

# ACM 模板

Wajov

September 2, 2017

## 目录

<b>1</b>	<b>字符串算法</b>	<b>3</b>
1.1	Manacher . . . . .	3
1.2	Knuth–Morris–Pratt . . . . .	3
1.3	扩展 Knuth–Morris–Pratt . . . . .	4
1.4	Aho–Corasick . . . . .	4
1.5	后缀数组 . . . . .	5
<b>2</b>	<b>图算法</b>	<b>5</b>
2.1	拓扑排序 . . . . .	5
2.2	Floyd–Warshall . . . . .	6
2.3	Floyd–Warshall (最小环) . . . . .	6
2.4	Bellman–Ford+ 队列 . . . . .	7
2.5	Dijkstra+ 堆 . . . . .	8
2.6	Kruskal . . . . .	9
2.7	Prim+ 堆 . . . . .	9
2.8	Tarjan (强连通分量) . . . . .	10
2.9	Tarjan (点双连通分量) . . . . .	11
2.10	Tarjan (边双连通分量) . . . . .	12
2.11	匈牙利 . . . . .	13
2.12	Kuhn–Munkres . . . . .	13
2.13	Dinic . . . . .	14
2.14	Edmonds–Karp (最小费用最大流) . . . . .	16
<b>3</b>	<b>树算法</b>	<b>17</b>
3.1	Tarjan (最近公共祖先) . . . . .	17
3.2	树链剖分 . . . . .	17
<b>4</b>	<b>数据结构</b>	<b>18</b>
4.1	字母树 . . . . .	18
4.2	并查集 . . . . .	19
4.3	树状数组 . . . . .	19
4.4	张昆玮线段树 . . . . .	20
4.5	线段树 . . . . .	20
4.6	伸展树 . . . . .	21
4.7	伸展树 (区间) . . . . .	24
4.8	节点大小平衡树 . . . . .	26
<b>5</b>	<b>数论</b>	<b>28</b>
5.1	快速幂 . . . . .	28
5.2	Euclid . . . . .	28
5.3	扩展 Euclid . . . . .	29
5.4	Euler 筛 . . . . .	29

<b>6</b>	<b>计算几何</b>	<b>30</b>
6.1	线段相交 . . . . .	30
6.2	多边形面积 . . . . .	30
6.3	Graham 扫描 . . . . .	31
6.4	最小圆覆盖 . . . . .	31

# 1 字符串算法

## 1.1 Manacher

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1000001;
4 int n, ans, p[N << 1];
5 char c, s[N << 1];
6 int main()
7 {
8     s[0] = '$';
9     while ((c = getchar()) != '\n')
10     {
11         s[++n] = '#';
12         s[++n] = c;
13     }
14     s[++n] = '#';
15     for (int i = 1, j = 0; i <= n; i++)
16     {
17         p[i] = i < j + p[j] ? min(p[(j << 1) - i], j + p[j] - i) : 1;
18         while (s[i + p[i]] == s[i - p[i]])
19             p[i]++;
20         if (i + p[i] > j + p[j])
21             j = i;
22         ans = max(ans, p[i] - 1);
23     }
24     printf("%d\n", ans);
25     return 0;
26 }
```

## 1.2 Knuth–Morris–Pratt

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1000001;
4 int n, m, num, p[N], ans[N];
5 char a[N + 10], b[N + 10];
6 int main()
7 {
8     scanf("%s%s", a + 1, b + 1);
9     n = strlen(a + 1);
10    m = strlen(b + 1);
11    for (int i = 2, j = 0; i <= m; i++)
12    {
13        for (; j > 0 && b[j + 1] != b[i]; j = p[j]);
14        if (b[j + 1] == b[i])
15            j++;
16        p[i] = j;
17    }
18    for (int i = 1, j = 0; i <= n; i++)
19    {
20        for (; j > 0 && b[j + 1] != a[i]; j = p[j]);
21        if (b[j + 1] == a[i])
22            j++;
23        if (j == m)
24        {
25            ans[++num] = i - j + 1;
26            j = p[j];
27        }
28    }
29    for (int i = 1; i < num; i++)
30        printf("%d_", ans[i]);
31    printf("%d\n", ans[num]);
32    return 0;
33 }
```

### 1.3 扩展 Knuth–Morris–Pratt

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1000001;
4 int n, m, p[N], ex[N];
5 char a[N + 10], b[N + 10];
6 int main()
7 {
8     scanf("%s%s", a + 1, b + 1);
9     n = strlen(a + 1);
10    m = strlen(b + 1);
11    for (int i = 2, j = 0; i <= m; i++)
12    {
13        p[i] = i < j + p[j] ? min(p[i - j + 1], j + p[j] - i) : 0;
14        for (; i + p[i] <= m && b[i + p[i]] == b[p[i] + 1]; p[i]++);
15        if (i + p[i] > j + p[j])
16            j = i;
17    }
18    for (int i = 1, j = 0; i <= n; i++)
19    {
20        ex[i] = i <= j + ex[j] ? min(p[i - j + 1], j + ex[j] - i) : 0;
21        for (; i + ex[i] <= n && ex[i] < m && a[i + ex[i]] == b[ex[i] + 1]; ex[i]++);
22        if (i + ex[i] > j + ex[j])
23            j = i;
24    }
25    for (int i = 1; i < n; i++)
26        printf("%d_", ex[i]);
27    printf("%d\n", ex[n]);
28    return 0;
29 }
```

### 1.4 Aho–Corasick

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1000001;
4 int n, t, tmp, now, pos, ans, son[N][26], num[N], p[N];
5 char a[N + 10], b[N + 10];
6 queue<int> q;
7 void Insert(char s[])
8 {
9     int t = 1, tmp;
10    for (int i = 0; s[i]; i++)
11    {
12        tmp = s[i] - 97;
13        if (!son[t][tmp])
14            son[t][tmp] = ++pos;
15        t = son[t][tmp];
16    }
17    num[t]++;
18 }
19 int main()
20 {
21     pos = 1;
22     scanf("%s%d", a, &n);
23     for (int i = 1; i <= n; i++)
24     {
25         scanf("%s", b);
26         Insert(b);
27     }
28     q.push(1);
29     while (!q.empty())
30     {
31         now = q.front();
32         q.pop();
33         for (int i = 0; i < 26; i++)
34             if (son[now][i])
35             {
36                 for (t = p[now]; t > 0 && son[t][i] == 0; t = p[t]);
```

```

37         p[son[now][i]] = t ? son[t][i] : 1;
38         q.push(son[now][i]);
39     }
40 }
41 t = 1;
42 for (int i = 0; a[i]; i++)
43 {
44     tmp = a[i] - 97;
45     for (; t > 0 && son[t][tmp] == 0; t = p[t]);
46     t = t ? son[t][tmp] : 1;
47     for (int j = t; j > 1 && num[j] > -1; j = p[j])
48     {
49         ans += num[j];
50         num[j] = -1;
51     }
52 }
53 printf("%d\n", ans);
54 return 0;
55 }

