x]][1]==x) ^ r[f[x]]);
68
         }
69
     }
70
```

3.2. SPLAY 区间操作版 15

```
void build(int &x,int l,int r) {
71
          if (1 > r) \{x = 0; return;\}
72
          x = z[top--];
73
          if (1 < r) {
74
              build(c[x][0],1,(1+r>>1)-1);
75
              build(c[x][1],(1+r>>1)+1,r);
76
          }
77
          f[c[x][0]] = f[c[x][1]] = x;
78
79
          d[x] = SplayNode(a[1+r>>1]);
          updata(x);
80
      }
81
      void init() {
82
          d[0] = SplayNode();
83
          f[rt=2] = -1;
84
          f[1] = 2; c[2][0] = 1;
85
86
          int x;
          build(x,1,n);
87
          c[1][1] = x; f[x] = 1;
88
          splay(x, -1);
89
90
      int find(int z) {
91
          int x = rt; pushdown(x);
92
          while (z != h[c[x][0]] + 1) {
93
               if (z > h[c[x][0]] + 1) {
94
95
                   z = h[c[x][0]] + 1;
                   x = c[x][1];
96
              }
97
              else x = c[x][0];
98
99
              pushdown(x);
          }
100
101
          return x;
102
      void getrange(int &x,int &y) {
103
          y = x + y - 1;
104
105
          x = find(x);
106
          y = find(y + 2);
          splay(y, -1);
107
          splay(x, y);
108
      }
109
110
      void recycle(int x) {
          if (!x) return;
111
          recycle(c[x][0]);
112
113
          recycle(c[x][1]);
          z[++top]=x;
114
          t[x] = d[x] = SplayNode();
115
          r[x] = b[x] = g[x] = f[x] = h[x] = 0;
116
          c[x][0] = c[x][1]=0;
117
118
      int main() {
119
          scanf("%d%d",&n,&m);
120
          for (int i = 1; i <= n; i++) scanf("%d",a+i);
121
          for (int i = Maxspace; i>=3; i--) z[++top] = i;
122
          init();
123
          for (int i = 1; i <= m; i++) {
124
125
               char op[10];
126
              int x, y, tmp;
              scanf("%s", op);
127
               if (!strcmp(op, "INSERT")) {
128
                   scanf("%d%d", &x, &y);
129
                   n += y;
130
                   if (!y) continue;
131
                   for (int i = 1; i <= y; i++) scanf("%d",a+i);
132
133
                   build(tmp, 1, y);
```

CHAPTER 3. 数据结构

```
x = find(x + 1); pushdown(x);
134
                   if (!c[x][1]) \{c[x][1] = tmp; f[tmp] = x;\}
135
                   else{
136
                       x = c[x][1]; pushdown(x);
137
                       while (c[x][0]) {
138
                            x = c[x][0];
139
                            pushdown(x);
140
                       }
141
142
                       c[x][0] = tmp; f[tmp] = x;
143
                   splay(tmp, -1);
144
              }
145
               else if (!strcmp(op, "DELETE")) {
146
                   scanf("%d%d", &x, &y); n -= y;
147
                   if (!y) continue;
148
149
                   getrange(x, y);
                   int k = (c[y][0] == x);
150
                   recycle(c[x][k]);
151
                   f[c[x][k]] = 0;
152
153
                   c[x][k] = 0;
                   splay(x, -1);
154
              }
155
              else if (!strcmp(op, "REVERSE")) {
156
                   scanf("%d%d", &x, &y);
157
158
                   if (!y) continue;
                   getrange(x, y);
159
                   int k = (c[y][0]==x);
160
                   makerev(c[x][k]);
161
                   splay(c[x][k], -1);
162
163
              else if (!strcmp(op, "GET-SUM")) {
164
                   scanf("%d%d", &x, &y);
165
                   if (!y) {
166
                       printf("0\n");
167
168
                       continue;
169
                   }
                   getrange(x,y);
170
                   int k = (c[y][0] == x);
171
                   printf("%d\n", t[c[x][k]].ms);
172
173
                   splay(c[x][k], -1);
174
              else if (!strcmp(op, "MAX-SUM")) {
175
176
                   x = 1; y = n;
                   getrange(x, y);
177
                   int k = (c[y][0] == x);
178
                   printf("%d\n", t[c[x][k]].zs);
179
180
                   splay(c[x][k], -1);
181
              else if (!strcmp(op, "MAKE-SAME")) {
182
                   scanf("%d%d%d", &x, &y, &tmp);
183
                   if (!y) continue;
184
                   getrange(x, y);
185
                   int k = (c[y][0] == x);
186
                   makesame(c[x][k], tmp);
187
188
                   splay(c[x][k], -1);
              }
189
          }
190
          return 0;
191
      }
192
```

3.3. 坚固的 TREAP 17

## 3.3 坚固的 Treap

使用条件及注意事项: 题目来源 UVA 12358

```
const int INF = 100000000;
     const int Maxspace = 500000;
     struct SplayNode{
3
         int ls, rs, zs, ms;
4
         SplayNode() {
5
             ms = 0;
6
             ls = rs = zs = -INF;
8
         SplayNode(int d) {
             ms = zs = ls = rs = d;
10
11
12
         SplayNode operator +(const SplayNode &p)const {
13
             SplayNode ret;
             ret.ls = max(ls, ms + p.ls);
14
             ret.rs = max(rs + p.ms, p.rs);
15
             ret.zs = max(rs + p.ls, max(zs, p.zs));
16
             ret.ms = ms + p.ms;
17
             return ret;
18
         }
19
     }t[MAXN], d[MAXN];
20
     int n, m, rt, top, a[MAXN], f[MAXN], c[MAXN][2], g[MAXN], h[MAXN], z[MAXN];
21
     bool r[MAXN], b[MAXN];
22
23
     void makesame(int x, int s) {
         if (!x) return;
24
         b[x] = true;
25
         d[x] = SplayNode(g[x] = s);
26
         t[x].zs = t[x].ms = g[x] * h[x];
27
28
         t[x].ls = t[x].rs = max(g[x], g[x] * h[x]);
29
     void makerev(int x) {
30
31
         if (!x) return;
32
         r[x] = 1;
         swap(c[x][0], c[x][1]);
33
         swap(t[x].ls, t[x].rs);
34
35
     void pushdown(int x) {
36
         if (!x) return;
37
         if (r[x]) {
38
             makerev(c[x][0]);
39
             makerev(c[x][1]);
40
             r[x]=0;
41
         }
42
         if (b[x]) {
43
             makesame(c[x][0],g[x]);
44
             makesame(c[x][1],g[x]);
45
             b[x]=g[x]=0;
46
         }
47
48
     void updata(int x) {
49
         if (!x) return;
50
         h[x]=h[c[x][0]]+h[c[x][1]]+1;
51
         t[x]=t[c[x][0]]+d[x]+t[c[x][1]];
52
53
     void rotate(int x,int k) {
54
55
         pushdown(x);pushdown(c[x][k]);
         int y = c[x][k]; c[x][k] = c[y][k^1]; c[y][k^1] = x;
56
         if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y; else rt = y;
57
         f[y] = f[x]; f[x] = y; f[c[x][k]] = x;
58
         updata(x); updata(y);
59
```

18 CHAPTER 3. 数据结构

```
}
60
      void splay(int x, int s) {
61
          while (f[x] != s) {
62
              if (f[f[x]]!=s) {
63
                   pushdown(f[f[x]]);
64
                   rotate(f[f[x]], (c[f[f[x]]][1] == f[x]) \hat{r} r[f[f[x]]]);
65
              }
66
67
              pushdown(f[x]);
68
              rotate(f[x], (c[f[x]][1]==x) \hat{r} r[f[x]]);
          }
69
      }
70
      void build(int &x,int l,int r) {
71
          if (1 > r) {x = 0; return;}
72
73
          x = z[top--];
          if (1 < r) {
74
              build(c[x][0],1,(1+r>>1)-1);
75
              build(c[x][1],(1+r>>1)+1,r);
76
77
          f[c[x][0]] = f[c[x][1]] = x;
78
79
          d[x] = SplayNode(a[1+r>>1]);
          updata(x);
80
      }
81
      void init() {
82
          d[0] = SplayNode();
          f[rt=2] = -1;
84
          f[1] = 2; c[2][0] = 1;
85
86
          int x;
          build(x,1,n);
87
          c[1][1] = x; f[x] = 1;
88
          splay(x, -1);
89
      }
90
      int find(int z) {
91
          int x = rt; pushdown(x);
92
          while (z != h[c[x][0]] + 1) {
93
94
              if (z > h[c[x][0]] + 1) {
                   z = h[c[x][0]] + 1;
95
                   x = c[x][1];
96
              }
97
98
               else x = c[x][0];
99
              pushdown(x);
          }
100
101
          return x;
102
      void getrange(int &x,int &y) {
103
          y = x + y - 1;
104
          x = find(x);
105
          y = find(y + 2);
106
107
          splay(y, -1);
          splay(x, y);
108
109
      void recycle(int x) {
110
          if (!x) return;
111
          recycle(c[x][0]);
112
          recycle(c[x][1]);
113
114
          z[++top]=x;
          t[x] = d[x] = SplayNode();
115
          r[x] = b[x] = g[x] = f[x] = h[x] = 0;
116
          c[x][0] = c[x][1]=0;
117
      }
118
      int main() {
119
          scanf("%d%d",&n,&m);
120
          for (int i = 1; i <= n; i++) scanf("%d",a+i);
121
122
          for (int i = Maxspace; i>=3; i--) z[++top] = i;
```

3.3. 坚固的 TREAP 19

```
init();
123
          for (int i = 1; i <= m; i++) {
124
               char op[10];
125
126
              int x, y, tmp;
127
               scanf("%s", op);
               if (!strcmp(op, "INSERT")) {
128
                   scanf("%d%d", &x, &y);
129
                   n += y;
130
131
                   if (!y) continue;
                   for (int i = 1; i <= y; i++) scanf("%d",a+i);
132
                   build(tmp, 1, y);
133
                   x = find(x + 1); pushdown(x);
134
                   if (!c[x][1]) \{c[x][1] = tmp; f[tmp] = x;\}
135
136
                   else{
                       x = c[x][1]; pushdown(x);
137
                       while (c[x][0]) {
138
                            x = c[x][0];
139
                            pushdown(x);
140
141
142
                       c[x][0] = tmp; f[tmp] = x;
                   }
143
                   splay(tmp, -1);
144
              }
145
              else if (!strcmp(op, "DELETE")) {
146
147
                   scanf("%d%d", &x, &y); n -= y;
                   if (!y) continue;
148
                   getrange(x, y);
149
                   int k = (c[y][0] == x);
150
                   recycle(c[x][k]);
151
                   f[c[x][k]] = 0;
152
                   c[x][k] = 0;
153
                   splay(x, -1);
154
155
              else if (!strcmp(op, "REVERSE")) {
156
                   scanf("%d%d", &x, &y);
157
                   if (!y) continue;
158
                   getrange(x, y);
159
                   int k = (c[y][0]==x);
160
                   makerev(c[x][k]);
161
162
                   splay(c[x][k], -1);
163
              else if (!strcmp(op, "GET-SUM")) {
164
165
                   scanf("%d%d", &x, &y);
                   if (!y) {
166
                       printf("0\n");
167
                       continue;
168
                   }
169
                   getrange(x,y);
170
                   int k = (c[y][0] == x);
171
                   printf("%d\n", t[c[x][k]].ms);
172
                   splay(c[x][k], -1);
173
              }
174
              else if (!strcmp(op, "MAX-SUM")) {
175
                   x = 1; y = n;
176
                   getrange(x, y);
177
178
                   int k = (c[y][0] == x);
                   printf("%d\n", t[c[x][k]].zs);
179
                   splay(c[x][k], -1);
180
              }
181
              else if (!strcmp(op, "MAKE-SAME")) {
182
                   scanf("%d%d%d", &x, &y, &tmp);
183
184
                   if (!y) continue;
185
                   getrange(x, y);
```

CHAPTER 3. 数据结构

```
int k = (c[y][0] == x);
makesame(c[x][k], tmp);
splay(c[x][k], -1);

splay(c[x][k], -1);

return 0;

return 0;
```

### 3.4 k-d 树

使用条件及注意事项: 这是求 k 远点的代码,要求 k 近点的话把堆的比较函数改一改,把朝左儿子或者是右儿子的方向改一改。

```
struct Heapnode{
1
2
         long long d;
3
         int pos;
         bool operator <(const Heapnode &p)const {
4
             return d > p.d || (d == p.d && pos < p.pos);
5
         }
6
     };
7
8
     struct MsgNode{
9
         int xmin, xmax, ymin, ymax;
10
         MsgNode() {}
11
         MsgNode(const Point &a) : xmin(a.x), xmax(a.x), ymin(a.y), ymax(a.y) {}
12
         long long dist(const Point &a) {
13
             int dx = std::max(std::abs(a.x - xmin), std::abs(a.x - xmax));
14
             int dy = std::max(std::abs(a.y - ymin), std::abs(a.y - ymax));
15
             return (long long)dx * dx + (long long)dy * dy;
16
         7
17
18
         MsgNode operator +(const MsgNode &rhs)const {
             MsgNode ret;
19
             ret.xmin = std::min(xmin, rhs.xmin);
20
21
             ret.xmax = std::max(xmax, rhs.xmax);
22
             ret.ymin = std::min(ymin, rhs.ymin);
             ret.ymax = std::max(ymax, rhs.ymax);
23
             return ret;
24
         }
25
     };
26
27
     struct TNode{
28
         int 1, r;
29
         Point p;
30
         MsgNode d;
31
     }tree[MAXN];
32
33
     void buildtree(int &rt, int 1, int r, int pivot) {
34
         if (1 > r) return;
35
         rt = ++size;
36
         int mid = 1 + r \gg 1;
37
         if (pivot == 1) std::nth_element(p + 1, p + mid, p + r + 1, cmpx);
38
         if (pivot == 0) std::nth_element(p + 1, p + mid, p + r + 1, cmpy);
39
         tree[rt].d = MsgNode(tree[rt].p = p[mid]);
40
         buildtree(tree[rt].1, 1, mid - 1, pivot
41
         buildtree(tree[rt].r, mid + 1, r, pivot ^ 1);
42
         if (tree[rt].1) tree[rt].d = tree[rt].d + tree[tree[rt].1].d;
43
         if (tree[rt].r) tree[rt].d = tree[rt].d + tree[tree[rt].r].d;
44
45
     }
46
     void query(int rt, const Point &a, int k, int pivot) {
47
         Heapnode now = (Heapnode){dist(a, tree[rt].p), tree[rt].p.pos};
48
         if (heap.size() < k) heap.push(now);</pre>
49
```

3.5. 树链剖分 21

```
else if (now < heap.top()) {heap.pop(); heap.push(now);}</pre>
50
         int lson = tree[rt].1, rson = tree[rt].r;
51
         if (pivot == 1 && cmpx(a, tree[rt].p)) std::swap(lson, rson);
52
         if (pivot == 0 && cmpy(a, tree[rt].p)) std::swap(lson, rson);
53
         if (lson && (heap.size() < k | | tree[lson].d.dist(a) >= heap.top().d)) query(lson, a, k,
54
         pivot ^ 1);
         if (rson && (heap.size() < k \mid \mid tree[rson].d.dist(a) >= heap.top().d)) query(rson, a, k,
55
         pivot ^ 1);
56
57
     int main() {
58
         for (int i = 1; i <= q; i++) {
59
60
             int k;
             Point now;
61
             now.read();
62
             scanf("%d", &k);
63
             while (!heap.empty()) heap.pop();
64
             query(rt, now, k, 1);
65
             printf("%d\n", heap.top().pos);
66
67
         }
68
         return 0;
     }
69
```

## 3.5 树链剖分

### 3.5.1 点操作版本

使用条件及注意事项:树上最大(非空)子段和,注意一条路径询问的时候信息统计的顺序。

