

模板

Wajov

January 19, 2018

目录

1	字符串算法	3
1.1	最小表示	3
1.2	Manacher	3
1.3	Knuth-Morris-Pratt	3
1.4	扩展 Knuth-Morris-Pratt	4
1.5	Aho-Corasick	4
1.6	后缀数组	5
2	图算法	6
2.1	拓扑排序	6
2.2	Floyd-Warshall	6
2.3	Floyd-Warshall (最小环)	7
2.4	Bellman-Ford+ 队列	7
2.5	Dijkstra+ 堆	8
2.6	Prim+ 堆	8
2.7	Tarjan (强连通分量)	9
2.8	Tarjan (点双连通分量)	10
2.9	Tarjan (边双连通分量)	11
2.10	匈牙利	11
2.11	Kuhn-Munkres	12
2.12	Dinic	13
2.13	Edmonds-Karp (最小费用最大流)	14
3	树算法	15
3.1	Tarjan (最近公共祖先)	15
3.2	树链剖分	16
4	数据结构	16
4.1	并查集	16
4.2	字母树	17
4.3	左偏树	17
4.4	树状数组	18
4.5	张昆玮线段树	18
4.6	线段树	19
4.7	伸展树 (区间)	19
4.8	节点大小平衡树	21
5	数学	23
5.1	快速幂	23
5.2	Euclid	23
5.3	扩展 Euclid	24
5.4	Miller-Rabin 测试	24
5.5	Euler 筛	25
5.6	Gauss 消元	25
5.7	快速 Fourier 变换	26

6	计算几何	27
6.1	线段相交	27
6.2	多边形面积	27
6.3	Graham 扫描	28
6.4	最小圆覆盖	28

1 字符串算法

1.1 最小表示

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

1.2 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     s[0] = '$';
8     while ((c = getchar()) != '\n') {
9         s[++n] = '#';
10        s[++n] = c;
11    }
12    s[++n] = '#';
13    for (int i = 1, j = 0; i <= n; i++) {
14        p[i] = i < j + p[j] ? min(p[(j << 1) - i], j + p[j] - i) : 1;
15        while (s[i + p[i]] == s[i - p[i]])
16            p[i]++;
17        if (i + p[i] > j + p[j])
18            j = i;
19        ans = max(ans, p[i] - 1);
20    }
21    printf("%d\n", ans);
22    return 0;
23 }
```

1.3 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   scanf("%s%s", a + 1, b + 1);
8   n = strlen(a + 1);
9   m = strlen(b + 1);
10  for (int i = 2, j = 0; i <= m; i++) {
11      for (; j > 0 && b[j + 1] != b[i]; j = p[j]);
12      if (b[j + 1] == b[i])
13          j++;
14      p[i] = j;
15  }
16  for (int i = 1, j = 0; i <= n; i++) {
17      for (; j > 0 && b[j + 1] != a[i]; j = p[j]);
18      if (b[j + 1] == a[i])
19          j++;
20      if (j == m) {
21          ans[++num] = i - j + 1;
22          j = p[j];
23      }
24  }
25  for (int i = 1; i < num; i++)
26      printf("%d□", ans[i]);
27  printf("%d\n", ans[num]);
28  return 0;
29 }

```

1.4 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      scanf("%s%s", a + 1, b + 1);
8      n = strlen(a + 1);
9      m = strlen(b + 1);
10     for (int i = 2, j = 0; i <= m; i++) {
11         p[i] = i < j + p[j] ? min(p[i - j + 1], j + p[j] - i) : 0;
12         for (; i + p[i] <= m && b[i + p[i]] == b[p[i] + 1]; p[i]++);
13         if (i + p[i] > j + p[j])
14             j = i;
15     }
16     for (int i = 1, j = 0; i <= n; i++) {
17         ex[i] = i <= j + ex[j] ? min(p[i - j + 1], j + ex[j] - i) : 0;
18         for (; i + ex[i] <= n && ex[i] < m && a[i + ex[i]] == b[ex[i] + 1]; ex[i]++);
19         if (i + ex[i] > j + ex[j])
20             j = i;
21     }
22     for (int i = 1; i < n; i++)
23         printf("%d□", ex[i]);
24     printf("%d\n", ex[n]);
25     return 0;
26 }

```

1.5 Aho-Corasick

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1000001;
4  int n, t, tmp, now, pos, 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      int p = 1, t;
9      for (int i = 0; s[i]; i++) {
10         t = s[i] - 97;
11         if (!son[p][t])
12             son[p][t] = ++pos;
13         p = son[p][t];
14     }