```

## 1.5 后缀数组

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int n, a[N], b[N], sum[N], tmp[N], id[N], rk[N + 10];
5  char s[N + 10];
6  void Sort(int a[], int m)
7  {
8      memset(sum, 0, sizeof(sum));
9      for (int i = 1; i <= n; i++)
10         sum[a[i]]++;
11     for (int i = 1; i <= m; i++)
12         sum[i] += sum[i - 1];
13     for (int i = n; i; i--)
14         tmp[id[i]] = sum[a[id[i]]]--;
15     for (int i = 1; i <= n; i++)
16         id[tmp[i]] = i;
17 }
18 int main()
19 {
20     scanf("%s", s + 1);
21     n = strlen(s + 1);
22     for (int i = 1; i <= n; i++)
23         a[id[i] = i] = s[i] - 97;
24     Sort(a, 25);
25     for (int i = 1; i <= n; i <= 1)
26     {
27         for (int j = 1, t = 0; j <= n; j++)
28             rk[id[j]] = a[id[j]] == a[id[j - 1]] && b[id[j]] == b[id[j - 1]] ? t : ++t;
29         for (int j = 1; j <= n; j++)
30         {
31             a[j] = rk[j];
32             b[j] = rk[min(i + j, n + 1)];
33         }
34         Sort(b, n);
35         Sort(a, n);
36     }
37     for (int i = 1; i < n; i++)
38         printf("%d_", rk[i]);
39     printf("%d\n", rk[n]);
40     return 0;
41 }

```

## 2 图算法

### 2.1 拓扑排序

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001, M = 1000001;
4  int n, m, u, v, tot, num, Head[N], Next[M], Link[M], ans[N];
5  bool flag[N];
6  inline void AddEdge(int u, int v)
7  {
8      Next[++tot] = Head[u];
9      Link[tot] = v;
10     Head[u] = tot;
11 }
12 void DFS(int x)
13 {
14     flag[x] = true;
15     for (int i = Head[x], j; i; i = Next[i])
16         if (!flag[j = Link[i]])
17             DFS(j);
18     ans[++num] = x;
19 }
20 int main()
21 {
22     scanf("%d%d", &n, &m);
23     for (int i = 1; i <= m; i++)
24     {
25         scanf("%d%d", &u, &v);
26         AddEdge(u, v);
27     }
28     for (int i = 1; i <= n; i++)
29         if (!flag[i])
30             DFS(i);
31     for (int i = n; i > 1; i--)
32         printf("%d_", ans[i]);
33     printf("%d\n", ans[1]);
34     return 0;
35 }

```

## 2.2 Floyd-Warshall

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 101;
4  int n, m, u, v, c, d[N][N];
5  int main()
6  {
7      scanf("%d%d", &n, &m);
8      for (int i = 1; i <= n; i++)
9          for (int j = 1; j <= n; j++)
10             d[i][j] = i == j ? 0 : INT_MAX >> 1;
11     for (int i = 1; i <= m; i++)
12     {
13         scanf("%d%d%d", &u, &v, &c);
14         d[u][v] = d[v][u] = min(d[u][v], c);
15     }
16     for (int k = 1; k <= n; k++)
17         for (int i = 1; i <= n; i++)
18             for (int j = 1; j <= n; j++)
19                 d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
20     for (int i = 1; i <= n; i++)
21     {
22         for (int j = 1; j < n; j++)
23             printf("%d_", d[i][j] == INT_MAX >> 1 ? -1 : d[i][j]);
24         printf("%d\n", d[i][n] == INT_MAX >> 1 ? -1 : d[i][n]);
25     }
26     return 0;
27 }

```

## 2.3 Floyd-Warshall (最小环)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 101;
4  int n, m, u, v, c, t, num, Min, a[N][N], d[N][N], p[N][N], ans[N];
5  int main()
6  {
7      scanf("%d%d", &n, &m);
8      for (int i = 1; i <= n; i++)
9          for (int j = 1; j <= n; j++)
10             {
11                 a[i][j] = i == j ? 0 : INT_MAX / 3;
12                 p[i][j] = i;
13             }
14      for (int i = 1; i <= m; i++)
15          {
16              scanf("%d%d%d", &u, &v, &c);
17              a[u][v] = a[v][u] = min(a[u][v], c);
18          }
19      memcpy(d, a, sizeof(d));
20      Min = INT_MAX / 3;
21      for (int k = 1; k <= n; k++)
22          {
23              for (int i = 1; i < k; i++)
24                  for (int j = 1; j < i; j++)
25                      if (d[i][j] + a[i][k] + a[k][j] < Min)
26                          {
27                              Min = d[i][j] + a[i][k] + a[k][j];
28                              for (num = 0, t = j; t != i; t = p[i][t])
29                                  ans[++num] = t;
30                              ans[++num] = i;
31                              ans[++num] = k;
32                          }
33              for (int i = 1; i <= n; i++)
34                  for (int j = 1; j <= n; j++)
35                      if (d[i][k] + d[k][j] < d[i][j])
36                          {
37                              d[i][j] = d[i][k] + d[k][j];
38                              p[i][j] = p[k][j];
39                          }
40          }
41      printf("%d\n", Min);
42      for (int i = 1; i < num; i++)
43          printf("%d_", ans[i]);
44      printf("%d\n", ans[num]);
45      return 0;
46 }

```

## 2.4 Bellman-Ford+ 队列

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001, M = 100001;
4  int n, m, s, u, v, c, now, tot, Head[N], Next[M << 1], Link[M << 1], Cost[M << 1], d[N];
5  bool flag[N];
6  queue<int> q;
7  inline void AddEdge(int u, int v, int c)
8  {
9      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Cost[tot] = c;
12     Head[u] = tot;
13 }
14 int main()
15 {
16     scanf("%d%d%d", &n, &m, &s);
17     for (int i = 1; i <= m; i++)
18         {
19             scanf("%d%d%d", &u, &v, &c);
20             AddEdge(u, v, c);
21             AddEdge(v, u, c);

```

```

22     }
23     for (int i = 1; i <= n; i++)
24         d[i] = INT_MAX;
25     d[s] = 0;
26     q.push(s);
27     flag[s] = true;
28     while (!q.empty())
29     {
30         now = q.front();
31         q.pop();
32         flag[now] = false;
33         for (int i = Head[now], j; i; i = Next[i])
34             if (d[now] + Cost[i] < d[j = Link[i]])
35             {
36                 d[j] = d[now] + Cost[i];
37                 if (!flag[j])
38                 {
39                     q.push(j);
40                     flag[j] = true;
41                 }
42             }
43     }
44     for (int i = 1; i < n; i++)
45         printf("%d_", d[i]);
46     printf("%d\n", d[n]);
47     return 0;
48 }

```

## 2.5 Dijkstra+ 堆

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001, M = 100001;
4  int n, m, s, u, v, c, now, tot, d[N], Head[N], Next[M << 1], Link[M << 1], Cost[M << 1];
5  bool flag[N];
6  priority_queue<pair<int, int> > q;
7  inline void AddEdge(int u, int v, int c)
8  {
9      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Cost[tot] = c;
12     Head[u] = tot;
13 }
14 int main()
15 {
16     scanf("%d%d%d", &n, &m, &s);
17     for (int i = 1; i <= m; i++)
18     {
19         scanf("%d%d%d", &u, &v, &c);
20         AddEdge(u, v, c);
21         AddEdge(v, u, c);
22     }
23     for (int i = 1; i <= n; i++)
24         d[i] = INT_MAX;
25     q.push(make_pair(d[s] = 0, s));
26     while (!q.empty())
27     {
28         now = q.top().second;
29         q.pop();
30         if (flag[now])
31             continue;
32         flag[now] = true;
33         for (int i = Head[now], j; i; i = Next[i])
34             if (d[now] + Cost[i] < d[j = Link[i]])
35             {
36                 d[j] = d[now] + Cost[i];
37                 q.push(make_pair(-d[j], j));
38             }
39     }
40     for (int i = 1; i < n; i++)
41         printf("%d_", d[i] == INT_MAX ? -1 : d[i]);

```



```

42     printf("%d\n", d[n] == INT_MAX ? -1 : d[n]);
43     return 0;
44 }