```
struct Node{
1
2
         int asum, lsum, rsum, zsum;
         Node() {
3
             asum = 0;
4
5
             lsum = -INF;
6
             rsum = -INF;
             zsum = -INF;
8
         Node(int d) : asum(d), lsum(d), rsum(d), zsum(d) {}
9
         Node operator +(const Node &rhs)const {
10
             Node ret;
11
12
             ret.asum = asum + rhs.asum;
             ret.lsum = std::max(lsum, asum + rhs.lsum);
13
             ret.rsum = std::max(rsum + rhs.asum, rhs.rsum);
14
             ret.zsum = std::max(zsum, rhs.zsum);
15
             ret.zsum = std::max(ret.zsum, rsum + rhs.lsum);
16
17
             return ret;
         }
18
     tree[MAXN * 6];
19
20
     int n, q, cnt, tot, h[MAXN], d[MAXN], t[MAXN], f[MAXN], s[MAXN], z[MAXN], w[MAXN], o[MAXN],
21
     \rightarrow a[MAXN];
     std::pair<bool, int> flag[MAXN * 6];
22
23
     void addedge(int x, int y) {
24
         cnt++; e[cnt] = (Edge){y, <math>h[x]}; h[x] = cnt;
25
         cnt++; e[cnt] = (Edge)\{x, h[y]\}; h[y] = cnt;
26
     }
^{27}
28
     void makesame(int n, int l, int r, int d) {
29
         flag[n] = std::make_pair(true, d);
30
         tree[n].asum = d * (r - 1 + 1);
31
         if (d > 0) {
32
```

CHAPTER 3. 数据结构

```
33
             tree[n].lsum = d * (r - 1 + 1);
             tree[n].rsum = d * (r - 1 + 1);
34
             tree[n].zsum = d * (r - 1 + 1);
35
         }
36
         else{
37
             tree[n].lsum = d;
38
             tree[n].rsum = d;
39
40
             tree[n].zsum = d;
41
         }
     }
42
43
     void pushdown(int n, int l, int r) {
44
         if (flag[n].first) {
45
             makesame(n \ll 1, 1, 1 + r \gg 1, flag[n].second);
46
             makesame(n << 1 ^ 1, (1 + r >> 1) + 1, r, flag[n].second);
47
             flag[n] = std::make_pair(false, 0);
48
49
     }
50
51
     void modify(int n, int l, int r, int x, int y, int d) {
52
         if (x \le 1 \&\& r \le y) {
53
             makesame(n, 1, r, d);
54
             return;
55
         }
56
57
         pushdown(n, l, r);
         if ((1 + r >> 1) < x) modify(n << 1 ^ 1, (1 + r >> 1) + 1, r, x, y, d);
58
         else if ((1 + r >> 1) + 1 > y) modify(n << 1, 1, 1 + r >> 1, x, y, d);
59
60
61
             modify(n << 1, 1, 1 + r >> 1, x, y, d);
             modify(n << 1 ^1, (1 + r >> 1) + 1, r, x, y, d);
62
63
         tree[n] = tree[n << 1] + tree[n << 1 ^ 1];
64
65
66
67
     Node query(int n, int l, int r, int x, int y) {
         if (x \le 1 \&\& r \le y) return tree[n];
68
         pushdown(n, 1, r);
69
         if ((1 + r >> 1) < x) return query(n << 1 ^ 1, (1 + r >> 1) + 1, r, x, y);
70
         else if ((1 + r >> 1) + 1 > y) return query(n << 1, 1, 1 + r >> 1, x, y);
71
72
         else{
             Node left = query(n << 1, 1, 1 + r >> 1, x, y);
73
             Node right = query(n << 1 ^ 1, (1 + r >> 1) + 1, r, x, y);
74
75
             return left + right;
         }
76
77
78
     void modify(int x, int y, int val) {
79
         int fx = t[x], fy = t[y];
80
         while (fx != fy) {
81
82
             if (d[fx] > d[fy]) {
                  modify(1, 1, n, w[fx], w[x], val);
83
                  x = f[fx]; fx = t[x];
84
             }
85
             else{
86
                  modify(1, 1, n, w[fy], w[y], val);
87
88
                  y = f[fy]; fy = t[y];
             }
89
         }
90
         if (d[x] < d[y]) modify(1, 1, n, w[x], w[y], val);
91
         else modify(1, 1, n, w[y], w[x], val);
92
     }
93
94
     Node query(int x, int y) {
```

3.5. 树链剖分 23

```
96
          int fx = t[x], fy = t[y];
          Node left = Node(), right = Node();
97
          while (fx != fy) {
98
              if (d[fx] > d[fy]) {
99
                   left = query(1, 1, n, w[fx], w[x]) + left;
100
                   x = f[fx]; fx = t[x];
101
              }
102
103
              else{
104
                   right = query(1, 1, n, w[fy], w[y]) + right;
                   y = f[fy]; fy = t[y];
105
              }
106
          }
107
          if (d[x] < d[y]) {
108
109
              right = query(1, 1, n, w[x], w[y]) + right;
          }
110
          else{
111
              left = query(1, 1, n, w[y], w[x]) + left;
112
113
          std::swap(left.lsum, left.rsum);
114
115
          return left + right;
116
117
      void predfs(int x) {
118
          s[x] = 1; z[x] = 0;
119
120
          for (int i = h[x]; i; i = e[i].next) {
              if (e[i].node == f[x]) continue;
121
              f[e[i].node] = x;
122
123
              d[e[i].node] = d[x] + 1;
              predfs(e[i].node);
124
              s[x] += s[e[i].node];
125
              if (s[z[x]] < s[e[i].node]) z[x] = e[i].node;
126
127
      }
128
129
      void getanc(int x, int anc) {
130
          t[x] = anc; w[x] = ++tot; o[tot] = x;
131
          if (z[x]) getanc(z[x], anc);
132
          for (int i = h[x]; i; i = e[i].next) {
133
              if (e[i].node == f[x] || e[i].node == z[x]) continue;
134
135
              getanc(e[i].node, e[i].node);
          }
136
      }
137
138
      void buildtree(int n, int l, int r) {
139
          if (1 == r) {
140
              tree[n] = Node(a[o[1]]);
141
              return;
142
143
          buildtree(n << 1, 1, 1 + r >> 1);
144
          buildtree(n << 1 ^ 1, (1 + r >> 1) + 1, r);
145
          tree[n] = tree[n << 1] + tree[n << 1 ^ 1];
146
147
148
      int main() {
149
          scanf("%d", &n);
150
          for (int i = 1; i \le n; i++) scanf("%d", a + i);
151
          for (int i = 1; i < n; i++) {
152
              int x, y; scanf("%d%d", &x, &y);
153
              addedge(x, y);
154
155
          predfs(1);
156
157
          getanc(1, 1);
158
          buildtree(1, 1, n);
```

24 CHAPTER 3. 数据结构

```
scanf("%d", &q);
159
          for (int i = 1; i <= q; i++) {
160
               int op, x, y, c;
161
               scanf("%d", &op);
162
               if (op == 1) {
163
                   scanf("%d%d", &x, &y);
164
                   Node ret = query(x, y);
165
                   printf("%d\n", std::max(0, ret.zsum));
166
167
               }
               else{
168
                   scanf("%d%d%d", &x, &y, &c);
169
170
                   modify(x, y, c);
               }
171
          }
172
          return 0;
173
      }
174
```

### 3.5.2 链操作版本

```
void modify(int x, int y) {
1
         int fx = t[x], fy = t[y];
2
         while (fx != fy) {
3
              if (d[fx] > d[fy]) {
4
                  modify(1, 1, n, w[fx], w[x]);
5
                  x = f[fx]; fx = t[x];
6
             }
7
             else{
                  modify(1, 1, n, w[fy], w[y]);
9
                  y = f[fy]; fy = t[y];
10
             }
11
         }
12
         if (x != y) {
13
              if (d[x] < d[y]) modify(1, 1, n, w[z[x]], w[y]);
14
              else modify(1, 1, n, w[z[y]], w[x]);
15
16
         }
     }
17
```

### 3.6 Link-Cut-Tree

```
1
     struct MsgNode{
         int leftColor, rightColor, answer;
2
         MsgNode() {
3
             leftColor = -1;
4
             rightColor = -1;
             answer = 0;
6
         }
         MsgNode(int c) {
8
9
             leftColor = rightColor = c;
             answer = 1;
10
11
         MsgNode operator +(const MsgNode &p)const {
12
             if (answer == 0) return p;
13
             if (p.answer == 0) return *this;
14
             MsgNode ret;
15
16
             ret.leftColor = leftColor;
17
             ret.rightColor = p.rightColor;
             ret.answer = answer + p.answer - (rightColor == p.leftColor);
18
             return ret;
19
20
     }d[MAXN], g[MAXN];
21
```

3.6. LINK-CUT-TREE

```
int n, m, c[MAXN][2], f[MAXN], p[MAXN], s[MAXN], flag[MAXN];
22
23
     bool r[MAXN];
     void init(int x, int value) {
24
         d[x] = g[x] = MsgNode(value);
25
         c[x][0] = c[x][1] = 0;
26
27
         f[x] = p[x] = flag[x] = -1;
         s[x] = 1;
28
     }
29
30
     void update(int x) {
         s[x] = s[c[x][0]] + s[c[x][1]] + 1;
31
         g[x] = MsgNode();
32
         if (c[x][0 \hat{r}[x]]) g[x] = g[x] + g[c[x][0 \hat{r}[x]]];
33
         g[x] = g[x] + d[x];
34
         if (c[x][1 \hat{r}[x]]) g[x] = g[x] + g[c[x][1 \hat{r}[x]]];
35
36
     void makesame(int x, int c) {
37
38
         flag[x] = c;
         d[x] = MsgNode(c);
39
         g[x] = MsgNode(c);
40
41
     void pushdown(int x) {
42
         if (r[x]) {
43
             std::swap(c[x][0], c[x][1]);
44
45
             r[c[x][0]] = 1;
             r[c[x][1]] = 1;
46
             std::swap(g[c[x][0]].leftColor, g[c[x][0]].rightColor);
47
             std::swap(g[c[x][1]].leftColor, g[c[x][1]].rightColor);
48
             r[x] = false;
49
50
         if (flag[x] != -1) {
51
              if (c[x][0]) makesame(c[x][0], flag[x]);
52
              if (c[x][1]) makesame(c[x][1], flag[x]);
53
             flag[x] = -1;
54
         }
55
56
     }
     void rotate(int x, int k) {
57
         pushdown(x); pushdown(c[x][k]);
58
         int y = c[x][k]; c[x][k] = c[y][k ^ 1]; c[y][k ^ 1] = x;
59
         if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y;
60
61
         f[y] = f[x]; f[x] = y; f[c[x][k]] = x; std::swap(p[x], p[y]);
         update(x); update(y);
62
63
     void splay(int x, int s = -1) {
64
         pushdown(x);
65
         while (f[x] != s) {
66
              if (f[f[x]] = s) rotate(f[f[x]], (c[f[f[x]]][1] = f[x]) r[f[f[x]]]);
67
             rotate(f[x], (c[f[x]][1] == x) \hat{r}[f[x]]);
68
69
         update(x);
70
71
     void access(int x) {
72
73
         int y = 0;
         while (x != -1) {
74
              splay(x); pushdown(x);
75
             f[c[x][1]] = -1; p[c[x][1]] = x;
76
             c[x][1] = y; f[y] = x; p[y] = -1;
77
             update(x); x = p[y = x];
78
         }
79
80
     void setroot(int x) {
81
         access(x);
82
83
         splay(x);
84
         r[x] = 1;
```

26 CHAPTER 3. 数据结构

## Chapter 4

# 图论

## 4.1 强连通分量

```
int stamp, comps, top;
     int dfn[N], low[N], comp[N], stack[N];
2
3
     void tarjan(int x) {
         dfn[x] = low[x] = ++stamp;
5
         stack[top++] = x;
6
         for (int i = 0; i < (int)edge[x].size(); ++i) {</pre>
7
              int y = edge[x][i];
              if (!dfn[y]) {
9
                  tarjan(y);
10
                  low[x] = std::min(low[x], low[y]);
11
              } else if (!comp[y]) {
                  low[x] = std::min(low[x], dfn[y]);
13
14
15
         if (low[x] == dfn[x]) {
16
              comps++;
17
              do {
18
                  int y = stack[--top];
19
20
                  comp[y] = comps;
              } while (stack[top] != x);
21
         }
22
     }
23
^{24}
     void solve() {
25
         stamp = comps = top = 0;
26
27
         std::fill(dfn, dfn + n, 0);
         std::fill(comp, comp + n, 0);
28
         for (int i = 0; i < n; ++i) {
29
              if (!dfn[i]) {
30
31
                  tarjan(i);
32
         }
33
     }
34
```

## 4.2 点双连通分量

### 4.2.1 坚固的点双连通分量

```
int n, m, x, y, ans1, ans2, tot1, tot2, flag, size, ind2, dfn[N], low[N], block[M], vis[N];
vector<int> a[N];
pair<int, int> stack[M];
void tarjan(int x, int p) {
    dfn[x] = low[x] = ++ind2;
```

28 CHAPTER 4. 图论

```
for (int i = 0; i < a[x].size(); ++i)</pre>
6
              if (dfn[x] > dfn[a[x][i]] \&\& a[x][i] != p){
                  stack[++size] = make_pair(x, a[x][i]);
8
                  if (i == a[x].size() - 1 || a[x][i] != a[x][i + 1])
9
                      if (!dfn[a[x][i]]){
10
                           tarjan(a[x][i], x);
11
                           low[x] = min(low[x], low[a[x][i]]);
12
13
                           if (low[a[x][i]] >= dfn[x]){
                               tot1 = tot2 = 0;
                               ++flag;
15
                               for (; ; ){
16
                                   if (block[stack[size].first] != flag) {
17
18
                                        block[stack[size].first] = flag;
19
20
                                   if (block[stack[size].second] != flag) {
21
                                        ++tot1;
22
                                        block[stack[size].second] = flag;
23
24
25
                                   if (stack[size].first == x && stack[size].second == a[x][i])
26
                                   ++tot2;
27
                                   --size;
28
                               }
29
30
                               for (; stack[size].first == x && stack[size].second == a[x][i]; --size)
                                   ++tot2;
31
                               if (tot2 < tot1)
32
33
                                   ans1 += tot2;
                               if (tot2 > tot1)
34
                                   ans2 += tot2;
35
                           }
36
                      }
37
                      else
38
                           low[x] = min(low[x], dfn[a[x][i]]);
39
              }
40
41
     int main(){
42
         for (; ; ){
43
              scanf("%d%d", &n, &m);
44
45
              if (n == 0 && m == 0) return 0;
              for (int i = 1; i <= n; ++i) {
46
                  a[i].clear();
47
48
                  dfn[i] = 0;
              }
49
              for (int i = 1; i <= m; ++i){
50
                  scanf("%d%d",&x, &y);
51
52
                  ++x, ++y;
                  a[x].push_back(y);
53
                  a[y].push_back(x);
54
              }
55
              for (int i = 1; i <= n; ++i)
56
                  sort(a[i].begin(), a[i].end());
57
              ans1 = ans2 = ind2 = 0;
58
              for (int i = 1; i <= n; ++i)</pre>
59
                  if (!dfn[i]) {
60
61
                      size = 0;
                      tarjan(i, 0);
62
63
              printf("%d %d\n", ans1, ans2);
64
65
         return 0;
66
     }
67
```

4.3. 2-SAT 问题 29

### 4.2.2 朴素的点双连通分量