```

## 2.6 Kruskal

```

1  #include <bits/stdc++.h>
2  #define fi first
3  #define se second
4  using namespace std;
5  const int N = 100001;
6  int n, m, x, y, ans, a[N];
7  pair<int, pair<int, int> > e[N];
8  int Get(int x)
9  {
10     if (a[x] != x)
11         a[x] = Get(a[x]);
12     return a[x];
13 }
14 int main()
15 {
16     scanf("%d%d", &n, &m);
17     for (int i = 1; i <= m; i++)
18         scanf("%d%d%d", &e[i].se.fi, &e[i].se.se, &e[i].fi);
19     for (int i = 1; i <= n; i++)
20         a[i] = i;
21     sort(e + 1, e + m + 1);
22     for (int i = 1; i <= m; i++)
23     {
24         x = Get(e[i].se.fi);
25         y = Get(e[i].se.se);
26         if (x != y)
27         {
28             a[x] = y;
29             ans += e[i].fi;
30         }
31     }
32     printf("%d\n", ans);
33     return 0;
34 }

```

## 2.7 Prim+ 堆

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001, M = 100001;
4  int n, m, s, u, v, c, now, ans, tot, Head[N], Next[M << 1], Link[M << 1], Cost[M << 1], d[N];
5  bool flag[N];
6  priority_queue<pair<int, int> > q;
7  inline void AddEdge(int u, int v, int c)
8  {
9     Next[++tot] = Head[u];
10    Link[tot] = v;
11    Cost[tot] = c;
12    Head[u] = tot;
13 }
14 int main()
15 {
16     scanf("%d%d", &n, &m);
17     for (int i = 1; i <= m; i++)
18     {
19         scanf("%d%d%d", &u, &v, &c);
20         AddEdge(u, v, c);
21         AddEdge(v, u, c);
22     }
23     for (int i = 1; i <= n; i++)
24         d[i] = INT_MAX;
25     q.push(make_pair(d[1] = 0, 1));

```

```

26     while (!q.empty())
27     {
28         now = q.top().second;
29         q.pop();
30         if (flag[now])
31             continue;
32         ans += d[now];
33         flag[now] = true;
34         for (int i = Head[now], j; i; i = Next[i])
35             if (Cost[i] < d[j = Link[i]])
36             {
37                 d[j] = Cost[i];
38                 q.push(make_pair(-d[j], j));
39             }
40     }
41     printf("%d\n", ans);
42     return 0;
43 }

```

## 2.8 Tarjan (强连通分量)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001, M = 1000001;
4  int n, m, u, v, tot, num, idx, Head[N], Next[M], Link[M], dfn[N], low[N];
5  bool flag[N];
6  stack<int> s;
7  vector<int> sub[N];
8  inline void AddEdge(int u, int v)
9  {
10     Next[++tot] = Head[u];
11     Link[tot] = v;
12     Head[u] = tot;
13 }
14 void DFS(int x)
15 {
16     s.push(x);
17     flag[x] = true;
18     low[x] = dfn[x] = ++idx;
19     for (int i = Head[x], j; i; i = Next[i])
20         if (!dfn[j = Link[i]])
21         {
22             DFS(j);
23             low[x] = min(low[x], low[j]);
24         }
25         else if (flag[j])
26             low[x] = min(low[x], dfn[j]);
27     if (low[x] == dfn[x])
28     {
29         int t;
30         num++;
31         do
32         {
33             t = s.top();
34             s.pop();
35             flag[t] = false;
36             sub[num].push_back(t);
37         }
38         while (t != x);
39     }
40 }
41 int main()
42 {
43     scanf("%d%d", &n, &m);
44     for (int i = 1; i <= m; i++)
45     {
46         scanf("%d%d", &u, &v);
47         AddEdge(u, v);
48     }
49     for (int i = 1; i <= n; i++)
50         if (!dfn[i])

```

```

51     DFS(i);
52     printf("%d\n", num);
53     for (int i = 1; i <= num; i++)
54     {
55         for (int j = 0; j < sub[i].size() - 1; j++)
56             printf("%d_", sub[i][j]);
57         printf("%d\n", sub[i][sub[i].size() - 1]);
58     }
59     return 0;
60 }

```

## 2.9 Tarjan (点双连通分量)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001, M = 1000001;
4  int n, m, u, v, tot, num, idx, Head[N], Next[M << 1], Link[M << 1], dfn[N], low[N];
5  bool flag[N];
6  stack<int> s;
7  vector<int> sub[N];
8  inline void AddEdge(int u, int v)
9  {
10     Next[++tot] = Head[u];
11     Link[tot] = v;
12     Head[u] = tot;
13 }
14 void DFS(int x, int y)
15 {
16     s.push(x);
17     flag[x] = true;
18     low[x] = dfn[x] = ++idx;
19     for (int i = Head[x]; i; i = Next[i])
20     {
21         if ((j = Link[i]) == y)
22             continue;
23         if (!dfn[j])
24         {
25             DFS(j, x);
26             low[x] = min(low[x], low[j]);
27         }
28         else if (flag[j])
29             low[x] = min(low[x], dfn[j]);
30     }
31     if (x != y && low[x] >= dfn[y])
32     {
33         int t;
34         num++;
35         do
36         {
37             t = s.top();
38             s.pop();
39             flag[t] = false;
40             sub[num].push_back(t);
41         }
42         while (t != y);
43         s.push(y);
44         flag[y] = true;
45     }
46 }
47 int main()
48 {
49     scanf("%d%d", &n, &m);
50     for (int i = 1; i <= m; i++)
51     {
52         scanf("%d%d", &u, &v);
53         AddEdge(u, v);
54         AddEdge(v, u);
55     }
56     for (int i = 1; i <= n; i++)
57         if (!dfn[i])
58             DFS(i, i);

```

```

59         DFS(i, i);
60         s.pop();
61         flag[i] = false;
62     }
63     printf("%d\n", num);
64     for (int i = 1; i <= num; i++)
65     {
66         for (int j = 0; j < sub[i].size() - 1; j++)
67             printf("%d_", sub[i][j]);
68         printf("%d\n", sub[i][sub[i].size() - 1]);
69     }
70     return 0;
71 }

```

## 2.10 Tarjan (边双连通分量)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001, M = 1000001;
4  int n, m, u, v, tot, num, idx, Head[N], Next[M << 1], Link[M << 1], dfn[M << 1], low[N];
5  bool flag[N];
6  stack<int> s;
7  vector<int> sub[N];
8  inline void AddEdge(int u, int v)
9  {
10     Next[++tot] = Head[u];
11     Link[tot] = v;
12     Head[u] = tot;
13 }
14 void DFS(int x, int y)
15 {
16     s.push(x);
17     flag[x] = true;
18     low[x] = dfn[x] = ++idx;
19     for (int i = Head[x], j; i; i = Next[i])
20     {
21         if ((j = Link[i]) == y)
22             continue;
23         if (!dfn[j])
24         {
25             DFS(j, x);
26             low[x] = min(low[x], low[j]);
27         }
28         else if (flag[j])
29             low[x] = min(low[x], dfn[j]);
30     }
31     if (low[x] > dfn[y])
32     {
33         int t;
34         num++;
35         do
36         {
37             t = s.top();
38             s.pop();
39             flag[t] = false;
40             sub[num].push_back(t);
41         }
42         while (t != x);
43     }
44 }
45 int main()
46 {
47     scanf("%d%d", &n, &m);
48     for (int i = 1; i <= m; i++)
49     {
50         scanf("%d%d", &u, &v);
51         AddEdge(u, v);
52         AddEdge(v, u);
53     }
54     for (int i = 1; i <= n; i++)
55         if (!dfn[i])