```
void tarjan(int x){
1
2
         dfn[x] = low[x] = ++ind2;
         v[x] = 1;
3
         for (int i = nt[x]; pt[i]; i = nt[i])
4
              if (!dfn[pt[i]]){
                  tarjan(pt[i]);
6
                  low[x] = min(low[x], low[pt[i]]);
                  if (dfn[x] <= low[pt[i]])</pre>
8
                      ++v[x];
9
              }
10
              else
11
                  low[x] = min(low[x], dfn[pt[i]]);
12
13
     int main(){
14
         for (; ; ){
15
              scanf("%d%d", &n, &m);
16
17
              if (n == 0 \&\& m == 0)
18
                  return 0;
              for (int i = 1; i <= ind; ++i)
19
                  nt[i] = pt[i] = 0;
20
21
              ind = n;
22
              for (int i = 1; i <= ind; ++i)
                  last[i] = i;
23
              for (int i = 1; i <= m; ++i){
24
                  scanf("%d%d", &x, &y);
25
                  ++x, ++y;
26
                  edge(x, y), edge(y, x);
27
28
              memset(dfn, 0, sizeof(dfn));
29
              memset(v, 0, sizeof(v));
30
              ans = num = ind2 = 0;
31
32
              for (int i = 1; i <= n; ++i)
                  if (!dfn[i]){
33
                      root = i;
34
                      size = 0;
35
36
                      ++num;
37
                      tarjan(i);
                       --v[root];
38
                  }
39
              for (int i = 1; i <= n; ++i)
40
                  if (v[i] + num - 1 > ans)
41
                      ans = v[i] + num - 1;
42
              printf("%d\n",ans);
43
         }
44
         return 0;
45
46
```

## 4.3 2-SAT 问题

```
int stamp, comps, top;
1
     int dfn[N], low[N], comp[N], stack[N];
2
3
     void add(int x, int a, int y, int b) {
4
         edge[x \ll 1 \mid a].push_back(y \ll 1 \mid b);
5
6
7
     void tarjan(int x) {
8
         dfn[x] = low[x] = ++stamp;
9
         stack[top++] = x;
10
```

30 CHAPTER 4. 图论

```
for (int i = 0; i < (int)edge[x].size(); ++i) {</pre>
11
              int y = edge[x][i];
12
              if (!dfn[y]) {
13
                  tarjan(y);
14
                  low[x] = std::min(low[x], low[y]);
15
              } else if (!comp[y]) {
16
                  low[x] = std::min(low[x], dfn[y]);
17
18
19
         }
         if (low[x] == dfn[x]) {
20
              comps++;
21
22
              do {
                  int y = stack[--top];
23
24
                  comp[y] = comps;
              } while (stack[top] != x);
25
         }
^{26}
27
28
     bool solve() {
29
30
         int counter = n + n + 1;
         stamp = top = comps = 0;
31
         std::fill(dfn, dfn + counter, 0);
32
         std::fill(comp, comp + counter, 0);
33
         for (int i = 0; i < counter; ++i) {</pre>
34
35
              if (!dfn[i]) {
                  tarjan(i);
36
              }
37
38
         for (int i = 0; i < n; ++i) {
39
              if (comp[i << 1] == comp[i << 1 | 1]) {
40
                  return false;
41
42
              answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
43
44
45
         return true;
46
     }
```

## 4.4 二分图最大匹配

### 4.4.1 Hungary 算法

时间复杂度:  $\mathcal{O}(V \cdot E)$ 

```
int n, m, stamp;
     int match[N], visit[N];
3
     bool dfs(int x) {
4
         for (int i = 0; i < (int)edge[x].size(); ++i) {</pre>
5
              int y = edge[x][i];
6
              if (visit[y] != stamp) {
                  visit[y] = stamp;
8
                   if (match[y] == -1 \mid \mid dfs(match[y])) {
9
                       match[y] = x;
10
                       return true;
11
12
              }
13
14
15
         return false;
     }
16
17
     int solve() {
18
         std::fill(match, match + m, -1);
19
```

4.4. 二分图最大匹配 31

### 4.4.2 Hopcroft Karp 算法

时间复杂度:  $\mathcal{O}(\sqrt{V} \cdot E)$ 

```
int matchx[N], matchy[N], level[N];
2
     bool dfs(int x) {
3
          for (int i = 0; i < (int)edge[x].size(); ++i) {</pre>
4
5
              int y = edge[x][i];
              int w = matchy[y];
6
              if (w == -1 \mid \mid level[x] + 1 == level[w] && dfs(w)) {
7
8
                  matchx[x] = y;
                  matchy[y] = x;
9
                  return true;
10
              }
11
12
          level[x] = -1;
13
          return false;
14
     }
15
16
     int solve() {
17
         std::fill(matchx, matchx + n, -1);
18
          std::fill(matchy, matchy + m, -1);
19
20
          for (int answer = 0; ; ) {
              std::vector<int> queue;
21
              for (int i = 0; i < n; ++i) {
22
                  if (matchx[i] == -1) {
23
24
                       level[i] = 0;
                       queue.push_back(i);
25
                  } else {
26
                       level[i] = -1;
27
28
29
              for (int head = 0; head < (int)queue.size(); ++head) {</pre>
30
31
                   int x = queue[head];
                   for (int i = 0; i < (int)edge[x].size(); ++i) {</pre>
32
                       int y = edge[x][i];
33
                       int w = matchy[y];
34
                       if (w != -1 \&\& level[w] < 0) {
35
                           level[w] = level[x] + 1;
36
                           queue.push_back(w);
37
                       }
38
                  }
39
              }
40
              int delta = 0;
41
              for (int i = 0; i < n; ++i) {</pre>
42
                   if (matchx[i] == -1 \&\& dfs(i)) {
43
                       delta++;
44
45
46
47
              if (delta == 0) {
                  return answer;
48
              } else {
49
50
                   answer += delta;
51
```

32 CHAPTER 4. 图论

```
52 }
53 }
```

## 4.5 二分图最大权匹配

时间复杂度:  $\mathcal{O}(V^4)$ 

```
int DFS(int x){
1
         visx[x] = 1;
2
         for (int y = 1; y \le ny; y ++){
3
              if (visy[y]) continue;
4
              int t = lx[x] + ly[y] - w[x][y];
              if (t == 0) {
6
                  visy[y] = 1;
                  if (link[y] == -1||DFS(link[y])){
                      link[y] = x;
10
                      return 1;
                  }
11
             }
12
             else slack[y] = min(slack[y],t);
13
         }
14
         return 0;
15
     }
16
17
     int KM(){
         int i,j;
18
         memset(link,-1,sizeof(link));
19
20
         memset(ly,0,sizeof(ly));
         for (i = 1; i <= nx; i++)
21
             for (j = 1, lx[i] = -inf; j \le ny; j++)
22
                  lx[i] = max(lx[i],w[i][j]);
23
         for (int x = 1; x \le nx; x++){
24
25
             for (i = 1; i <= ny; i++) slack[i] = inf;
             while (true) {
26
                  memset(visx, 0, sizeof(visx));
27
                  memset(visy, 0, sizeof(visy));
                  if (DFS(x)) break;
29
                  int d = inf;
30
                  for (i = 1; i <= ny;i++)
31
                      if (!visy[i] && d > slack[i]) d = slack[i];
32
                  for (i = 1; i \le nx; i++)
33
                      if (visx[i]) lx[i] -= d;
34
                  for (i = 1; i <= ny; i++)
35
                      if (visy[i]) ly[i] += d;
36
                      else slack[i] -= d;
37
             }
38
         }
39
         int res = 0;
40
         for (i = 1;i <= ny;i ++)
41
              if (link[i] > -1) res += w[link[i]][i];
42
43
         return res;
     }
44
```

## 4.6 最大流

#### 4.6.1 Dinic

使用方法以及注意事项: n 个点,m 条边,inf 为一个很大的值,源点 s,汇点 t,图中最大点的编号为 t。邻接表: p 数组记录节点,nxt 数组指向下一个位置,c 数组记录可增广量,h 数组记录表头 (初始全为-1)。时间复杂度:  $\mathcal{O}(V^2 \cdot E)$ 

4.6. 最大流 33

```
int bfs(){
1
         for (int i = 1; i \le t; i ++) d[i] = -1;
2
         int 1,r;
3
         q[1 = r = 0] = s, d[s] = 0;
4
         for (;1 <= r;1 ++)
5
             for (int k = h[q[1]]; k > -1; k = nxt[k])
6
                 if (d[p[k]] == -1 \&\& c[k] > 0) d[p[k]] = d[q[1]] + 1, q[++ r] = p[k];
7
         return d[t] > -1 ? 1 : 0;
8
     }
9
     int dfs(int u,int ext){
10
         if (u == t) return ext;
11
         int k = w[u], ret = 0;
12
                                                   //w 数组为当前弧
         for (; k > -1; k = nxt[k], w[u] = k){
13
             if (ext == 0) break;
14
             if (d[p[k]] == d[u] + 1 && c[k] > 0){
15
16
                 int flow = dfs(p[k], min(c[k], ext));
17
                 if (flow > 0){
                     c[k] = flow, c[k ^ 1] += flow;
18
                                                     //ret 累计增广量, ext 记录还可增广的量
                     ret += flow, ext -= flow;
19
                 }
20
             }
21
         }
22
         if (k == -1) d[u] = -1;
23
24
         return ret;
25
     void dinic() {
26
         while (bfs()) {
^{27}
28
             for (int i = 1; i <= t;i ++) w[i] = h[i];
29
             dfs(s, inf);
         }
30
     }
31
```

### 4.6.2 ISAP

时间复杂度:  $\mathcal{O}(V^2 \cdot E)$ 

```
int Maxflow_Isap(int s,int t,int n) {
1
         std::fill(pre + 1, pre + n + 1, 0);
2
         std::fill(d + 1, d + n + 1, 0);
3
4
         std::fill(gap + 1, gap + n + 1, 0);
         for (int i = 1; i <= n; i++) cur[i] = h[i];
         gap[0] = n;
6
         int u = pre[s] = s, v, maxflow = 0;
7
         while (d[s] < n) {
8
             v = n + 1;
9
             for (int i = cur[u]; i; i = e[i].next)
10
             if (e[i].flow && d[u] == d[e[i].node] + 1) {
11
                  v = e[i].node; cur[u]=i; break;
12
             }
13
             if (v <= n) {
14
                  pre[v] = u; u = v;
15
                  if (v == t) {
16
                      int dflow = INF, p = t; u = s;
17
                      while (p != s) {
18
                          p = pre[p];
19
                          dflow = std::min(dflow, e[cur[p]].flow);
20
21
                      }
                      maxflow += dflow; p = t;
22
                      while (p != s) {
23
^{24}
                          p = pre[p];
                          e[cur[p]].flow -= dflow;
25
```

34 CHAPTER 4. 图论

```
e[e[cur[p]].opp].flow += dflow;
26
27
                  }
28
              }
29
              else{
30
                  int mindist = n + 1;
31
                  for (int i = h[u]; i; i = e[i].next)
32
                      if (e[i].flow && mindist > d[e[i].node]) {
33
34
                           mindist = d[e[i].node]; cur[u] = i;
35
                  if (!--gap[d[u]]) return maxflow;
36
                  gap[d[u] = mindist + 1]++; u = pre[u];
37
              }
38
39
         return maxflow;
40
41
```

### 4.6.3 SAP

时间复杂度:  $\mathcal{O}(V^2 \cdot E)$ 

```
int Maxflow_Isap(int s,int t,int n) {
1
         std::fill(pre + 1, pre + n + 1, 0);
2
3
         std::fill(d + 1, d + n + 1, 0);
         std::fill(gap + 1, gap + n + 1, 0);
         for (int i = 1; i <= n; i++) cur[i] = h[i];
5
         gap[0] = n;
6
         int u = pre[s] = s, v, maxflow = 0;
         while (d[s] < n) {
             v = n + 1;
9
             for (int i = cur[u]; i; i = e[i].next)
10
             if (e[i].flow && d[u] == d[e[i].node] + 1) {
11
                  v = e[i].node; cur[u]=i; break;
12
13
             if (v \le n) {
14
                  pre[v] = u; u = v;
                  if (v == t) {
16
                      int dflow = INF, p = t; u = s;
17
                      while (p != s) {
18
19
                          p = pre[p];
                          dflow = std::min(dflow, e[cur[p]].flow);
20
21
                      maxflow += dflow; p = t;
22
23
                      while (p != s) {
24
                          p = pre[p];
                          e[cur[p]].flow -= dflow;
25
                          e[e[cur[p]].opp].flow += dflow;
26
                      }
27
                  }
28
             }
29
             else{
30
                  int mindist = n + 1;
31
                  for (int i = h[u]; i; i = e[i].next)
32
                      if (e[i].flow && mindist > d[e[i].node]) {
33
34
                          mindist = d[e[i].node]; cur[u] = i;
35
                  if (!--gap[d[u]]) return maxflow;
36
                  gap[d[u] = mindist + 1]++; u = pre[u];
37
             }
38
39
         return maxflow;
40
     }
41
```

4.7. 上下界网络流 35

### 4.7 上下界网络流

B(u,v) 表示边 (u,v) 流量的下界,C(u,v) 表示边 (u,v) 流量的上界,F(u,v) 表示边 (u,v) 的流量。设 G(u,v)=F(u,v)-B(u,v),显然有

$$0 \le G(u, v) \le C(u, v) - B(u, v)$$

### 4.7.1 无源汇的上下界可行流

建立超级源点  $S^*$  和超级汇点  $T^*$ ,对于原图每条边 (u,v) 在新网络中连如下三条边:  $S^* \to v$ ,容量为 B(u,v);  $u \to T^*$ ,容量为 B(u,v);  $u \to v$ ,容量为 C(u,v) - B(u,v)。最后求新网络的最大流,判断从超级源点  $S^*$  出发的边是否都满流即可,边 (u,v) 的最终解中的实际流量为 G(u,v) + B(u,v)。

### 4.7.2 有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为  $\infty$ ,下界为 0 的边。按照**无源汇的上下界可行流**一样做即可,流量即为  $T\to S$  边上的流量。

### 4.7.3 有源汇的上下界最大流

- 1. 在**有源汇的上下界可行流**中,从汇点 T 到源点 S 的边改为连一条上界为  $\infty$ ,下届为 x 的边。x 满足二分性质,找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。
- 2. 从汇点 T 到源点 S 连一条上界为 ∞,下界为 0 的边,变成无源汇的网络。按照**无源汇的上下界可行流**的方法,建立超级源点  $S^*$  和超级汇点  $T^*$ ,求一遍  $S^*$   $\to$   $T^*$  的最大流,再将从汇点 T 到源点 S 的这条边拆掉,求一次  $S \to T$  的最大流即可。

### 4.7.4 有源汇的上下界最小流

- 1. 在**有源汇的上下界可行流**中,从汇点 T 到源点 S 的边改为连一条上界为 x,下界为 0 的边。x 满足二分性质,找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。
- 2. 按照**无源汇的上下界可行流**的方法,建立超级源点  $S^*$  与超级汇点  $T^*$ ,求一遍  $S^* \to T^*$  的最大流,但是注意这一次不加上汇点 T 到源点 S 的这条边,即不使之改为无源汇的网络去求解。求完后,再加上那条汇点 T 到源点 S 上界  $\infty$  的边。因为这条边下界为 0,所以  $S^*$ , $T^*$  无影响,再直接求一次  $S^* \to T^*$  的最大流。若超级源点  $S^*$  出发的边全部满流,则  $T \to S$  边上的流量即为原图的最小流,否则无解。

## 4.8 最小费用最大流

#### 4.8.1 稀疏图

时间复杂度:  $\mathcal{O}(V \cdot E^2)$ 