```

```

56     {
57         DFS(i, i);
58         num++;
59         while (!s.empty())
60         {
61             flag[s.top()] = false;
62             sub[num].push_back(s.top());
63             s.pop();
64         }
65     }
66     printf("%d\n", num);
67     for (int i = 1; i <= num; i++)
68     {
69         for (int j = 0; j < sub[i].size() - 1; j++)
70             printf("%d_", sub[i][j]);
71         printf("%d\n", sub[i][sub[i].size() - 1]);
72     }
73     return 0;
74 }

```

## 2.11 匈牙利

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1001, M = 10001;
4  int n, m, k, u, v, tot, ans, Head[N], Next[M], Link[M], p[N];
5  bool flag[N];
6  inline void AddEdge(int u, int v)
7  {
8      Next[++tot] = Head[u];
9      Link[tot] = v;
10     Head[u] = tot;
11 }
12 bool DFS(int x)
13 {
14     for (int i = Head[x], j; i; i = Next[i])
15         if (!flag[j = Link[i]])
16         {
17             flag[j] = true;
18             if (p[j] == 0 || DFS(p[j]))
19             {
20                 p[j] = x;
21                 return true;
22             }
23         }
24     return false;
25 }
26 int main()
27 {
28     scanf("%d%d%d", &n, &m, &k);
29     for (int i = 1; i <= k; i++)
30     {
31         scanf("%d%d", &u, &v);
32         AddEdge(u, v);
33     }
34     for (int i = 1; i <= n; i++)
35     {
36         memset(flag, false, sizeof(flag));
37         if (DFS(i))
38             ans++;
39     }
40     printf("%d\n", ans);
41     return 0;
42 }

```

## 2.12 Kuhn-Munkres

```

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

```

```

2  using namespace std;
3  const int N = 101;
4  int n, m, t, ans, a[N][N], lx[N], ly[N], slack[N], p[N];
5  bool fx[N], fy[N];
6  bool DFS(int x)
7  {
8      fx[x] = true;
9      for (int i = 1; i <= m; i++)
10         if (!fy[i])
11             {
12                 t = lx[x] + ly[i] - a[x][i];
13                 if (!t)
14                     {
15                         fy[i] = true;
16                         if (p[i] == 0 || DFS(p[i]))
17                             {
18                                 p[i] = x;
19                                 return true;
20                             }
21                     }
22                 else
23                     slack[i] = min(slack[i], lx[x] + ly[i] - a[x][i]);
24             }
25         return false;
26 }
27 bool Find(int x)
28 {
29     memset(fx, false, sizeof(fx));
30     memset(fy, false, sizeof(fy));
31     return DFS(x);
32 }
33 int main()
34 {
35     scanf("%d%d", &n, &m);
36     for (int i = 1; i <= n; i++)
37         for (int j = 1; j <= m; j++)
38             scanf("%d", &a[i][j]);
39     for (int i = 1; i <= n; i++)
40     {
41         lx[i] = INT_MIN;
42         for (int j = 1; j <= m; j++)
43             lx[i] = max(lx[i], a[i][j]);
44     }
45     for (int i = 1; i <= n; i++)
46     {
47         for (int j = 1; j <= m; j++)
48             slack[j] = INT_MAX;
49         while (!Find(i))
50         {
51             t = INT_MAX;
52             for (int j = 1; j <= m; j++)
53                 if (!fy[j])
54                     t = min(t, slack[j]);
55             for (int j = 1; j <= n; j++)
56                 if (fx[j])
57                     lx[j] -= t;
58             for (int j = 1; j <= m; j++)
59                 if (fy[j])
60                     ly[j] += t;
61             else
62                 slack[j] -= t;
63         }
64     }
65     for (int i = 1; i <= m; i++)
66         if (p[i])
67             ans += a[p[i]][i];
68     printf("%d\n", ans);
69     return 0;
70 }

```

## 2.13 Dinic

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1001, M = 10001;
4  int n, m, S, T, u, v, r, tot, ans
5  int Head[N], cur[N], Next[M << 1], Link[M << 1], Rest[M << 1], d[N], From[N], Edge[N];
6  queue<int> q;
7  inline void AddEdge(int u, int v, int r)
8  {
9      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Rest[tot] = r;
12     Head[u] = tot;
13 }
14 bool BFS()
15 {
16     for (int i = 1; i <= n; i++)
17         d[i] = INT_MAX;
18     d[S] = 0;
19     q.push(S);
20     while (!q.empty())
21     {
22         int now = q.front();
23         q.pop();
24         for (int i = Head[now], j; i; i = Next[i])
25             if (Rest[i] > 0 && d[now] + 1 < d[j = Link[i]])
26             {
27                 d[j] = d[now] + 1;
28                 q.push(j);
29             }
30     }
31     return d[T] < INT_MAX;
32 }
33 bool DFS(int x)
34 {
35     if (x == T)
36     {
37         int tmp = INT_MAX;
38         for (int i = T; i != S; i = From[i])
39             tmp = min(tmp, Rest[Edge[i]]);
40         for (int i = T; i != S; i = From[i])
41         {
42             Rest[Edge[i]] -= tmp;
43             Rest[Edge[i] ^ 1] += tmp;
44         }
45         ans += tmp;
46         return true;
47     }
48     for (int &i = cur[x], j; i; i = Next[i])
49         if (Rest[i] > 0 && d[x] + 1 == d[j = Link[i]])
50         {
51             From[j] = x;
52             Edge[j] = i;
53             if (DFS(j))
54                 return true;
55         }
56     return false;
57 }
58 int main()
59 {
60     scanf("%d%d%d%d", &n, &m, &S, &T);
61     tot = 1;
62     for (int i = 1; i <= m; i++)
63     {
64         scanf("%d%d%d", &u, &v, &r);
65         AddEdge(u, v, r);
66         AddEdge(v, u, 0);
67     }
68     while (BFS())
69     {
70         memcpy(cur, Head, sizeof(cur));
71         while (DFS(S));
72     }

```

```

73     printf("%d\n", ans);
74     return 0;
75 }

```

## 2.14 Edmonds-Karp (最小费用最大流)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1001, M = 10001;
4  int n, m, S, T, u, v, r, c, tmp, tot, sum, ans1, ans2;
5  int Head[N], Next[M << 1], Link[M << 1], Rest[M << 1], Cost[M << 1], d[N], From[N], Edge[N];
6  bool flag[N];
7  queue<int> q;
8  inline void AddEdge(int u, int v, int r, int c)
9  {
10     Next[++tot] = Head[u];
11     Link[tot] = v;
12     Rest[tot] = r;
13     Cost[tot] = c;
14     Head[u] = tot;
15 }
16 bool BFS()
17 {
18     for (int i = 1; i <= n; i++)
19         d[i] = INT_MAX;
20     d[S] = 0;
21     q.push(S);
22     flag[S] = true;
23     while (!q.empty())
24     {
25         int now = q.front();
26         q.pop();
27         flag[now] = false;
28         for (int i = Head[now], j; i; i = Next[i])
29             if (Rest[i] > 0 && d[now] + Cost[i] < d[j = Link[i]])
30             {
31                 d[j] = d[now] + Cost[i];
32                 From[j] = now;
33                 Edge[j] = i;
34                 if (!flag[j])
35                 {
36                     q.push(j);
37                     flag[j] = true;
38                 }
39             }
40     }
41     return d[T] < INT_MAX;
42 }
43 int main()
44 {
45     scanf("%d%d%d%d", &n, &m, &S, &T);
46     tot = 1;
47     for (int i = 1; i <= m; i++)
48     {
49         scanf("%d%d%d%d", &u, &v, &r, &c);
50         AddEdge(u, v, r, c);
51         AddEdge(v, u, 0, -c);
52     }
53     while (BFS())
54     {
55         sum = 0;
56         tmp = INT_MAX;
57         for (int i = T; i != S; i = From[i])
58         {
59             sum += Cost[Edge[i]];
60             tmp = min(tmp, Rest[Edge[i]]);
61         }
62         for (int i = T; i != S; i = From[i])
63         {
64             Rest[Edge[i]] -= tmp;
65             Rest[Edge[i] ^ 1] += tmp;

```