```
struct EdgeList {
1
         int size;
2
         int last[N];
3
         int succ[M], other[M], flow[M], cost[M];
4
         void clear(int n) {
5
             size = 0;
6
              std::fill(last, last + n, -1);
8
         void add(int x, int y, int c, int w) {
9
             succ[size] = last[x];
10
             last[x] = size;
11
             other[size] = y;
12
             flow[size] = c;
13
             cost[size++] = w;
14
15
     } e;
16
17
     int n, source, target;
18
     int prev[N];
19
20
     void add(int x, int y, int c, int w) {
21
         e.add(x, y, c, w);
```

36 CHAPTER 4. 图论

```
23
         e.add(y, x, 0, -w);
     }
24
25
     bool augment() {
26
27
         static int dist[N], occur[N];
         std::vector<int> queue;
28
         std::fill(dist, dist + n, INT_MAX);
29
         std::fill(occur, occur + n, 0);
30
31
         dist[source] = 0;
         occur[source] = true;
32
         queue.push_back(source);
33
         for (int head = 0; head < (int)queue.size(); ++head) {</pre>
34
              int x = queue[head];
35
              for (int i = e.last[x]; ~i; i = e.succ[i]) {
36
                  int y = e.other[i];
37
                  if (e.flow[i] \&\& dist[y] > dist[x] + e.cost[i]) {
38
                      dist[y] = dist[x] + e.cost[i];
39
                      prev[y] = i;
40
                      if (!occur[y]) {
41
42
                           occur[y] = true;
                           queue.push_back(y);
43
44
                  }
45
              }
46
47
              occur[x] = false;
48
         return dist[target] < INT_MAX;</pre>
49
     }
50
51
     std::pair<int, int> solve() {
52
         std::pair<int, int> answer = std::make_pair(0, 0);
53
         while (augment()) {
54
              int number = INT_MAX;
55
              for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
56
57
                  number = std::min(number, e.flow[prev[i]]);
58
              answer.first += number;
59
              for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
60
61
                  e.flow[prev[i]] -= number;
                  e.flow[prev[i] ^ 1] += number;
62
                  answer.second += number * e.cost[prev[i]];
63
              }
64
65
         }
66
         return answer;
     }
67
```

#### 4.8.2 稠密图

使用条件: 费用非负 时间复杂度:  $\mathcal{O}(V \cdot E^2)$ 

```
int aug(int no,int res) {
1
         if(no == t) return cost += pi1 * res,res;
2
         v[no] = true;
3
         int flow = 0;
         for(int i = h[no]; ~ i ;i = nxt[i])
5
             if(cap[i] && !expense[i] && !v[p[i]]) {
6
                 int d = aug(p[i],min(res,cap[i]));
7
                 cap[i] = d, cap[i ^ 1] + d, flow + d, res = d;
                 if( !res ) return flow;
9
             }
10
         return flow;
11
     }
12
```

4.9. 一般图最大匹配 37

```
bool modlabel() {
13
         int d = maxint;
14
         for(int i = 1;i <= t;++ i)
15
              if(v[i]) {
16
17
                  for(int j = h[i]; ~ j ; j = nxt[j])
                       if(cap[j] \&\& !v[p[j]] \&\& expense[j] < d) d = expense[j];
18
              }
19
         if(d == maxint)return false;
20
21
         for(int i = 1;i <= t;++ i)
              if(v[i]) {
22
                  for(int j = h[i];~ j;j = nxt[j])
23
                      expense[j] -= d, expense[j ^ 1] += d;
24
25
         pi1 += d;
26
         return true;
27
28
29
     void minimum_cost_flow_zkw() {
         cost = 0;
30
         do{
31
32
              do{
                  memset(v,false,sizeof v);
33
              }while (aug(s,maxint));
34
         }while (modlabel());
35
     }
36
```

#### 4.9 一般图最大匹配

时间复杂度:  $\mathcal{O}(V^3)$ 

```
int match[N], belong[N], next[N], mark[N], visit[N];
1
     std::vector<int> queue;
3
     int find(int x) {
4
         if (belong[x] != x) {
5
6
              belong[x] = find(belong[x]);
7
         return belong[x];
8
9
10
     void merge(int x, int y) {
11
         x = find(x);
12
13
         y = find(y);
         if (x != y) {
14
              belong[x] = y;
15
         }
16
     }
17
18
     int lca(int x, int y) {
19
         static int stamp = 0;
20
21
         stamp++;
         while (true) {
22
              if (x != -1) {
23
                  x = find(x);
24
                  if (visit[x] == stamp) {
25
26
                      return x;
27
                  visit[x] = stamp;
28
29
                  if (match[x] != -1) {
                      x = next[match[x]];
30
                  } else {
31
32
                      x = -1;
33
```

38 CHAPTER 4. 图论

```
34
              std::swap(x, y);
35
         }
36
     }
37
38
     void group(int a, int p) {
39
          while (a != p) {
40
              int b = match[a], c = next[b];
41
42
              if (find(c) != p) {
                  next[c] = b;
43
              }
44
              if (mark[b] == 2) {
45
                  mark[b] = 1;
46
47
                  queue.push_back(b);
48
              if (mark[c] == 2) {
49
                  mark[c] = 1;
50
                  queue.push_back(c);
51
              }
52
53
              merge(a, b);
              merge(b, c);
54
              a = c;
55
         }
56
     }
57
58
     void augment(int source) {
59
          queue.clear();
60
61
          for (int i = 0; i < n; ++i) {
              next[i] = visit[i] = -1;
62
              belong[i] = i;
63
              mark[i] = 0;
64
65
          mark[source] = 1;
66
          queue.push_back(source);
67
          for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
68
              int x = queue[head];
69
              for (int i = 0; i < (int)edge[x].size(); ++i) {</pre>
70
                  int y = edge[x][i];
71
                  if (match[x] == y \mid \mid find(x) == find(y) \mid \mid mark[y] == 2) {
72
73
                       continue;
                  }
74
                  if (mark[y] == 1) {
75
76
                       int r = lca(x, y);
77
                       if (find(x) != r) {
                           next[x] = y;
78
                       }
79
                       if (find(y) != r) {
80
                           next[y] = x;
81
82
83
                       group(x, r);
                       group(y, r);
84
                  } else if (match[y] == -1) {
85
                       next[y] = x;
86
                       for (int u = y; u != -1; ) {
87
                           int v = next[u];
                           int mv = match[v];
89
                           match[v] = u;
90
                           match[u] = v;
91
                           u = mv;
92
                       }
93
                       break;
94
                  } else {
95
96
                       next[y] = x;
```

4.10. 无向图全局最小割 39

```
mark[y] = 2;
97
                        mark[match[y]] = 1;
98
                        queue.push_back(match[y]);
99
                   }
100
               }
101
          }
102
      }
103
104
105
      int solve() {
          std::fill(match, match + n, -1);
106
          for (int i = 0; i < n; ++i) {
107
               if (match[i] == -1) {
108
                   augment(i);
109
110
          }
111
          int answer = 0;
112
          for (int i = 0; i < n; ++i) {
113
               answer += (match[i] !=-1);
114
115
116
          return answer;
      }
117
```

## 4.10 无向图全局最小割

时间复杂度:  $\mathcal{O}(V^3)$ 注意事项: 处理重边时,应该对边权累加

```
int node[N], dist[N];
     bool visit[N];
2
3
     int solve(int n) {
         int answer = INT_MAX;
5
         for (int i = 0; i < n; ++i) {
6
             node[i] = i;
7
         }
         while (n > 1) {
9
             int max = 1;
10
             for (int i = 0; i < n; ++i) {
11
                  dist[node[i]] = graph[node[0]][node[i]];
12
                  if (dist[node[i]] > dist[node[max]]) {
13
                      max = i;
14
                  }
15
             }
16
             int prev = 0;
17
             memset(visit, 0, sizeof(visit));
18
             visit[node[0]] = true;
19
             for (int i = 1; i < n; ++i) {
20
                  if (i == n - 1) {
21
                      answer = std::min(answer, dist[node[max]]);
22
23
                      for (int k = 0; k < n; ++k) {
                           graph[node[k]][node[prev]] =
24
                               (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
25
^{26}
27
                      node[max] = node[--n];
28
                  visit[node[max]] = true;
29
30
                  prev = max;
31
                  \max = -1;
                  for (int j = 1; j < n; ++j) {
32
                      if (!visit[node[j]]) {
33
                           dist[node[j]] += graph[node[prev]][node[j]];
34
                           if (max == -1 || dist[node[max]] < dist[node[j]]) {</pre>
35
```

40 CHAPTER 4. 图论

## 4.11 最小树形图

```
int n, m, used[N], pass[N], eg[N], more, queue[N];
1
     double g[N][N];
2
3
     void combine(int id, double &sum) {
4
         int tot = 0, from, i, j, k;
5
         for (; id != 0 && !pass[id]; id = eg[id]) {
6
             queue[tot++] = id;
7
             pass[id] = 1;
8
9
10
         for (from = 0; from < tot && queue[from] != id; from++);
11
             (from == tot) return;
12
         more = 1;
13
14
         for (i = from; i < tot; i++) {
             sum += g[eg[queue[i]]][queue[i]];
15
             if (i != from) {
16
                  used[queue[i]] = 1;
17
                  for (j = 1; j \le n; j++) if (!used[j]) {
18
19
                      if (g[queue[i]][j] < g[id][j]) g[id][j] = g[queue[i]][j];
20
             }
21
         }
22
23
         for (i = 1; i <= n; i++) if (!used[i] && i != id) {
24
             for (j = from; j < tot; j++) {
25
                  k = queue[j];
26
                  if (g[i][id] > g[i][k] - g[eg[k]][k]) g[i][id] = g[i][k] - g[eg[k]][k];
27
28
         }
29
30
     }
31
     double mdst(int root) {
32
         int i, j, k;
33
34
         double sum = 0;
         memset(used, 0, sizeof(used));
35
         for (more = 1; more; ) {
36
             more = 0;
37
             memset(eg, 0, sizeof(eg));
38
             for (i = 1; i <= n; i++) if (!used[i] && i != root) {
39
                  for (j = 1, k = 0; j \le n; j++) if (!used[j] \&\& i != j)
40
                      if (k == 0 \mid | g[j][i] < g[k][i]) k = j;
41
                  eg[i] = k;
42
             }
43
44
             memset(pass, 0, sizeof(pass));
45
46
             for (i = 1; i <= n; i++) if (!used[i] && !pass[i] && i != root) combine(i, sum);
         }
47
48
         for (i = 1; i <= n; i++) if (!used[i] && i != root) sum += g[eg[i]][i];
49
50
         return sum;
```

4.12. 有根树的同构 41

51 }

## 4.12 有根树的同构

时间复杂度:  $\mathcal{O}(VlogV)$ 

```
const unsigned long long MAGIC = 4423;
1
2
     unsigned long long magic[N];
3
     std::pair<unsigned long long, int> hash[N];
5
     void solve(int root) {
6
         magic[0] = 1;
         for (int i = 1; i <= n; ++i) {
              magic[i] = magic[i - 1] * MAGIC;
9
         }
10
         std::vector<int> queue;
11
         queue.push_back(root);
12
         for (int head = 0; head < (int)queue.size(); ++head) {</pre>
13
              int x = queue[head];
14
              for (int i = 0; i < (int)son[x].size(); ++i) {</pre>
15
16
                  int y = son[x][i];
                  queue.push_back(y);
17
              }
18
19
         for (int index = n - 1; index >= 0; --index) {
20
              int x = queue[index];
21
              hash[x] = std::make_pair(0, 0);
22
23
              std::vector<std::pair<unsigned long long, int> > value;
24
              for (int i = 0; i < (int)son[x].size(); ++i) {</pre>
25
^{26}
                  int y = son[x][i];
27
                  value.push_back(hash[y]);
28
              std::sort(value.begin(), value.end());
29
30
              hash[x].first = hash[x].first * magic[1] + 37;
31
              hash[x].second++;
32
              for (int i = 0; i < (int)value.size(); ++i) {</pre>
33
                  hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
34
                  hash[x].second += value[i].second;
35
36
              hash[x].first = hash[x].first * magic[1] + 41;
37
38
              hash[x].second++;
         }
39
     }
40
```

## 4.13 度限制生成树

```
int n, m, S, K, ans , cnt , Best[N], fa[N], FE[N];
1
     int f[N], p[M], t[M], c[M], o, Cost[N];
     bool u[M], d[M];
     pair<int, int> MinCost[N];
     struct Edge {
5
6
         int a, b, c;
         bool operator < (const Edge & E) const { return c < E.c; }</pre>
     }E[M];
8
     vector<int> SE;
9
10
     inline int F(int x) {
         return fa[x] == x ? x : fa[x] = F(fa[x]);
11
```

42 CHAPTER 4. 图论

```
}
12
     inline void AddEdge(int a, int b, int C) {
13
         p[++o] = b; c[o] = C;
14
         t[o] = f[a]; f[a] = o;
15
16
     void dfs(int i, int father) {
17
         fa[i] = father;
18
19
         if (father == S) Best[i] = -1;
20
         else {
             Best[i] = i;
21
              if (~Best[father] && Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
22
23
         for (int j = f[i]; j; j = t[j])
24
         if (!d[j] && p[j] != father) {
25
             Cost[p[j]] = c[j];
26
             FE[p[j]] = j;
27
              dfs(p[j], i);
28
         }
29
30
31
     inline bool Kruskal() {
32
         cnt = n - 1, ans = 0; o = 1;
         for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
33
         sort(E + 1, E + m + 1);
34
         for (int i = 1; i <= m; i++) {
35
36
              if (E[i].b == S) swap(E[i].a, E[i].b);
              if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
37
                  fa[F(E[i].a)] = F(E[i].b);
38
                  ans += E[i].c;
39
                  cnt --;
40
                  u[i] = true;
41
                  AddEdge(E[i].a, E[i].b, E[i].c);
42
                  AddEdge(E[i].b, E[i].a, E[i].c);
43
             }
44
45
         for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);</pre>
46
         for (int i = 1; i <= m; i++)
47
         if (E[i].a == S) {
48
             SE.push_back(i);
49
             MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
50
         }
51
         int dif = 0;
52
         for (int i = 1; i <= n; i++)
53
         if (i != S && fa[i] == i) {
54
              if (MinCost[i].second == INF) return false;
55
              if (++ dif > K) return false;
56
             dfs(E[MinCost[i].second].b, S);
57
             u[MinCost[i].second] = true;
58
             ans += MinCost[i].first;
59
60
61
         return true;
62
     bool Solve() {
63
         memset(d,false,sizeof d);
64
         memset(u,false,sizeof u);
65
         if (!Kruskal()) return false;
66
         for (int i = cnt + 1; i <= K && i <= n; i++) {
67
              int MinD = INF, MinID = -1;
68
             for (int j = (int) SE.size() - 1; j >= 0; j--)
69
              if (u[SE[j]])
70
                  SE.erase(SE.begin() + j);
71
             for (int j = 0; j < (int) SE.size(); j++) {</pre>
72
                  int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
73
74
                  if (tmp < MinD) {</pre>
```

4.14. 弦图相关 43