```

66     }
67     ans1 += tmp;
68     ans2 += tmp * sum;
69 }
70 printf("%d_%d\n", ans1, ans2);
71 return 0;
72 }

```

## 3 树算法

### 3.1 Tarjan (最近公共祖先)

```

1  #include <bits/stdc++.h>
2  #define fi first
3  #define se second
4  using namespace std;
5  const int N = 1000001, M = 1000001;
6  int n, m, u, v, tot, Head[N], Next[N << 1], Link[N << 1], a[N], ans[M];
7  bool flag[N];
8  vector<pair<int, int> > Q[N];
9  inline void AddEdge(int u, int v)
10 {
11     Next[++tot] = Head[u];
12     Link[tot] = v;
13     Head[u] = tot;
14 }
15 int Get(int x)
16 {
17     if (a[x] != x)
18         a[x] = Get(a[x]);
19     return a[x];
20 }
21 void DFS(int x)
22 {
23     flag[x] = true;
24     a[x] = x;
25     for (int i = 0; i < Q[x].size(); i++)
26         ans[Q[x][i].se] = Get(a[Q[x][i].fi]);
27     for (int i = Head[x], j; i; i = Next[i])
28         if (!flag[j = Link[i]])
29         {
30             DFS(j);
31             a[j] = x;
32         }
33 }
34 int main()
35 {
36     scanf("%d", &n);
37     for (int i = 1; i < n; i++)
38     {
39         scanf("%d%d", &u, &v);
40         AddEdge(u, v);
41         AddEdge(v, u);
42     }
43     scanf("%d", &m);
44     for (int i = 1; i <= m; i++)
45     {
46         scanf("%d%d", &u, &v);
47         Q[u].push_back({v, i});
48         Q[v].push_back({u, i});
49     }
50     DFS(1);
51     for (int i = 1; i <= m; i++)
52         printf("%d\n", ans[i]);
53     return 0;
54 }

```

### 3.2 树链剖分

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int n, u, v, num, tot
5  int Head[N], Next[N << 1], Link[N << 1], d[N], f[N], s[N], son[N], top[N], idx[N], key[N];
6  inline void AddEdge(int u, int v)
7  {
8      Next[++tot] = Head[u];
9      Link[tot] = v;
10     Head[u] = tot;
11 }
12 void DFS1(int x)
13 {
14     d[x] = d[f[x]] + 1;
15     s[x] = 1;
16     for (int i = Head[x], j; i; i = Next[i])
17         if (!d[j = Link[i]])
18             {
19                 f[j] = x;
20                 DFS1(j);
21                 s[x] += s[j];
22                 if (s[j] > s[son[x]])
23                     son[x] = j;
24             }
25 }
26 void DFS2(int x)
27 {
28     top[x] = x == son[f[x]] ? top[f[x]] : x;
29     key[idx[x] = ++num] = x;
30     if (son[x])
31         DFS2(son[x]);
32     for (int i = Head[x], j; i; i = Next[i])
33     {
34         j = Link[i];
35         if (f[j] == x && j != son[x])
36             DFS2(j);
37     }
38 }
39 int main()
40 {
41     scanf("%d", &n);
42     for (int i = 1; i < n; i++)
43     {
44         scanf("%d%d", &u, &v);
45         AddEdge(u, v);
46         AddEdge(v, u);
47     }
48     DFS1(1);
49     DFS2(1);
50     for (int i = 1; i < n; i++)
51         printf("%d_", key[i]);
52     printf("%d\n", key[n]);
53     return 0;
54 }

```

## 4 数据结构

### 4.1 字母树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001;
4  int pos, son[N][26], num[N];
5  void Insert(char s[])
6  {
7      int t = 1, tmp;
8      for (int i = 0; s[i]; i++)
9      {
10         tmp = s[i] - 97;

```

```

11         if (!son[t][tmp])
12             son[t][tmp] = ++pos;
13         t = son[t][tmp];
14     }
15     num[t]++;
16 }
17 int Find(char s[])
18 {
19     int t = 1, tmp;
20     for (int i = 0; s[i]; i++)
21     {
22         tmp = s[i] - 97;
23         if (!son[t][tmp])
24             return 0;
25         t = son[t][tmp];
26     }
27     return num[t];
28 }
29 int main()
30 {
31     pos = 1;
32     return 0;
33 }

```

## 4.2 并查集

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001;
4  int n, a[N], b[N];
5  int Find(int x)
6  {
7      if (a[x] != x)
8          a[x] = Find(a[x]);
9      return a[x];
10 }
11 void Merge(int x, int y)
12 {
13     if ((x = Find(x)) == (y = Find(y)))
14         return;
15     b[x] < b[y] ? a[x] = y : a[y] = x;
16     if (b[x] == b[y])
17         b[x]++;
18 }
19 int main()
20 {
21     for (int i = 1; i <= n; i++)
22         a[i] = i;
23     return 0;
24 }

```

## 4.3 树状数组

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int n, sum[N];
5  void Add(int x, int y)
6  {
7      for (; x <= n; x += x & -x)
8          sum[x] += y;
9  }
10 int Sum(int x)
11 {
12     int ans = 0;
13     for (; x; x -= x & -x)
14         ans += sum[x];
15     return ans;

```

```

16 }
17 int main()
18 {
19     return 0;
20 }

```

## 4.4 张昆玮线段树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int n, SIZE, a[N], sum[N << 2];
5  void Build()
6  {
7      for (SIZE = 1; SIZE < n + 2; SIZE <= 1);
8      for (int i = 1; i <= n; i++)
9          sum[SIZE + i] = a[i];
10     for (int i = SIZE - 1; i; i--)
11         sum[i] = sum[i << 1] + sum[(i << 1) + 1];
12 }
13 void Add(int x, int y)
14 {
15     for (x += SIZE; x; x >>= 1)
16         sum[x] += y;
17 }
18 int Sum(int x, int y)
19 {
20     int ans = 0;
21     for (x += SIZE - 1, y += SIZE + 1; x ^ y ^ 1; x >>= 1, y >>= 1)
22     {
23         if ((x & 1) == 0)
24             ans += sum[x ^ 1];
25         if ((y & 1) == 1)
26             ans += sum[y ^ 1];
27     }
28     return ans;
29 }
30 int main()
31 {
32     return 0;
33 }

```

## 4.5 线段树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int num, a[N], l[N << 1], r[N << 1], sum[N << 1], lab[N << 1];
5  inline void Label(int p, int x, int y, int z)
6  {
7      sum[p] += (y - x + 1) * z;
8      lab[p] += z;
9  }
10 inline void Down(int p, int x, int y)
11 {
12     if (x < y)
13     {
14         int z = x + y >> 1;
15         Label(l[p], x, z, lab[p]);
16         Label(r[p], z + 1, y, lab[p]);
17     }
18     lab[p] = 0;
19 }
20 inline void Up(int p)
21 {
22     sum[p] = sum[l[p]] + sum[r[p]];
23 }
24 void Build(int p, int x, int y)

```