```
75
                       MinD = tmp;
                       MinID= SE[j];
76
                  }
77
              }
78
              if (MinID == -1) return true;
              if (MinD >= 0) break;
80
              ans += MinD;
81
82
              u[MinID] = true;
83
              d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
              dfs(E[MinID].b, S);
84
         }
85
86
         return true;
     }
87
     int main(){
88
         Solve();
89
         return 0;
90
91
```

### 4.14 弦图相关

#### 4.14.1 弦图的判定

```
int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L[1001],
     \rightarrow R[1001],
     v[1001], idx[1001], pos[1001];
2
     bool b[1001][1001];
3
4
     inline void makelist(int x, int y){
5
         where [++1] = y;
6
         next[1] = first[x];
         first[x] = 1;
8
     }
9
10
     bool cmp(const int &x, const int &y){
11
12
         return(idx[x] < idx[y]);
13
14
     int main(){
15
         for (;;)
16
17
             n = read(); m = read();
18
19
             if (!n && !m) return 0;
             memset(first, 0, sizeof(first)); l = 0;
20
             memset(b, false, sizeof(b));
21
             for (int i = 1; i <= m; i++)
22
23
                  int x = read(), y = read();
24
                  if (x != y \&\& !b[x][y])
25
26
27
                     b[x][y] = true; b[y][x] = true;
                     makelist(x, y); makelist(y, x);
28
                  }
29
30
             memset(f, 0, sizeof(f));
31
             memset(L, 0, sizeof(L));
32
             memset(R, 255, sizeof(R));
33
             L[0] = 1; R[0] = n;
34
35
             for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
             memset(idx, 0, sizeof(idx));
36
             memset(v, 0, sizeof(v));
37
             for (int i = n; i; --i)
38
39
```

44 CHAPTER 4. 图论

```
40
                  int now = c[i];
                  R[f[now]]--;
41
                  if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
42
                  idx[now] = i; v[i] = now;
43
                  for (int x = first[now]; x; x = next[x])
44
                      if (!idx[where[x]])
45
                      {
46
47
                         swap(c[pos[where[x]]], c[R[f[where[x]]]]);
48
                         pos[c[pos[where[x]]]] = pos[where[x]];
                         pos[where[x]] = R[f[where[x]]];
49
                         L[f[where[x]] + 1] = R[f[where[x]]]--;
50
                         if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
51
                         if (R[f[where[x]] + 1] == -1)
52
                             R[f[where[x]] + 1] = L[f[where[x]] + 1];
53
                         ++f [where[x]];
54
                      }
55
             }
56
             bool ok = true;
57
             //v 是完美消除序列.
58
             for (int i = 1; i <= n && ok; i++)
59
60
                  int cnt = 0;
61
                  for (int x = first[v[i]]; x; x = next[x])
62
                      if (idx[where[x]] > i) c[++cnt] = where[x];
63
                  sort(c + 1, c + cnt + 1, cmp);
64
                  bool can = true;
65
                  for (int j = 2; j \le cnt; j++)
66
                      if (!b[c[1]][c[j]])
67
68
                          ok = false;
69
                          break;
70
71
72
             if (ok) printf("Perfect\n");
73
             else printf("Imperfect\n");
74
             printf("\n");
75
         }
76
     }
77
```

#### 4.14.2 弦图的闭数

```
int n, m, first[100001], next[2000001], where[2000001], 1, L[100001], R[100001], c[100001],
     \hookrightarrow f[100001],
     pos[100001], idx[100001], v[100001], ans;
2
3
     inline void makelist(int x, int y){
         where [++1] = y;
5
         next[l] = first[x];
6
         first[x] = 1;
7
     }
8
9
     int read(){
10
11
         char ch;
         for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
12
         int cnt = 0;
13
         for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
14
         return(cnt);
15
16
     }
17
     int main(){
18
         //freopen("1006.in", "r", stdin);
19
         //freopen("1006.out", "w", stdout);
20
```

```
memset(first, 0, sizeof(first)); 1 = 0;
21
         n = read(); m = read();
22
         for (int i = 1; i <= m; i++)
23
24
             int x, y;
25
             x = read(); y = read();
26
             makelist(x, y); makelist(y, x);
27
         }
28
         memset(L, 0, sizeof(L));
         memset(R, 255, sizeof(R));
30
         memset(f, 0, sizeof(f));
31
         memset(idx, 0, sizeof(idx));
32
         for (int i = 1; i \le n; i++) c[i] = i, pos[i] = i;
33
         L[0] = 1; R[0] = n; ans = 0;
34
         for (int i = n; i; --i)
35
36
             int now = c[i], cnt = 1;
37
             idx[now] = i; v[i] = now;
38
             if (-R[f[now]] < L[f[now]]) R[f[now]] = -1;
39
             for (int x = first[now]; x; x = next[x])
40
                  if (!idx[where[x]])
41
                  {
42
                      swap(c[pos[where[x]]], c[R[f[where[x]]]]);
43
                      pos[c[pos[where[x]]]] = pos[where[x]];
                      pos[where[x]] = R[f[where[x]]];
45
                      L[f[where[x]] + 1] = R[f[where[x]]] --;
46
                      if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;</pre>
47
                      if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
48
49
                      ++f[where[x]];
                  }
50
51
                  else ++cnt;
             ans = max(ans, cnt);
52
53
         printf("%d\n", ans);
54
     }
55
```

## 4.15 哈密尔顿回路 (ORE 性质的图)

ORE 性质:

```
\forall x, y \in V \land (x, y) \notin E \quad s.t. \quad deg_x + deg_y \ge n
```

返回结果: 从顶点1出发的一个哈密尔顿回路

使用条件:  $n \ge 3$ 

```
int left[N], right[N], next[N], last[N];
1
2
     void cover(int x) {
3
         left[right[x]] = left[x];
4
         right[left[x]] = right[x];
5
     }
6
     int adjacent(int x) {
8
         for (int i = right[0]; i <= n; i = right[i]) {
9
              if (graph[x][i]) {
10
                  return i;
11
12
13
14
         return 0;
     }
15
16
17
     std::vector<int> solve() {
         for (int i = 1; i <= n; ++i) {
18
```

46 CHAPTER 4. 图论

```
left[i] = i - 1;
19
              right[i] = i + 1;
20
21
         int head, tail;
22
23
         for (int i = 2; i <= n; ++i) {
              if (graph[1][i]) {
24
                  head = 1;
25
                  tail = i;
26
27
                  cover(head);
                  cover(tail);
28
                  next[head] = tail;
29
                  break;
30
              }
31
32
         while (true) {
33
              int x;
34
              while (x = adjacent(head)) {
35
                  next[x] = head;
36
                  head = x;
37
38
                  cover(head);
              }
39
              while (x = adjacent(tail)) {
40
                  next[tail] = x;
41
42
                  tail = x;
43
                  cover(tail);
44
              if (!graph[head][tail]) {
45
46
                  for (int i = head, j; i != tail; i = next[i]) {
                      if (graph[head][next[i]] && graph[tail][i]) {
47
                           for (j = head; j != i; j = next[j]) {
48
                               last[next[j]] = j;
49
50
                           j = next[head];
51
                           next[head] = next[i];
52
                           next[tail] = i;
53
                           tail = j;
                           for (j = i; j != head; j = last[j]) {
55
                               next[j] = last[j];
56
                           }
57
58
                           break;
                      }
59
                  }
60
              }
61
              next[tail] = head;
62
              if (right[0] > n) {
63
                  break;
64
65
              for (int i = head; i != tail; i = next[i]) {
66
                  if (adjacent(i)) {
67
68
                      head = next[i];
                      tail = i;
69
                      next[tail] = 0;
70
                      break;
71
                  }
72
              }
73
         }
74
         std::vector<int> answer;
75
         for (int i = head; ; i = next[i]) {
76
              if (i == 1) {
77
                  answer.push_back(i);
78
                  for (int j = next[i]; j != i; j = next[j]) {
79
                       answer.push_back(j);
80
                  }
81
```

```
answer.push_back(i);
82
                  break;
83
             }
84
             if (i == tail) {
85
86
                  break;
             }
87
         }
88
         return answer;
89
     }
90
```

## Chapter 5

# 字符串

## 5.1 模式串匹配

```
void build(char *pattern) {
1
2
         int length = (int)strlen(pattern + 1);
3
         fail[0] = -1;
         for (int i = 1, j; i <= length; ++i) {</pre>
4
             for (j = fail[i - 1]; j != -1 \&\& pattern[i] != pattern[j + 1]; j = fail[j]);
5
             fail[i] = j + 1;
6
7
     }
8
9
     void solve(char *text, char *pattern) {
10
         int length = (int)strlen(text + 1);
11
         for (int i = 1, j; i <= length; ++i) {</pre>
12
             for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);
13
             match[i] = j + 1;
14
         }
15
     }
16
```

## 5.2 坚固的模式串匹配

```
lenA = strlen(A); lenB = strlen(B);
     nxt[0] = lenB, nxt[1] = lenB - 1;
     for (int i = 0;i <= lenB;i ++)</pre>
3
         if (B[i] != B[i + 1]) {nxt[1] = i; break;}
     int j, k = 1, p, L;
     for (int i = 2;i < lenB;i ++) {
6
         p = k + nxt[k] - 1; L = nxt[i - k];
7
         if (i + L <= p) nxt[i] = L;
8
9
         else {
10
              j = p - i + 1;
              if (j < 0) j = 0;
11
              while (i + j < lenB \&\& B[i + j] == B[j]) j++;
12
13
              nxt[i] = j; k = i;
         }
14
15
     int minlen = lenA <= lenB ? lenA : lenB; ex[0] = minlen;</pre>
16
17
     for (int i = 0;i < minlen;i ++)</pre>
         if (A[i] != B[i]) {ex[0] = i; break;}
18
     k = 0;
19
     for (int i = 1;i < lenA;i ++){</pre>
20
21
         p = k + ex[k] - 1; L = next[i - k];
         if (i + L <= p) ex[i] = L;
22
         else {
23
              j = p - i + 1;
24
              if (j < 0) j = 0;
25
```

5.3. AC 自动机 49

```
26 while (i + j < lenA && j < lenB && A[i + j] == B[j]) j++;
27 ex[i] = j; k = i;
28 }
29 }
```

## 5.3 AC 自动机

```
1
     int size, c[MAXT][26], f[MAXT], fail[MAXT], d[MAXT];
2
     int alloc() {
3
4
         size++;
         std::fill(c[size], c[size] + 26, 0);
5
         f[size] = fail[size] = d[size] = 0;
6
         return size;
7
9
     void insert(char *s) {
10
11
         int len = strlen(s + 1), p = 1;
         for (int i = 1; i <= len; i++) {
12
             if (c[p][s[i] - 'a']) p = c[p][s[i] - 'a'];
13
             else{
14
                  int newnode = alloc();
15
                  c[p][s[i] - 'a'] = newnode;
16
                  d[newnode] = s[i] - 'a';
17
                  f[newnode] = p;
18
19
                  p = newnode;
             }
20
         }
21
     }
22
23
24
     void buildfail() {
         static int q[MAXT];
25
         int left = 0, right = 0;
26
27
         fail[1] = 0;
         for (int i = 0; i < 26; i++) {
28
             c[0][i] = 1;
29
              if (c[1][i]) q[++right] = c[1][i];
30
31
         while (left < right) {
32
             left++;
33
             int p = fail[f[q[left]]];
34
35
             while (!c[p][d[q[left]]]) p = fail[p];
             fail[q[left]] = c[p][d[q[left]]];
36
             for (int i = 0; i < 26; i++) {</pre>
37
                  if (c[q[left]][i]) {
38
                      q[++right] = c[q[left]][i];
39
40
             }
41
42
         }
43
         for (int i = 1; i <= size; i++)
             for (int j = 0; j < 26; j++) {
44
                  int p = i;
45
                  while (!c[p][j]) p = fail[p];
46
                  c[i][j] = c[p][j];
47
             }
48
     }
49
```

50 CHAPTER 5. 字符串

## 5.4 后缀数组

```
namespace suffix_array{
1
         int wa[MAXN], wb[MAXN], ws[MAXN], wv[MAXN];
2
         bool cmp(int *r, int a, int b, int l) {
3
             return r[a] == r[b] \&\& r[a + 1] == r[b + 1];
4
5
         void DA(int *r, int *sa, int n, int m) {
6
             int *x = wa, *y = wb, *t;
             for (int i = 0; i < m; i++) ws[i] = 0;
             for (int i = 0; i < n; i++) ws[x[i] = r[i]]++;
9
             for (int i = 1; i < m; i++) ws[i] += ws[i - 1];
10
             for (int i = n - 1; i \ge 0; i--) sa[--ws[x[i]]] = i;
11
             for (int i, j = 1, p = 1; p < n; j <<= 1, m = p) {
12
                 for (p = 0, i = n - j; i < n; i++) y[p++] = i;
13
                 for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
14
                 for (i = 0; i < n; i++) wv[i] = x[y[i]];
15
16
                 for (i = 0; i < m; i++) ws[i] = 0;
                 for (i = 0; i < n; i++) ws[wv[i]]++;
17
                 for (i = 1; i < m; i++) ws[i] += ws[i-1];
18
                 for (i = n - 1; i \ge 0; i--) sa[--ws[wv[i]]] = y[i];
19
                 for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
20
                     x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
21
             }
22
23
24
         void getheight(int *r, int *sa, int *rk, int *h, int n) {
             for (int i = 1; i <= n; i++) rk[sa[i]] = i;
25
             for (int i = 0, j, k = 0; i < n; h[rk[i++]] = k)
26
27
                 for (k ? k-- : 0, j = sa[rk[i] - 1]; r[i + k] == r[j + k]; k++);
28
         }
     };
29
```

## 5.5 广义后缀自动机

```
// Generalized Suffix Automaton
     void add(int x, int &last) {
2
         int lastnode = last;
3
         if (c[lastnode][x]) {
4
             int nownode = c[lastnode][x];
5
             if (l[nownode] == l[lastnode] + 1) last = nownode;
6
             else{
                 int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
8
                 for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
9
                 f[auxnode] = f[nownode]; f[nownode] = auxnode;
10
                 for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
11
                      c[lastnode][x] = auxnode;
12
13
                 last = auxnode;
14
             }
15
         }
16
         else{
17
             int newnode = ++size; l[newnode] = l[lastnode] + 1;
18
             for (; lastnode && !c[lastnode][x]; lastnode = f[lastnode]) c[lastnode][x] = newnode;
19
             if (!lastnode) f[newnode] = 1;
20
             else{
21
                 int nownode = c[lastnode][x];
22
23
                 if (l[lastnode] + 1 == l[nownode]) f[newnode] = nownode;
                 else{
24
                      int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
25
                      for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];</pre>
26
                      f[auxnode] = f[nownode]; f[nownode] = f[newnode] = auxnode;
27
```

5.6. MANACHER 算法 51

## 5.6 Manacher 算法

```
void manacher(char *text, int length) {
1
         palindrome[0] = 1;
2
3
         for (int i = 1, j = 0; i < length; ++i) {
              if (j + palindrome[j] <= i) {</pre>
                  palindrome[i] = 0;
5
              } else {
6
                  palindrome[i] = std::min(palindrome[(j << 1) - i], j + palindrome[j] - i);</pre>
              while (i - palindrome[i] >= 0 && i + palindrome[i] < length</pre>
9
                      && text[i - palindrome[i]] == text[i + palindrome[i]]) {
10
                  palindrome[i]++;
11
12
              if (i + palindrome[i] > j + palindrome[j]) {
13
14
              }
15
         }
16
     }
17
```

## 5.7 回文树

```
struct Palindromic_Tree{
2
         int nTree, nStr, last, c[MAXT][26], fail[MAXT], r[MAXN], l[MAXN], s[MAXN];
         int allocate(int len) {
3
             l[nTree] = len;
             r[nTree] = 0;
5
             fail[nTree] = 0;
6
             memset(c[nTree], 0, sizeof(c[nTree]));
7
             return nTree++;
         }
         void init() {
10
             nTree = nStr = 0;
11
             int newEven = allocate(0);
12
             int newOdd = allocate(-1);
13
             last = newEven;
14
             fail[newEven] = newOdd;
15
             fail[newOdd] = newEven;
16
             s[0] = -1;
17
         }
18
         void add(int x) {
19
             s[++nStr] = x;
20
             int nownode = last;
21
             while (s[nStr - 1[nownode] - 1] != s[nStr]) nownode = fail[nownode];
22
23
             if (!c[nownode][x]) {
24
                 int newnode = allocate(l[nownode] + 2), &newfail = fail[newnode];
                 newfail = fail[nownode];
25
                 while (s[nStr - l[newfail] - 1] != s[nStr]) newfail = fail[newfail];
26
27
                 newfail = c[newfail][x];
                  c[nownode][x] = newnode;
28
```

52 CHAPTER 5. 字符串

```
29
              last = c[nownode][x];
30
              r[last]++;
31
         }
32
         void count() {
33
              for (int i = nTree - 1; i >= 0; i--) {
34
                  r[fail[i]] += r[i];
35
36
         }
37
38
     }
```

## 5.8 循环串最小表示

```
int solve(char *text, int length) {
1
2
         int i = 0, j = 1, delta = 0;
         while (i < length && j < length && delta < length) {
3
             char tokeni = text[(i + delta) % length];
4
             char tokenj = text[(j + delta) % length];
5
             if (tokeni == tokenj) {
6
                  delta++;
7
8
             } else {
9
                  if (tokeni > tokenj) {
                      i += delta + 1;
10
                  } else {
11
                      j += delta + 1;
12
                  }
13
                  if (i == j) {
14
                      j++;
15
16
17
                  delta = 0;
             }
18
         }
19
         return std::min(i, j);
20
21
     }
```

## Chapter 6

# 计算几何

## 6.1 二维基础

#### 6.1.1 点类

```
struct Point{
         double x, y;
2
         Point() {}
3
         Point(double x, double y):x(x), y(y) {}
         Point operator +(const Point &p)const {
5
             return Point(x + p.x, y + p.y);
6
         Point operator -(const Point &p)const {
             return Point(x - p.x, y - p.y);
9
10
         Point operator *(const double &p)const {
11
12
             return Point(x * p, y * p);
13
         Point operator /(const double &p)const {
14
             return Point(x / p, y / p);
15
16
         int read() {
17
             return scanf("%lf%lf", &x, &y);
18
         }
19
20
     };
21
     struct Line{
22
         Point a, b;
23
^{24}
         Line(Point a, Point b):a(a), b(b) {}
25
     };
26
```

#### 6.1.2 凸包

```
bool Pair_Comp(const Point &a, const Point &b) {
1
2
         if (dcmp(a.x - b.x) < 0) return true;
         if (dcmp(a.x - b.x) > 0) return false;
3
         return dcmp(a.y - b.y) < 0;</pre>
4
5
     int Convex_Hull(int n, Point *P, Point *C) {
         sort(P, P + n, Pair_Comp);
8
         int top = 0;
9
10
         for (int i = 0; i < n; i++) {
             while (top \geq 2 && dcmp(det(C[top - 1] - C[top - 2], P[i] - C[top - 2])) \leq 0) top--;
11
             C[top++] = P[i];
12
13
         int lasttop = top;
14
```

CHAPTER 6. 计算几何

#### 6.1.3 半平面交

```
bool isOnLeft(const Point &x, const Line &1) {
1
         double d = det(x - 1.a, 1.b - 1.a);
2
         return dcmp(d) <= 0;
3
4
     // 传入一个线段的集合 L, 传出 A, 并且返回 A 的大小
5
6
     int getIntersectionOfHalfPlane(int n, Line *L, Line *A) {
         Line *q = new Line[n + 1];
         Point *p = new Point[n + 1];
8
         sort(L, L + n, Polar_Angle_Comp_Line);
9
10
         int 1 = 1, r = 0;
         for (int i = 0; i < n; i++) {
11
             while (1 < r \&\& !isOnLeft(p[r - 1], L[i])) r--;
12
             while (1 < r \&\& !isOnLeft(p[1], L[i])) 1++;
13
             q[++r] = L[i];
14
             if (1 < r \&\& is\_Colinear(q[r], q[r - 1])) {
15
16
                 if (isOnLeft(L[i].a, q[r])) q[r] = L[i];
17
             }
18
             if (1 < r) p[r - 1] = getIntersection(q[r - 1], q[r]);
19
20
         while (1 < r \&\& !isOnLeft(p[r - 1], q[1])) r--;
21
         if (r - 1 + 1 \le 2) return 0;
22
         int tot = 0;
23
         for (int i = 1; i <= r; i++) A[tot++] = q[i];
24
25
         return tot;
     }
26
```

#### 6.1.4 最近点对

```
7bool comparex(const Point &a, const Point &b) {
1
2
         return sgn(a.x - b.x) < 0;
3
4
     bool comparey(const Point &a, const Point &b) {
5
         return sgn(a.y - b.y) < 0;
6
8
     double solve(const std::vector<Point> &point, int left, int right) {
9
         if (left == right) {
10
             return INF;
11
         }
12
         if (left + 1 == right) {
13
             return dist(point[left], point[right]);
14
15
         int mid = left + right >> 1;
16
         double result = std::min(solve(left, mid), solve(mid + 1, right));
17
18
         std::vector<Point> candidate;
         for (int i = left; i <= right; ++i) {</pre>
19
             if (std::abs(point[i].x - point[mid].x) <= result) {</pre>
20
21
                  candidate.push_back(point[i]);
22
```

6.2. 三维基础 55

```
23
         std::sort(candidate.begin(), candidate.end(), comparey);
24
         for (int i = 0; i < (int)candidate.size(); ++i) {</pre>
25
             for (int j = i + 1; j < (int)candidate.size(); ++j) {
26
                  if (std::abs(candidate[i].y - candidate[j].y) >= result) {
27
28
                      break;
                  }
29
30
                  result = std::min(result, dist(candidate[i], candidate[j]));
31
             }
32
         return result;
33
     }
34
35
36
     double solve(std::vector<Point> point) {
         std::sort(point.begin(), point.end(), comparex);
37
         return solve(point, 0, (int)point.size() - 1);
38
39
```

#### 6.2 三维基础

#### 6.2.1 点类

```
int dcmp(const double &x) {
1
2
         return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1);
3
4
     struct TPoint{
5
         double x, y, z;
6
         TPoint() {}
7
         TPoint(double x, double y, double z) : x(x), y(y), z(z) {}
8
         TPoint operator +(const TPoint &p)const {
             return TPoint(x + p.x, y + p.y, z + p.z);
10
11
12
         TPoint operator -(const TPoint &p)const {
13
             return TPoint(x - p.x, y - p.y, z - p.z);
14
         TPoint operator *(const double &p)const {
15
             return TPoint(x * p, y * p, z * p);
16
17
         TPoint operator /(const double &p)const {
18
             return TPoint(x / p, y / p, z / p);
19
20
         bool operator <(const TPoint &p)const {</pre>
21
              int dX = dcmp(x - p.x), dY = dcmp(y - p.y), dZ = dcmp(z - p.z);
22
             return dX < 0 \mid \mid (dX == 0 \&\& (dY < 0 \mid \mid (dY == 0 \&\& dZ < 0)));
23
24
         }
         bool read() {
25
             return scanf("%lf%lf", &x, &y, &z) == 3;
26
         }
27
28
     };
29
     double sqrdist(const TPoint &a) {
30
31
         double ret = 0;
         ret += a.x * a.x;
32
33
         ret += a.y * a.y;
         ret += a.z * a.z;
34
35
         return ret;
36
     double sqrdist(const TPoint &a, const TPoint &b) {
37
         double ret = 0;
38
         ret += (a.x - b.x) * (a.x - b.x);
39
         ret += (a.y - b.y) * (a.y - b.y);
40
```

CHAPTER 6. 计算几何

```
ret += (a.z - b.z) * (a.z - b.z);
41
42
         return ret;
43
     double dist(const TPoint &a) {
44
         return sqrt(sqrdist(a));
45
46
     double dist(const TPoint &a, const TPoint &b) {
47
48
         return sqrt(sqrdist(a, b));
49
     TPoint det(const TPoint &a, const TPoint &b) {
50
         TPoint ret;
51
         ret.x = a.y * b.z - b.y * a.z;
52
         ret.y = a.z * b.x - b.z * a.x;
53
         ret.z = a.x * b.y - b.x * a.y;
54
55
         return ret;
56
     double dot(const TPoint &a, const TPoint &b) {
57
         double ret = 0;
58
         ret += a.x * b.x;
59
         ret += a.y * b.y;
60
         ret += a.z * b.z;
61
         return ret;
62
63
     double detdot(const TPoint &a, const TPoint &b, const TPoint &c, const TPoint &d) {
64
65
         return dot(det(b - a, c - a), d - a);
66
```

#### 6.2.2 凸包

```
struct Triangle{
1
2
         TPoint a, b, c;
         Triangle() {}
3
         Triangle(TPoint \ a, \ TPoint \ b, \ TPoint \ c) \ : \ a(a), \ b(b), \ c(c) \ \{\}
4
5
         double getArea() {
6
              TPoint ret = det(b - a, c - a);
              return dist(ret) / 2.0;
         }
8
     };
9
     namespace Convex_Hull {
10
         struct Face{
11
              int a, b, c;
12
              bool isOnConvex;
13
              Face() {}
14
              Face(int a, int b, int c) : a(a), b(b), c(c) {}
15
         };
16
17
         int nFace, left, right, whe[MAXN][MAXN];
18
         Face queue[MAXF], tmp[MAXF];
19
20
         bool isVisible(const std::vector<TPoint> &p, const Face &f, const TPoint &a) {
21
              return dcmp(detdot(p[f.a], p[f.b], p[f.c], a)) > 0;
22
         }
23
24
         bool init(std::vector<TPoint> &p) {
25
26
              bool check = false;
              for (int i = 1; i < (int)p.size(); i++) {</pre>
27
                  if (dcmp(sqrdist(p[0], p[i]))) {
28
29
                       std::swap(p[1], p[i]);
                       check = true;
30
                       break;
31
                  }
32
              }
33
```

6.2. 三维基础 57

```
if (!check) return false;
34
              check = false;
35
             for (int i = 2; i < (int)p.size(); i++) {</pre>
36
                  if (dcmp(sqrdist(det(p[i] - p[0], p[1] - p[0])))) {
37
                      std::swap(p[2], p[i]);
38
                      check = true;
39
                      break;
40
                  }
41
42
             }
              if (!check) return false;
43
              check = false;
44
             for (int i = 3; i < (int)p.size(); i++) {</pre>
45
                  if (dcmp(detdot(p[0], p[1], p[2], p[i]))) {
46
                      std::swap(p[3], p[i]);
47
                      check = true;
48
                      break;
49
50
              }
51
             if (!check) return false;
52
             for (int i = 0; i < (int)p.size(); i++)</pre>
53
                  for (int j = 0; j < (int)p.size(); j++) {</pre>
54
                      whe[i][j] = -1;
55
                  }
56
57
             return true;
         }
58
59
         void pushface(const int &a, const int &b, const int &c) {
60
             nFace++;
61
              tmp[nFace] = Face(a, b, c);
62
              tmp[nFace].isOnConvex = true;
63
             whe[a][b] = nFace;
64
             whe[b][c] = nFace;
65
             whe[c][a] = nFace;
66
         }
67
68
         bool deal(const std::vector<TPoint> &p, const std::pair<int, int> &now, const TPoint &base)
69
              int id = whe[now.second][now.first];
70
              if (!tmp[id].isOnConvex) return true;
71
72
              if (isVisible(p, tmp[id], base)) {
                  queue[++right] = tmp[id];
73
                  tmp[id].isOnConvex = false;
74
                  return true;
75
             }
76
             return false;
77
         }
78
79
         std::vector<Triangle> getConvex(std::vector<TPoint> &p) {
80
             static std::vector<Triangle> ret;
81
             ret.clear();
82
              if (!init(p)) return ret;
83
              if (!isVisible(p, Face(0, 1, 2), p[3])) pushface(0, 1, 2); else pushface(0, 2, 1);
84
              if (!isVisible(p, Face(0, 1, 3), p[2])) pushface(0, 1, 3); else pushface(0, 3, 1);
85
              if (!isVisible(p, Face(0, 2, 3), p[1])) pushface(0, 2, 3); else pushface(0, 3, 2);
86
              if (!isVisible(p, Face(1, 2, 3), p[0])) pushface(1, 2, 3); else pushface(1, 3, 2);
             for (int a = 4; a < (int)p.size(); a++) {
88
                  TPoint base = p[a];
89
                  for (int i = 1; i <= nFace; i++) {
90
                      if (tmp[i].isOnConvex && isVisible(p, tmp[i], base)) {
91
                           left = 0, right = 0;
92
                           queue[++right] = tmp[i];
93
94
                           tmp[i].isOnConvex = false;
95
                           while (left < right) {
```

58 CHAPTER 6. 计算几何

```
96
                                Face now = queue[++left];
                                if (!deal(p, std::make_pair(now.a, now.b), base)) pushface(now.a,
97
      → now.b, a);
                                if (!deal(p, std::make_pair(now.b, now.c), base)) pushface(now.b,
98
          now.c, a);
                                if (!deal(p, std::make_pair(now.c, now.a), base)) pushface(now.c,
99
          now.a, a);
100
101
                            break;
                       }
102
                   }
103
              }
104
              for (int i = 1; i <= nFace; i++) {
105
                   Face now = tmp[i];
106
                   if (now.isOnConvex) {
107
                       ret.push_back(Triangle(p[now.a], p[now.b], p[now.c]));
108
109
               }
110
              return ret;
111
112
          }
      };
113
114
115
      int n;
116
      std::vector<TPoint> p;
      std::vector<Triangle> answer;
117
118
      int main() {
119
          scanf("%d", &n);
120
          for (int i = 1; i <= n; i++) {
121
              TPoint a;
122
123
              a.read();
              p.push_back(a);
124
125
          answer = Convex_Hull::getConvex(p);
126
127
          double areaCounter = 0.0;
          for (int i = 0; i < (int)answer.size(); i++) {</pre>
128
               areaCounter += answer[i].getArea();
129
130
          printf("%.3f\n", areaCounter);
131
132
          return 0;
      }
133
```

#### 6.2.3 绕轴旋转

使用方法及注意事项: 逆时针绕轴 AB 旋转  $\theta$  角