```

25 {
26     if (x == y)
27     {
28         sum[p] = a[x];
29         return;
30     }
31     int z = x + y >> 1;
32     Build(l[p] = ++num, x, z);
33     Build(r[p] = ++num, z + 1, y);
34     Up(p);
35 }
36 void Add(int p, int x, int y, int a, int b, int c)
37 {
38     Down(p, x, y);
39     if (x == a && y == b)
40     {
41         Label(p, x, y, c);
42         return;
43     }
44     int z = x + y >> 1;
45     if (b <= z)
46         Add(l[p], x, z, a, b, c);
47     else if (a > z)
48         Add(r[p], z + 1, y, a, b, c);
49     else
50     {
51         Add(l[p], x, z, a, z, c);
52         Add(r[p], z + 1, y, z + 1, b, c);
53     }
54     Up(p);
55 }
56 int Sum(int p, int x, int y, int a, int b)
57 {
58     Down(p, x, y);
59     if (x == a && y == b)
60         return sum[p];
61     int z = x + y >> 1;
62     if (b <= z)
63         return Sum(l[p], x, z, a, b);
64     else if (a > z)
65         return Sum(r[p], z + 1, y, a, b);
66     else
67         return Sum(l[p], x, z, a, z) + Sum(r[p], z + 1, y, z + 1, b);
68 }
69 int main()
70 {
71     num = 1;
72     return 0;
73 }

```

## 4.6 伸展树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int root, pos, l[N], r[N], f[N], key[N], s[N], num[N];
5  inline void L(int p)
6  {
7      int t = f[p];
8      if (r[t] = l[p])
9          f[l[p]] = t;
10     if (f[p] = f[t])
11         t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
12     f[t] = p;
13     l[p] = t;
14     s[p] = s[t];
15     s[t] = s[l[t]] + s[r[t]] + num[t];
16 }
17 inline void R(int p)
18 {
19     int t = f[p];

```

```

20     if (l[t] = r[p])
21         f[r[p]] = t;
22     if (f[p] = f[t])
23         t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
24     f[t] = p;
25     r[p] = t;
26     s[p] = s[t];
27     s[t] = s[l[t]] + s[r[t]] + num[t];
28 }
29 void Splay(int p)
30 {
31     for (int t; t = f[p]; )
32         if (!f[t])
33             p == l[t] ? R(p) : L(p);
34         else
35             if (p == l[t])
36                 t == l[f[t]] ? (R(t), R(p)) : (R(p), L(p));
37             else
38                 t == r[f[t]] ? (L(t), L(p)) : (L(p), R(p));
39     root = p;
40 }
41 void Insert(int x)
42 {
43     int p, t;
44     bool flag = false;
45     for (p = root; p; p = x < key[p] ? l[p] : r[p])
46     {
47         t = p;
48         s[p]++;
49         if (key[p] == x)
50         {
51             flag = true;
52             break;
53         }
54     }
55     if (flag)
56         num[p]++;
57     else
58     {
59         p = ++pos;
60         key[p] = x;
61         s[p] = num[p] = 1;
62         if (root)
63         {
64             f[p] = t;
65             x < key[t] ? l[t] = p : r[t] = p;
66         }
67     }
68     Splay(p);
69 }
70 void Delete(int x)
71 {
72     int p, q, t;
73     for (p = root; key[p] != x; p = x < key[p] ? l[p] : r[p])
74         s[p]--;
75     s[p]--;
76     if (!(--num[p]))
77         if (!l[p] || !r[p])
78         {
79             if (p == root)
80                 root = l[p] + r[p];
81             else
82                 p == l[f[p]] ? l[f[p]] = l[p] + r[p] : r[f[p]] = l[p] + r[p];
83             f[l[p] + r[p]] = f[p];
84         }
85     else
86     {
87         for (q = l[p]; r[q]; q = r[q]);
88         for (t = l[p]; r[t]; t = r[t])
89             s[t] -= num[q];
90         q == l[f[q]] ? l[f[q]] = l[q] + r[q] : r[f[q]] = l[q] + r[q];
91         f[l[q] + r[q]] = f[q];
92         key[p] = key[q];

```

```

93         num[p] = num[q];
94     }
95 }
96 int Rank(int x)
97 {
98     int p = root, t = s[l[root]];
99     while (key[p] != x)
100         if (x < key[p])
101         {
102             p = l[p];
103             t -= s[r[p]] + num[p];
104         }
105         else
106         {
107             t += num[p];
108             p = r[p];
109             t += s[l[p]];
110         }
111     Splay(p);
112     return t + 1;
113 }
114 int Select(int x)
115 {
116     int p = root, t = s[l[root]];
117     while (x < t + 1 || x > t + num[p])
118         if (x < t + 1)
119         {
120             p = l[p];
121             t -= s[r[p]] + num[p];
122         }
123         else
124         {
125             t += num[p];
126             p = r[p];
127             t += s[l[p]];
128         }
129     Splay(p);
130     return key[p];
131 }
132 int Pred(int x)
133 {
134     int p = root, t;
135     while (p)
136         if (x > key[p])
137         {
138             t = p;
139             p = r[p];
140         }
141         else
142             p = l[p];
143     Splay(t);
144     return key[t];
145 }
146 int Succ(int x)
147 {
148     int p = root, t;
149     while (p)
150         if (x < key[p])
151         {
152             t = p;
153             p = l[p];
154         }
155         else
156             p = r[p];
157     Splay(t);
158     return key[t];
159 }
160 int main()
161 {
162     return 0;
163 }

```

## 4.7 伸展树 (区间)

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int root, pos, l[N], r[N], f[N], s[N], key[N], lab[N], sum[N];
5  bool flag[N];
6  inline void Down(int p)
7  {
8      if (l[p])
9      {
10         key[l[p]] += lab[p];
11         lab[l[p]] += lab[p];
12         sum[l[p]] += s[l[p]] * lab[p];
13         if (flag[p])
14         {
15             flag[l[p]] = !flag[l[p]];
16             swap(l[l[p]], r[l[p]]);
17         }
18     }
19     if (r[p])
20     {
21         key[r[p]] += lab[p];
22         lab[r[p]] += lab[p];
23         sum[r[p]] += s[r[p]] * lab[p];
24         if (flag[p])
25         {
26             flag[r[p]] = !flag[r[p]];
27             swap(l[r[p]], r[r[p]]);
28         }
29     }
30     lab[p] = 0;
31     flag[p] = false;
32 }
33 inline void Up(int p)
34 {
35     s[p] = s[l[p]] + s[r[p]] + 1;
36     sum[p] = sum[l[p]] + sum[r[p]] + key[p];
37 }
38 inline void L(int p)
39 {
40     int t = f[p];
41     if (r[t] == l[p])
42         f[l[p]] = t;
43     if (f[p] == f[t])
44         t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
45     f[t] = p;
46     l[p] = t;
47 }
48 inline void R(int p)
49 {
50     int t = f[p];
51     if (l[t] == r[p])
52         f[r[p]] = t;
53     if (f[p] == f[t])
54         t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
55     f[t] = p;
56     r[p] = t;
57 }
58 void Splay(int p, int T)
59 {
60     for (int q, t; (q = f[p]) != T; )
61         if (f[q] == T)
62         {
63             p == l[q] ? R(p) : L(p);
64             Up(q), Up(p);
65         }
66         else
67         {
68             t = f[q];
69             if (p == l[q])
70                 q == l[t] ? (R(q), R(p)) : (R(p), L(p));
71             else
```



```

72         q == r[t] ? (L(q), L(p)) : (L(p), R(p));
73         Up(t), Up(q), Up(p);
74     }
75     if (!T)
76         root = p;
77 }
78 int Select(int x)
79 {
80     int p = root, t = s[l[root]];
81     Down(p);
82     while (x != t + 1)
83     {
84         if (x < t + 1)
85             t -= s[r[p = l[p]]] + 1;
86         else
87             t += s[l[p = r[p]]] + 1;
88         Down(p);
89     }
90     return p;
91 }
92 void Insert(int x, int y)
93 {
94     int p = Select(x + 1);
95     Splay(p, 0);
96     Down(p);
97     for (p = r[p]; l[p]; p = l[p])
98         Down(p);
99     Down(p);
100     l[p] = ++pos;
101     f[pos] = p;
102     sum[pos] = key[pos] = y;
103     Splay(pos, 0);
104 }
105 void Delete(int x)
106 {
107     int p = Select(x + 1);
108     Splay(p, 0);
109     Down(p);
110     for (p = l[p]; r[p]; p = r[p])
111         Down(p);
112     Down(p);
113     f[r[root]] = p;
114     r[p] = r[root];
115     f[l[root]] = 0;
116     Splay(p, 0);
117 }
118 void Add(int x, int y, int z)
119 {
120     Splay(Select(x), 0);
121     Splay(Select(y + 2), root);
122     key[l[r[root]]] += z;
123     lab[l[r[root]]] += z;
124     sum[l[r[root]]] += s[l[r[root]]] * z;
125     Up(r[root]), Up(root);
126 }
127 void Reverse(int x, int y)
128 {
129     Splay(Select(x), 0);
130     Splay(Select(y + 2), root);
131     flag[l[r[root]]] = !flag[l[r[root]]];
132     swap(l[l[r[root]]], r[l[r[root]]]);
133     Up(r[root]), Up(root);
134 }
135 int Sum(int x, int y)
136 {
137     Splay(Select(x), 0);
138     Splay(Select(y + 2), root);
139     return sum[l[r[root]]];
140 }
141 int main()
142 {
143     root = 1;
144     pos = 2;

```

```

145     r[1] = s[1] = 2;
146     f[2] = s[2] = 1;
147     return 0;
148 }

```

## 4.8 节点大小平衡树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  int root, pos, l[N], r[N], f[N], s[N], num[N], key[N];
5  inline void L(int p)
6  {
7      int t = r[p];
8      if (r[p] = l[t])
9          f[l[t]] = p;
10     l[t] = p;
11     if (f[t] = f[p])
12         p == l[f[p]] ? l[f[p]] = t : r[f[p]] = t;
13     f[p] = t;
14     s[t] = s[p];
15     s[p] = s[l[p]] + s[r[p]] + num[p];
16     if (p == root)
17         root = t;
18 }
19 inline void R(int p)
20 {
21     int t = l[p];
22     if (l[p] = r[t])
23         f[r[t]] = p;
24     r[t] = p;
25     if (f[t] = f[p])
26         p == l[f[p]] ? l[f[p]] = t : r[f[p]] = t;
27     f[p] = t;
28     s[t] = s[p];
29     s[p] = s[l[p]] + s[r[p]] + num[p];
30     if (p == root)
31         root = t;
32 }
33 void Fix(int p, bool flag)
34 {
35     if (flag)
36         if (s[l[r[p]]] > s[l[p]])
37             R(r[p]), L(p);
38         else
39             if (s[r[r[p]]] > s[l[p]])
40                 L(p);
41             else
42                 return;
43     else
44         if (s[r[l[p]]] > s[r[p]])
45             L(l[p]), R(p);
46         else
47             if (s[l[l[p]]] > s[r[p]])
48                 R(p);
49             else
50                 return;
51     Fix(l[p], 0);
52     Fix(r[p], 1);
53     Fix(p, 0);
54     Fix(p, 1);
55 }
56 void Insert(int p, int q, int x)
57 {
58     if (!p)
59     {
60         p = ++pos;
61         if (q)
62             x < key[q] ? l[q] = p : r[q] = p;
63         else
64             root = p;

```

```

65     key[p] = x;
66     f[p] = q;
67     s[p] = num[p] = 1;
68 }
69 else
70 {
71     s[p]++;
72     if (x == key[p])
73         num[p]++;
74     else
75     {
76         Insert(x < key[p] ? l[p] : r[p], p, x);
77         Fix(p, x > key[p]);
78     }
79 }
80 }
81 void Delete(int x)
82 {
83     int p, q, t;
84     for (p = root; key[p] != x; p = x < key[p] ? l[p] : r[p])
85         s[p]--;
86     s[p]--;
87     if (!(--num[p]))
88         if (!l[p] || !r[p])
89         {
90             if (p == root)
91                 root = l[p] + r[p];
92             else
93                 p == l[f[p]] ? l[f[p]] = l[p] + r[p] : r[f[p]] = l[p] + r[p];
94             f[l[p] + r[p]] = f[p];
95         }
96     else
97     {
98         for (q = l[p]; r[q]; q = r[q]);
99         for (t = l[p]; r[t]; t = r[t])
100             s[t] -= num[q];
101         q == l[f[q]] ? l[f[q]] = l[q] + r[q] : r[f[q]] = l[q] + r[q];
102         f[l[q] + r[q]] = f[q];
103         key[p] = key[q];
104         num[p] = num[q];
105     }
106 }
107 int Rank(int x)
108 {
109     int p = root, t = s[l[root]];
110     while (key[p] != x)
111         if (x < key[p])
112         {
113             p = l[p];
114             t -= s[r[p]] + num[p];
115         }
116         else
117         {
118             t += num[p];
119             p = r[p];
120             t += s[l[p]];
121         }
122     return t + 1;
123 }
124 int Select(int x)
125 {
126     int p = root, t = s[l[root]];
127     while (x < t + 1 || x > t + num[p])
128         if (x < t + 1)
129         {
130             p = l[p];
131             t -= s[r[p]] + num[p];
132         }
133         else
134         {
135             t += num[p];
136             p = r[p];
137             t += s[l[p]];

```

```

138     }
139     return key[p];
140 }
141 int Pred(int x)
142 {
143     int p = root, t;
144     while (p)
145         if (x > key[p])
146         {
147             t = p;
148             p = r[p];
149         }
150         else
151             p = l[p];
152     return key[t];
153 }
154 int Succ(int x)
155 {
156     int p = root, t;
157     while (p)
158         if (x < key[p])
159         {
160             t = p;
161             p = l[p];
162         }
163         else
164             p = r[p];
165     return key[t];
166 }
167 int main()
168 {
169     return 0;
170 }

```

## 5 数论

### 5.1 快速幂

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int a, b, ans;
4  int main()
5  {
6      scanf("%d%d", &a, &b);
7      ans = 1;
8      while (b)
9      {
10         if (b & 1)
11             ans = ans * a;
12         a = a * a;
13         b >>= 1;
14     }
15     printf("%d\n", ans);
16     return 0;
17 }

```

### 5.2 Euclid

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int a, b;
4  int gcd(int a, int b)
5  {
6      return b ? gcd(b, a % b) : a;
7  }
8  int main()
9  {
10     scanf("%d%d", &a, &b);