```
1
     Matrix getTrans(const double &a, const double &b, const double &c) {
2
         Matrix ret;
         ret.a[0][0] = 1; ret.a[0][1] = 0; ret.a[0][2] = 0; ret.a[0][3] = 0;
3
         ret.a[1][0] = 0; ret.a[1][1] = 1; ret.a[1][2] = 0; ret.a[1][3] = 0;
4
         ret.a[2][0] = 0; ret.a[2][1] = 0; ret.a[2][2] = 1; ret.a[2][3] = 0;
5
         ret.a[3][0] = a; ret.a[3][1] = b; ret.a[3][2] = c; ret.a[3][3] = 1;
6
7
         return ret;
8
     Matrix getRotate(const double &a, const double &b, const double &c, const double &theta) {
9
         Matrix ret;
10
         ret.a[0][0] = a * a * (1 - cos(theta)) + cos(theta);
11
         ret.a[0][1] = a * b * (1 - cos(theta)) + c * sin(theta);
12
13
         ret.a[0][2] = a * c * (1 - cos(theta)) - b * sin(theta);
         ret.a[0][3] = 0;
14
15
         ret.a[1][0] = b * a * (1 - cos(theta)) - c * sin(theta);
16
         ret.a[1][1] = b * b * (1 - cos(theta)) + cos(theta);
17
```

6.3. 多边形 59

```
ret.a[1][2] = b * c * (1 - cos(theta)) + a * sin(theta);
18
19
         ret.a[1][3] = 0;
20
         ret.a[2][0] = c * a * (1 - cos(theta)) + b * sin(theta);
21
         ret.a[2][1] = c * b * (1 - cos(theta)) - a * sin(theta);
22
         ret.a[2][2] = c * c * (1 - cos(theta)) + cos(theta);
23
         ret.a[2][3] = 0;
24
25
26
         ret.a[3][0] = 0;
         ret.a[3][1] = 0;
27
         ret.a[3][2] = 0;
28
29
         ret.a[3][3] = 1;
30
         return ret;
31
     Matrix getRotate(const double &ax, const double &ay, const double &az, const double &bx, const
32
        double &by, const double &bz, const double &theta) {
         double 1 = dist(Point(0, 0, 0), Point(bx, by, bz));
33
         Matrix ret = getTrans(-ax, -ay, -az);
34
         ret = ret * getRotate(bx / 1, by / 1, bz / 1, theta);
35
         ret = ret * getTrans(ax, ay, az);
36
37
         return ret;
     }
38
```

## 6.3 多边形

### 6.3.1 判断点在多边形内部

```
bool point_on_line(const Point &p, const Point &a, const Point &b) {
1
         return sgn(det(p, a, b)) == 0 && sgn(dot(p, a, b)) <= 0;
2
3
     bool point_in_polygon(const Point &p, const std::vector<Point> &polygon) {
5
         int counter = 0;
6
         for (int i = 0; i < (int)polygon.size(); ++i) {</pre>
7
8
             Point a = polygon[i], b = polygon[(i + 1) % (int)polygon.size()];
             if (point_on_line(p, a, b)) {
9
                       Point on the boundary are excluded.
10
                 //
                 return false;
11
             }
12
             int x = sgn(det(a, p, b));
13
             int y = sgn(a.y - p.y);
14
15
             int z = sgn(b.y - p.y);
             counter += (x > 0 & y <= 0 & z > 0);
16
             counter -= (x < 0 \&\& z <= 0 \&\& y > 0);
17
         }
18
19
         return counter;
     }
20
```

#### 6.3.2 多边形内整点计数

```
int getInside(int n, Point *P) { // 求多边形 P 内有多少个整数点
1
2
         int OnEdge = n;
         double area = getArea(n, P);
3
         for (int i = 0; i < n - 1; i++) {
4
             Point now = P[i + 1] - P[i];
5
6
             int y = (int)now.y, x = (int)now.x;
             OnEdge += abs(gcd(x, y)) - 1;
         }
8
        Point now = P[0] - P[n - 1];
9
         int y = (int)now.y, x = (int)now.x;
10
         OnEdge += abs(gcd(x, y)) - 1;
11
```

CHAPTER 6. 计算几何

```
double ret = area - (double)OnEdge / 2 + 1;
return (int)ret;
}
```

#### 6.4 圆

#### 6.4.1 最小覆盖圆

```
Point getmid(Point a, Point b) {
1
         return Point((a.x + b.x) / 2, (a.y + b.y) / 2);
2
3
     Point getcross(Point a, Point vA, Point b, Point vB) {
         Point u = a - b;
5
         double t = det(vB, u) / det(vA, vB);
6
7
         return a + vA * t;
9
     Point getcir(Point a,Point b,Point c) {
         Point midA = getmid(a,b), vA = Point(-(b - a).y, (b - a).x);
10
         Point midB = getmid(b,c), vB = Point(-(c - b).y, (c - b).x);
11
         return getcross(midA, vA, midB, vB);
12
13
     double mincir(Point *p,int n) {
14
         std::random_shuffle(p + 1, p + n + 1);
15
         Point 0 = p[1];
16
         double r = 0;
17
         for (int i = 2; i <= n; i++) {
18
19
              if (dist(0, p[i]) <= r) continue;</pre>
             0 = p[i]; r = 0;
20
             for (int j = 1; j < i; j++) {
21
                  if (dist(0, p[j]) <= r) continue;</pre>
22
23
                  0 = getmid(p[i], p[j]); r = dist(0,p[i]);
                  for (int k = 1; k < j; k++) {
24
                      if (dist(0,p[k]) <= r) continue;</pre>
25
                      0 = getcir(p[i], p[j], p[k]);
26
27
                      r = dist(0,p[i]);
28
             }
29
         }
30
31
         return r;
32
```

#### 6.4.2 最小覆盖球

```
double eps(1e-8);
1
2
     int sign(const double & x) {
3
         return (x > eps) - (x + eps < 0);
4
     bool equal(const double & x, const double & y) {
5
         return x + eps > y and y + eps > x;
6
     struct Point {
8
9
         double x, y, z;
         Point() {
10
11
         Point(const double & x, const double & y, const double & z) : x(x), y(y), z(z){
12
13
         }
14
         void scan() {
             scanf("%lf%lf%lf", &x, &y, &z);
15
16
17
         double sqrlen() const {
18
             return x * x + y * y + z * z;
```

6.4. 圆

```
19
         double len() const {
20
             return sqrt(sqrlen());
21
22
         void print() const {
             printf("(\frac{1}{n} \frac{1}{n}, x, y, z);
24
25
     } a[33];
26
27
     Point operator + (const Point & a, const Point & b) {
         return Point(a.x + b.x, a.y + b.y, a.z + b.z);
28
29
     Point operator - (const Point & a, const Point & b) {
30
         return Point(a.x - b.x, a.y - b.y, a.z - b.z);
31
32
     Point operator * (const double & x, const Point & a) {
33
34
         return Point(x * a.x, x * a.y, x * a.z);
35
     double operator % (const Point & a, const Point & b) {
36
         return a.x * b.x + a.y * b.y + a.z * b.z;
37
38
     Point operator * (const Point & a, const Point & b) {
39
         return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
40
41
42
     struct Circle {
43
         double r;
         Point o;
44
         Circle() {
45
             o.x = o.y = o.z = r = 0;
46
47
         Circle(const Point & o, const double & r) : o(o), r(r) {
48
49
         void scan() {
50
             o.scan();
51
             scanf("%lf", &r);
52
53
         void print() const {
54
             o.print();
55
             printf("%lf\n", r);
56
         }
57
58
     };
     struct Plane {
59
         Point nor;
60
         double m;
61
         Plane(const Point & nor, const Point & a) : nor(nor){
62
             m = nor \% a;
63
         }
64
     };
65
     Point intersect(const Plane & a, const Plane & b, const Plane & c) {
66
         Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z,
67
         c.nor.z), c4(a.m, b.m, c.m);
         return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
68
69
     bool in(const Point & a, const Circle & b) {
70
         return sign((a - b.o).len() - b.r) <= 0;
71
72
     bool operator < (const Point & a, const Point & b) {</pre>
73
         if(!equal(a.x, b.x)) {
74
             return a.x < b.x;
75
76
         if(!equal(a.y, b.y)) {
77
             return a.y < b.y;
78
79
80
         if(!equal(a.z, b.z)) {
```

CHAPTER 6. 计算几何

```
return a.z < b.z;
81
82
          return false;
83
      }
84
      bool operator == (const Point & a, const Point & b) {
85
          return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
86
87
88
      vector<Point> vec;
89
      Circle calc() {
          if(vec.empty()) {
90
              return Circle(Point(0, 0, 0), 0);
91
          }else if(1 == (int)vec.size()) {
92
              return Circle(vec[0], 0);
93
          }else if(2 == (int)vec.size()) {
94
              return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
95
          }else if(3 == (int)vec.size()) {
96
              double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() /
97
         2 / fabs(((vec[0] - vec[2]) * (vec[1] - vec[2])).len()));
              return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
98
99
                                   Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),
                           Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
100
          }else {
101
              Point o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
102
                         Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),
103
                         Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0])));
104
              return Circle(o, (o - vec[0]).len());
105
          }
106
107
      Circle miniBall(int n) {
108
          Circle res(calc());
109
          for(int i(0); i < n; i++) {
110
              if(!in(a[i], res)) {
111
                   vec.push_back(a[i]);
112
                  res = miniBall(i);
113
114
                  vec.pop_back();
                   if(i) {
115
                       Point tmp(a[i]);
116
                       memmove(a + 1, a, sizeof(Point) * i);
117
                       a[0] = tmp;
118
119
                  }
              }
120
          }
121
122
          return res;
      }
123
      int main() {
124
          int n;
125
126
          for(;;) {
              scanf("%d", &n);
127
              if(!n) {
128
129
                  break;
130
              for(int i(0); i < n; i++) {</pre>
131
                  a[i].scan();
132
133
              sort(a, a + n);
134
135
              n = unique(a, a + n) - a;
              vec.clear();
136
              printf("%.10f\n", miniBall(n).r);
137
          }
138
      }
139
```

6.4. 圆

#### 6.4.3 多边形与圆的交面积

```
// 求扇形面积
1
     double getSectorArea(const Point &a, const Point &b, const double &r) {
2
         double c = (2.0 * r * r - sqrdist(a, b)) / (2.0 * r * r);
3
         double alpha = acos(c);
4
         return r * r * alpha / 2.0;
5
6
     // 求二次方程 ax^2 + bx + c = 0 的解
7
     std::pair<double, double> getSolution(const double &a, const double &b, const double &c) {
         double delta = b * b - 4.0 * a * c;
9
         if (dcmp(delta) < 0) return std::make_pair(0, 0);</pre>
10
         else return std::make_pair((-b - sqrt(delta)) / (2.0 * a), (-b + sqrt(delta)) / (2.0 * a));
11
12
     // 直线与圆的交点
13
     std::pair<Point, Point> getIntersection(const Point &a, const Point &b, const double &r) {
14
         Point d = b - a;
15
         double A = dot(d, d);
16
         double B = 2.0 * dot(d, a);
17
18
         double C = dot(a, a) - r * r;
19
         std::pair<double, double> s = getSolution(A, B, C);
         return std::make_pair(a + d * s.first, a + d * s.second);
20
21
     // 原点到线段 AB 的距离
22
     double getPointDist(const Point &a, const Point &b) {
23
24
         Point d = b - a;
         int sA = dcmp(dot(a, d)), sB = dcmp(dot(b, d));
25
         if (sA * sB <= 0) return det(a, b) / dist(a, b);</pre>
26
         else return std::min(dist(a), dist(b));
27
28
     // a 和 b 和原点组成的三角形与半径为 r 的圆的交的面积
29
     double getArea(const Point &a, const Point &b, const double &r) {
30
         double dA = dot(a, a), dB = dot(b, b), dC = getPointDist(a, b), ans = 0.0;
31
         if (dcmp(dA - r * r) \le 0 \&\& dcmp(dB - r * r) \le 0) return det(a, b) / 2.0;
32
         Point tA = a / dist(a) * r;
33
         Point tB = b / dist(b) * r;
34
         if (dcmp(dC - r) > 0) return getSectorArea(tA, tB, r);
35
         std::pair<Point, Point> ret = getIntersection(a, b, r);
36
         if (dcmp(dA - r * r) > 0 \&\& dcmp(dB - r * r) > 0) {
37
             ans += getSectorArea(tA, ret.first, r);
38
39
             ans += det(ret.first, ret.second) / 2.0;
             ans += getSectorArea(ret.second, tB, r);
40
             return ans;
41
         }
42
         if (dcmp(dA - r * r) > 0) return det(ret.first, b) / 2.0 + getSectorArea(tA, ret.first, r);
43
         else return det(a, ret.second) / 2.0 + getSectorArea(ret.second, tB, r);
44
45
     // 求圆与多边形的交的主过程
46
     double getArea(int n, Point *p, const Point &c, const double r) {
47
         double ret = 0.0;
48
         for (int i = 0; i < n; i++) {
49
             int sgn = dcmp(det(p[i] - c, p[(i + 1) \% n] - c));
50
51
             if (sgn > 0) ret += getArea(p[i] - c, p[(i + 1) % n] - c, r);
             else ret -= getArea(p[(i + 1) % n] - c, p[i] - c, r);
52
         }
53
         return fabs(ret);
54
    }
55
```

## Chapter 7

# 其它

## 7.1 STL 使用方法

#### 7.1.1 nth\_element

用法:  $nth_element(a + 1, a + id, a + n + 1)$ ; 作用: 将排名为 id 的元素放在第 id 个位置。

#### 7.1.2 next\_permutation

用法:  $next_permutation(a + 1, a + n + 1)$ ;

作用:以 a 中从小到大排序后为第一个排列,求得当期数组 a 中的下一个排列,返回值为当期排列是否为最后一个排列。

## 7.2 博弈论相关

#### 7.2.1 巴什博奕

- 1. 只有一堆 n 个物品,两个人轮流从这堆物品中取物,规定每次至少取一个,最多取 m 个。最后取光者得胜。
- 2. 显然,如果 n=m+1,那么由于一次最多只能取 m 个,所以,无论先取者拿走多少个,后取者都能够一次拿走剩余的物品,后者取胜。因此我们发现了如何取胜的法则: 如果 n=m+1 r+s,(r 为任意自然数, $s \le m$ ),那么先取者要拿走 s 个物品,如果后取者拿走  $k(k \le m)$  个,那么先取者再拿走 m+1-k 个,结果剩下 (m+1)(r-1) 个,以后保持这样的取法,那么先取者肯定获胜。总之,要保持给对手留下 (m+1) 的倍数,就能最后获胜。

#### 7.2.2 威佐夫博弈

- 1. 有两堆各若干个物品,两个人轮流从某一堆或同时从两堆中取同样多的物品,规定每次至少取一个,多者不限,最后取光者得胜。
- 2. 判断一个局势 (a,b) 为奇异局势(必败态)的方法:

$$a_k = [k(1+\sqrt{5})/2] b_k = a_k + k$$

#### 7.2.3 阶梯博奕

- 1. 博弈在一列阶梯上进行,每个阶梯上放着自然数个点,两个人进行阶梯博弈,每一步则是将一个阶梯上的若干个点(至少一个)移到前面去,最后没有点可以移动的人输。
- 2. 解决方法: 把所有奇数阶梯看成 N 堆石子, 做 NIM。(把石子从奇数堆移动到偶数堆可以理解为拿走石子, 就相当于几个奇数堆的石子在做 Nim)

#### 7.2.4 图上删边游戏

#### 链的删边游戏

- 1. 游戏规则:对于一条链,其中一个端点是根,两人轮流删边,脱离根的部分也算被删去,最后没边可删的人输。
- 2. 做法: sg[i] = n dist(i) 1 (其中 n 表示总点数, dist(i) 表示离根的距离)

7.3. JAVA REFERENCE 65

#### 树的删边游戏

- 1. 游戏规则:对于一棵有根树,两人轮流删边,脱离根的部分也算被删去,没边可删的人输。
- 2. 做法: 叶子结点的 sg = 0,其他节点的 sg 等于儿子结点的 sg + 1 的异或和。

#### 局部连通图的删边游戏

- 1. 游戏规则:在一个局部连通图上,两人轮流删边,脱离根的部分也算被删去,没边可删的人输。局部连通图的构图规则是,在一棵基础树上加边得到,所有形成的环保证不共用边,且只与基础树有一个公共点。
- 2. 做法:去掉所有的偶环,将所有的奇环变为长度为1的链,然后做树的删边游戏。

#### 7.3 Java Reference