```

```

11     printf("%d\n", gcd(a, b));
12     return 0;
13 }

```

### 5.3 扩展 Euclid

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int a, b, x, y, t;
4  int gcd(int a, int b, int &x, int &y)
5  {
6      if (b)
7      {
8          int t, xt, yt;
9          t = gcd(b, a % b, xt, yt);
10         x = yt;
11         y = xt - a / b * yt;
12         return t;
13     }
14     else
15     {
16         x = 1;
17         y = 0;
18         return a;
19     }
20 }
21 int main()
22 {
23     scanf("%d%d", &a, &b);
24     t = gcd(a, b, x, y);
25     printf("%d_ %d_ %d\n", x, y, t);
26     return 0;
27 }

```

### 5.4 Euler 筛

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001;
4  int n, num, p[N], fai[N], miu[N];
5  bool flag[N];
6  int main()
7  {
8      scanf("%d", &n);
9      fai[1] = miu[1] = 1;
10     for (int i = 2; i <= n; i++)
11     {
12         if (!flag[i])
13         {
14             p[++num] = i;
15             fai[i] = i - 1;
16             miu[i] = -1;
17         }
18         for (int j = 1; j <= num; j++)
19         {
20             if (i * p[j] > n)
21                 break;
22             flag[i * p[j]] = true;
23             if (i % p[j] == 0)
24             {
25                 fai[i * p[j]] = fai[i] * p[j];
26                 miu[i * p[j]] = 0;
27                 break;
28             }
29             else
30             {
31                 fai[i * p[j]] = fai[i] * (p[j] - 1);
32                 miu[i * p[j]] = -miu[i];

```

```

33     }
34 }
35 }
36 printf("%d\n", num);
37 for (int i = 1; i < num; i++)
38     printf("%d_", p[i]);
39 printf("%d\n", p[num]);
40 for (int i = 1; i < n; i++)
41     printf("%d_", fai[i]);
42 printf("%d\n", fai[n]);
43 for (int i = 1; i < n; i++)
44     printf("%d_", miu[i]);
45 printf("%d\n", miu[n]);
46 return 0;
47 }

```

## 6 计算几何

### 6.1 线段相交

```

1  #include <bits/stdc++.h>
2  #define x first
3  #define y second
4  #define x1 first.first
5  #define y1 first.second
6  #define x2 second.first
7  #define y2 second.second
8  using namespace std;
9  typedef pair<double, double> Point;
10 typedef pair<Point, Point> Segment;
11 Segment a, b;
12 inline double Cross(Point a, Point b, Point c)
13 {
14     return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
15 }
16 int main()
17 {
18     scanf("%lf%lf%lf%lf%lf%lf%lf%lf", &a.x1, &a.y1, &a.x2, &a.y2, &b.x1, &b.y1, &b.x2, &b.y2);
19     if (max(a.x1, a.x2) < min(b.x1, b.x2) || max(b.x1, b.x2) < min(a.x1, a.x2))
20         puts("NO");
21     else if (max(a.y1, a.y2) < min(b.y1, b.y2) || max(b.y1, b.y2) < min(a.y1, a.y2))
22         puts("NO");
23     else
24         puts(Cross(a.x, a.y, b.x) * Cross(a.x, a.y, b.y) <= 0 &&
25             Cross(b.x, b.y, a.x) * Cross(b.x, b.y, a.y) <= 0 ? "YES" : "NO");
26     return 0;
27 }

```

### 6.2 多边形面积

```

1  #include <bits/stdc++.h>
2  #define x first
3  #define y second
4  using namespace std;
5  typedef pair<double, double> Point;
6  const int N = 1000001;
7  int n;
8  double ans;
9  Point p[N];
10 inline double Cross(Point a, Point b, Point c)
11 {
12     return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
13 }
14 int main()
15 {
16     scanf("%d", &n);
17     for (int i = 1; i <= n; i++)

```

```

18     scanf("%lf%lf", &p[i].x, &p[i].y);
19     for (int i = 3; i <= n; i++)
20         ans += Cross(p[1], p[i - 1], p[i]);
21     printf("%.5f\n", ans / 2);
22     return 0;
23 }

```

### 6.3 Graham 扫描

```

1  #include <bits/stdc++.h>
2  #define x first
3  #define y second
4  using namespace std;
5  typedef pair<double, double> Point;
6  const int N = 100001;
7  int n, top;
8  Point p[N], s[N];
9  inline double Sqr(double x)
10 {
11     return x * x;
12 }
13 inline double Dist(Point a, Point b)
14 {
15     return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
16 }
17 inline double Cross(Point a, Point b, Point c)
18 {
19     return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
20 }
21 inline bool cmp(Point a, Point b)
22 {
23     return Cross(p[0], a, b) > 0 || Cross(p[0], a, b) == 0 && Dist(p[0], a) < Dist(p[0], b);
24 }
25 int main()
26 {
27     scanf("%d", &n);
28     for (int i = 0; i < n; i++)
29     {
30         scanf("%lf%lf", &p[i].x, &p[i].y);
31         if (p[i].y < p[0].y || p[i].y == p[0].y && p[i].x < p[0].x)
32             swap(p[0], p[i]);
33     }
34     sort(p + 1, p + n, cmp);
35     s[top = 1] = p[0];
36     for (int i = 1; i < n; i++)
37     {
38         for (; top > 1 && Cross(s[top - 1], s[top], p[i]) < 0; top--);
39         s[++top] = p[i];
40     }
41     for (; top > 2 && Cross(s[top - 1], s[top], s[1]) < 0; top--);
42     printf("%d\n", top);
43     for (int i = 1; i <= top; i++)
44         printf("%.5f_%.5f\n", s[i].x, s[i].y);
45     return 0;
46 }

```

### 6.4 最小圆覆盖

```

1  #include <bits/stdc++.h>
2  #define x first
3  #define y second
4  using namespace std;
5  typedef pair<double, double> Point;
6  const int N = 1000001;
7  int x, y, n;
8  double r;
9  Point O, p[N];
10 inline double Sqr(double x)

```

```

11 {
12     return x * x;
13 }
14 inline double Dist(Point a, Point b)
15 {
16     return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
17 }
18 inline Point Calc(Point a, Point b, Point c)
19 {
20     if (fabs((b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y)) < 1e-5)
21         if (Dist(a, c) > Dist(b, c))
22             return {(a.x + c.x) / 2, (a.y + c.y) / 2};
23         else
24             return {(b.x + c.x) / 2, (b.y + c.y) / 2};
25     double k1, k2, b1, b2;
26     k1 = (a.x - c.x) / (c.y - a.y);
27     b1 = (a.y + c.y) / 2 - k1 * (a.x + c.x) / 2;
28     k2 = (b.x - c.x) / (c.y - b.y);
29     b2 = (b.y + c.y) / 2 - k2 * (b.x + c.x) / 2;
30     return {(b2 - b1) / (k1 - k2), (k1 * b2 - k2 * b1) / (k1 - k2)};
31 }
32 int main()
33 {
34     scanf("%d", &n);
35     for (int i = 1; i <= n; i++)
36         scanf("%lf%lf", &p[i].x, &p[i].y);
37     random_shuffle(p + 1, p + n + 1);
38     O = p[1];
39     r = 0;
40     for (int i = 2; i <= n; i++)
41         if (Dist(O, p[i]) > r)
42         {
43             O = p[i];
44             r = 0;
45             for (int j = 1; j < i; j++)
46                 if (Dist(O, p[j]) > r)
47                 {
48                     O = {(p[i].x + p[j].x) / 2, (p[i].y + p[j].y) / 2};
49                     r = Dist(O, p[j]);
50                     for (int k = 1; k < j; k++)
51                         if (Dist(O, p[k]) > r)
52                         {
53                             O = Calc(p[i], p[j], p[k]);
54                             r = Dist(O, p[k]);
55                         }
56                 }
57         }
58     printf("%.5f_%.5f\n%.5f\n", O.x, O.y, r);
59     return 0;
60 }

```