```
1 import java.io.*;
 2 import java.util.*;
 3 import java.math.*;
 4
 5 public class Main {
 6
        static int get(char c) {
 7
            if (c <= '9')
 8
                return c - '0';
 9
            else if (c <= 'Z')</pre>
                return c - 'A' + 10;
10
11
            else
12
                return c - 'a' + 36;
13
14
        static char get(int x) {
1.5
            if (x <= 9)
16
                return (char) (x + '0');
17
            else if (x <= 35)
18
               return (char) (x - 10 + 'A');
19
            else
20
                return (char) (x - 36 + 'a');
21
22
        static BigInteger get(String s, BigInteger x) {
23
            BigInteger ans = BigInteger.valueOf(0), now = BigInteger.valueOf(1);
24
            for (int i = s.length() - 1; i >= 0; i---) {
2.5
                ans = ans.add(now.multiply(BigInteger.valueOf(get(s.charAt(i)))));
26
                now = now.multiply(x);
27
            }
28
            return ans;
29
30
        public static void main(String [] args) {
31
            Scanner cin = new Scanner(new BufferedInputStream(System.in));
32
            for (; ; ) {
33
                BigInteger x = cin.nextBigInteger();
34
                if (x.compareTo(BigInteger.valueOf(0)) == 0)
3.5
36
                String s = cin.next(), t = cin.next(), r = "";
37
                BigInteger ans = get(s, x).mod(get(t, x));
38
                if (ans.compareTo(BigInteger.valueOf(0)) == 0)
39
                    r = "0";
40
                for (; ans.compareTo(BigInteger.valueOf(0)) > 0;) {
41
                    r = get(ans.mod(x).intValue()) + r;
42
                    ans = ans.divide(x);
43
44
                System.out.println(r);
45
           }
46
        }
47 }
48
49 // Arrays
50 int a[];
51
   .fill(a[, int fromIndex, int toIndex],val); | .sort(a[, int fromIndex, int toIndex])
52
   // String
53 String s;
```

66 CHAPTER 7. 其它

```
54 .charAt(int i); | compareTo(String) | compareToIgnoreCase () | contains(String) |
55 length () | substring(int 1, int len)
56 // BigInteger
57
   .abs() | .add() | bitLength () | subtract () | divide () | remainder () | divideAndRemainder
       () | modPow(b, c) |
58 pow(int) | multiply () | compareTo () |
59 gcd() | intValue () | longValue () | isProbablePrime(int c) (1 - 1/2^c) |
60 nextProbablePrime () | shiftLeft(int) | valueOf ()
61 // BigDecimal
62 .ROUND_CEILING | ROUND_DOWN_FLOOR | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_HALF_UP |
       ROUND UP
63 .divide(BigDecimal b, int scale , int round mode) | doubleValue () | movePointLeft(int) | pow(
64 setScale(int scale , int round_mode) | stripTrailingZeros ()
65 // StringBuilder
66 StringBuilder sb = new StringBuilder ();
67 sb.append(elem) | out.println(sb)
```

## Chapter 8

# 数学公式

## 8.1 常用数学公式

### 8.1.1 求和公式

1. 
$$\sum_{k=1}^{n} (2k-1)^2 = \frac{n(4n^2-1)}{3}$$

2. 
$$\sum_{k=1}^{n} k^3 = \left[\frac{n(n+1)}{2}\right]^2$$

3. 
$$\sum_{k=1}^{n} (2k-1)^3 = n^2(2n^2-1)$$

4. 
$$\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

5. 
$$\sum_{k=1}^{n} k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

6. 
$$\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$$

7. 
$$\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$$

8. 
$$\sum_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$$

#### 8.1.2 斐波那契数列

1. 
$$fib_0 = 0, fib_1 = 1, fib_n = fib_{n-1} + fib_{n-2}$$

2. 
$$fib_{n+2} \cdot fib_n - fib_{n+1}^2 = (-1)^{n+1}$$

3. 
$$fib_{-n} = (-1)^{n-1} fib_n$$

4. 
$$fib_{n+k} = fib_k \cdot fib_{n+1} + fib_{k-1} \cdot fib_n$$

5. 
$$gcd(fib_m, fib_n) = fib_{gcd(m,n)}$$

6. 
$$fib_m|fib_n^2 \Leftrightarrow nfib_n|m$$

## 8.1.3 错排公式

1. 
$$D_n = (n-1)(D_{n-2} - D_{n-1})$$

2. 
$$D_n = n! \cdot \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \dots + \frac{(-1)^n}{n!}\right)$$

#### 8.1.4 莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & \text{若}n = 1 \\ (-1)^k & \text{若}n$$
天平方数因子,且 $n = p_1 p_2 \dots p_k \\ 0 & \text{若}n$ 有大于1的平方数因数 
$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \text{若}n = 1 \\ 0 & \text{其他情况} \end{cases}$$
 
$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$$

$$g(x) = \sum_{n=1}^{[x]} f(\frac{x}{n}) \Leftrightarrow f(x) = \sum_{n=1}^{[x]} \mu(n)g(\frac{x}{n})$$

#### 8.1.5 Burnside 引理

设 G 是一个有限群,作用在集合 X 上。对每个 g 属于 G,令  $X^g$  表示 X 中在 g 作用下的不动元素,轨道数(记作 |X/G|)由如下公式给出:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|.$$

#### 8.1.6 五边形数定理

设 p(n) 是 n 的拆分数,有

$$p(n) = \sum_{k \in \mathbb{Z} \setminus \{0\}} (-1)^{k-1} p\left(n - \frac{k(3k-1)}{2}\right)$$

#### 8.1.7 树的计数

1. 有根树计数: n+1 个结点的有根树的个数为

$$a_{n+1} = \frac{\sum_{j=1}^{n} j \cdot a_j \cdot S_{n,j}}{n}$$

其中,

$$S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$$

2. 无根树计数: 当 n 为奇数时, n 个结点的无根树的个数为

$$a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$$

当 n 为偶数时, n 个结点的无根树的个数为

$$a_n - \sum_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1)$$

3. n 个结点的完全图的生成树个数为

$$n^{n-1}$$

4. 矩阵 - 树定理:图 G 由 n 个结点构成,设 A[G] 为图 G 的邻接矩阵、D[G] 为图 G 的度数矩阵,则图 G 的不同生成树的个数为 C[G] = D[G] - A[G] 的任意一个 n-1 阶主子式的行列式值。

#### 8.1.8 欧拉公式

平面图的顶点个数、边数和面的个数有如下关系:

$$V - E + F = C + 1$$

其中,V 是顶点的数目,E 是边的数目,F 是面的数目,C 是组成图形的连通部分的数目。当图是单连通图的时候,公式简化为:

$$V - E + F = 2$$

#### 8.1.9 皮克定理

给定顶点坐标均是整点(或正方形格点)的简单多边形,其面积 A 和内部格点数目 i、边上格点数目 b 的关系:

$$A = i + \frac{b}{2} - 1$$

#### 8.1.10 牛顿恒等式

设

$$\prod_{i=1}^{n} (x - x_i) = a_n + a_{n-1}x + \dots + a_1x^{n-1} + a_0x^n$$

$$p_k = \sum_{i=1}^{n} x_i^k$$

则

$$a_0p_k + a_1p_{k-1} + \dots + a_{k-1}p_1 + ka_k = 0$$

特别地,对于

$$|\mathbf{A} - \lambda \mathbf{E}| = (-1)^n (a_n + a_{n-1}\lambda + \dots + a_1\lambda^{n-1} + a_0\lambda^n)$$

有

$$p_k = Tr(\boldsymbol{A}^k)$$

## 8.2 平面几何公式

#### 8.2.1 三角形

1. 半周长

$$p = \frac{a+b+c}{2}$$

2. 面积

$$S = \frac{a \cdot H_a}{2} = \frac{ab \cdot sinC}{2} = \sqrt{p(p-a)(p-b)(p-c)}$$

3. 中线

$$M_a = \frac{\sqrt{2(b^2 + c^2) - a^2}}{2} = \frac{\sqrt{b^2 + c^2 + 2bc \cdot cosA}}{2}$$

4. 角平分线

$$T_a = \frac{\sqrt{bc \cdot [(b+c)^2 - a^2]}}{b+c} = \frac{2bc}{b+c} cos \frac{A}{2}$$

5. 高线

$$H_a = bsinC = csinB = \sqrt{b^2 - (\frac{a^2 + b^2 - c^2}{2a})^2}$$

6. 内切圆半径

$$\begin{split} r &= \frac{S}{p} = \frac{arcsin\frac{B}{2} \cdot sin\frac{C}{2}}{sin\frac{B+C}{2}} = 4R \cdot sin\frac{A}{2}sin\frac{B}{2}sin\frac{C}{2} \\ &= \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \cdot tan\frac{A}{2}tan\frac{B}{2}tan\frac{C}{2} \end{split}$$

7. 外接圆半径

$$R = \frac{abc}{4S} = \frac{a}{2sinA} = \frac{b}{2sinB} = \frac{c}{2sinC}$$

#### 8.2.2 四边形

 $D_1, D_2$  为对角线, M 对角线中点连线, A 为对角线夹角, p 为半周长

1. 
$$a^2 + b^2 + c^2 + d^2 = D_1^2 + D_2^2 + 4M^2$$

- 2.  $S = \frac{1}{2}D_1D_2sinA$
- 3. 对于圆内接四边形

$$ac + bd = D_1D_2$$

4. 对于圆内接四边形

$$S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$$

#### 8.2.3 正 n 边形

R 为外接圆半径, r 为内切圆半径

1. 中心角

$$A = \frac{2\pi}{n}$$

2. 内角

$$C = \frac{n-2}{n}\pi$$

3. 边长

$$a=2\sqrt{R^2-r^2}=2R\cdot sin\frac{A}{2}=2r\cdot tan\frac{A}{2}$$

4. 面积

$$S = \frac{nar}{2} = nr^2 \cdot tan\frac{A}{2} = \frac{nR^2}{2} \cdot sinA = \frac{na^2}{4 \cdot tan\frac{A}{2}}$$

#### 8.2.4 圆

1. 弧长

$$l = rA$$

2. 弦长

$$a = 2\sqrt{2hr - h^2} = 2r \cdot \sin\frac{A}{2}$$

3. 弓形高

$$h=r-\sqrt{r^2-\frac{a^2}{4}}=r(1-\cos\frac{A}{2})=\frac{1}{2}\cdot arctan\frac{A}{4}$$

4. 扇形面积

$$S_1 = \frac{rl}{2} = \frac{r^2 A}{2}$$

5. 弓形面积

$$S_2 = \frac{rl - a(r - h)}{2} = \frac{r^2}{2}(A - sinA)$$

#### 8.2.5 棱柱

1. 体积

$$V = Ah$$

A 为底面积,h 为高

2. 侧面积

$$S = lp$$

l 为棱长, p 为直截面周长

3. 全面积

$$T = S + 2A$$

#### 8.2.6 棱锥

1. 体积

$$V = Ah$$

A 为底面积, h 为高

2. 正棱锥侧面积

$$S = lp$$

l 为棱长, p 为直截面周长

3. 正棱锥全面积

$$T = S + 2A$$

## 8.2.7 棱台

1. 体积

$$V = (A_1 + A_2 + \sqrt{A_1 A_2}) \cdot \frac{h}{3}$$

 $A_1, A_2$  为上下底面积, h 为高

2. 正棱台侧面积

$$S = \frac{p_1 + p_2}{2}l$$

 $p_1, p_2$  为上下底面周长, l 为斜高

3. 正棱台全面积

$$T = S + A_1 + A_2$$

#### 8.2.8 圆柱

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

$$T = 2\pi r(h+r)$$

3. 体积

$$V = \pi r^2 h$$

## 8.2.9 圆锥

1. 母线

$$l = \sqrt{h^2 + r^2}$$

2. 侧面积

$$S = \pi r l$$

3. 全面积

$$T = \pi r(l+r)$$

4. 体积

$$V = \frac{\pi}{3}r^2h$$

### 8.2.10 圆台

1. 母线

$$l = \sqrt{h^2 + (r_1 - r_2)^2}$$

2. 侧面积

$$S = \pi(r_1 + r_2)l$$

3. 全面积

$$T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$$

4. 体积

$$V = \frac{\pi}{3}(r_1^2 + r_2^2 + r_1r_2)h$$

#### 8.2.11 球

1. 全面积

$$T = 4\pi r^2$$

2. 体积

$$V = \frac{4}{3}\pi r^3$$

#### 8.2.12 球台

1. 侧面积

$$S = 2\pi rh$$

2. 全面积

$$T = \pi(2rh + r_1^2 + r_2^2)$$

3. 体积

$$V = \frac{\pi h[3(r_1^2 + r_2^2) + h^2]}{6}$$

#### 8.2.13 球扇形

1. 全面积

$$T = \pi r (2h + r_0)$$

h 为球冠高,  $r_0$  为球冠底面半径

2. 体积

$$V = \frac{2}{3}\pi r^2 h$$

## 8.3 立体几何公式

#### 8.3.1 球面三角公式

设 a,b,c 是边长,A,B,C 是所对的二面角,有余弦定理

$$cosa = cosb \cdot cosc + sinb \cdot sinc \cdot cosA$$

正弦定理

$$\frac{sinA}{sina} = \frac{sinB}{sinb} = \frac{sinC}{sinc}$$

三角形面积是  $A + B + C - \pi$ 

#### 8.3.2 四面体体积公式

U, V, W, u, v, w 是四面体的 6 条棱, U, V, W 构成三角形, (U, u), (V, v), (W, w) 互为对棱, 则

$$V = \frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw}$$

其中

$$\begin{cases} a &= \sqrt{xYZ}, \\ b &= \sqrt{yZX}, \\ c &= \sqrt{zXY}, \\ d &= \sqrt{xyz}, \\ s &= a+b+c+d, \\ X &= (w-U+v)(U+v+w), \\ x &= (U-v+w)(v-w+U), \\ Y &= (u-V+w)(V+w+u), \\ y &= (V-w+u)(w-u+V), \\ Z &= (v-W+u)(W+u+v), \\ z &= (W-u+v)(u-v+W) \end{cases}$$