```

```

15     num[p]++;
16 }
17 int main() {
18     pos = 1;
19     scanf("%s%d", a, &n);
20     for (int i = 1; i <= n; i++) {
21         scanf("%s", b);
22         Insert(b);
23     }
24     q.push(1);
25     while (!q.empty()) {
26         now = q.front();
27         q.pop();
28         for (int i = 0; i < 26; i++)
29             if (son[now][i]) {
30                 for (t = p[now]; t > 0 && son[t][i] == 0; t = p[t]);
31                 p[son[now][i]] = t ? son[t][i] : 1;
32                 q.push(son[now][i]);
33             }
34     }
35     t = 1;
36     for (int i = 0; a[i]; i++) {
37         tmp = a[i] - 97;
38         for (; t > 0 && son[t][tmp] == 0; t = p[t]);
39         t = t ? son[t][tmp] : 1;
40         for (int j = t; j > 1 && num[j] > -1; j = p[j]) {
41             ans += num[j];
42             num[j] = -1;
43         }
44     }
45     printf("%d\n", ans);
46     return 0;
47 }

```

1.6 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      memset(sum, 0, sizeof(sum));
8      for (int i = 1; i <= n; i++)
9          sum[a[i]]++;
10     for (int i = 1; i <= m; i++)
11         sum[i] += sum[i - 1];
12     for (int i = n; i; i--)
13         tmp[id[i]] = sum[a[id[i]]]--;
14     for (int i = 1; i <= n; i++)
15         id[tmp[i]] = i;
16 }
17 int main() {
18     scanf("%s", s + 1);
19     n = strlen(s + 1);
20     for (int i = 1; i <= n; i++)
21         a[id[i] = i] = s[i] - 96;
22     Sort(a, 26);
23     for (int i = 1, t = 0; i <= n; i++)
24         rk[id[i]] = a[id[i]] == a[id[i - 1]] ? t : ++t;
25     for (int i = 1; i <= n; i <= 1) {
26         for (int j = 1; j <= n; j++) {
27             a[j] = rk[j];
28             b[j] = rk[min(i + j, n + 1)];
29         }
30         Sort(b, n);
31         Sort(a, n);
32         for (int j = 1, t = 0; j <= n; j++)
33             rk[id[j]] = a[id[j]] == a[id[j - 1]] && b[id[j]] == b[id[j - 1]] ? t : ++t;
34     }
35     for (int i = 1; i < n; i++)

```

```

36     printf("%d□", rk[i]);
37     printf("%d\n", rk[n]);
38     return 0;
39 }

```

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      Next[++tot] = Head[u];
8      Link[tot] = v;
9      Head[u] = tot;
10 }
11 void DFS(int x) {
12     flag[x] = true;
13     for (int i = Head[x], j; i; i = Next[i])
14         if (!flag[j = Link[i]])
15             DFS(j);
16     ans[++num] = x;
17 }
18 int main() {
19     scanf("%d%d", &n, &m);
20     for (int i = 1; i <= m; i++) {
21         scanf("%d%d", &u, &v);
22         AddEdge(u, v);
23     }
24     for (int i = 1; i <= n; i++)
25         if (!flag[i])
26             DFS(i);
27     for (int i = n; i > 1; i--)
28         printf("%d□", ans[i]);
29     printf("%d\n", ans[1]);
30     return 0;
31 }

```

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      scanf("%d%d", &n, &m);
7      for (int i = 1; i <= n; i++)
8          for (int j = 1; j <= n; j++)
9              d[i][j] = i == j ? 0 : INT_MAX >> 1;
10     for (int i = 1; i <= m; i++) {
11         scanf("%d%d%d", &u, &v, &c);
12         d[u][v] = d[v][u] = min(d[u][v], c);
13     }
14     for (int k = 1; k <= n; k++)
15         for (int i = 1; i <= n; i++)
16             for (int j = 1; j <= n; j++)
17                 d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
18     for (int i = 1; i <= n; i++) {
19         for (int j = 1; j < n; j++)
20             printf("%d□", d[i][j] == INT_MAX >> 1 ? -1 : d[i][j]);
21         printf("%d\n", d[i][n] == INT_MAX >> 1 ? -1 : d[i][n]);
22     }
23     return 0;
24 }

```

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     scanf("%d%d", &n, &m);
7     for (int i = 1; i <= n; i++)
8         for (int j = 1; j <= n; j++) {
9             a[i][j] = i == j ? 0 : INT_MAX / 3;
10            p[i][j] = i;
11        }
12    for (int i = 1; i <= m; i++) {
13        scanf("%d%d%d", &u, &v, &c);
14        a[u][v] = a[v][u] = min(a[u][v], c);
15    }
16    memcpy(d, a, sizeof(d));
17    Min = INT_MAX / 3;
18    for (int k = 1; k <= n; k++) {
19        for (int i = 1; i < k; i++)
20            for (int j = 1; j < i; j++)
21                if (d[i][j] + a[i][k] + a[k][j] < Min) {
22                    Min = d[i][j] + a[i][k] + a[k][j];
23                    for (num = 0, t = j; t != i; t = p[i][t])
24                        ans[++num] = t;
25                    ans[++num] = i;
26                    ans[++num] = k;
27                }
28        for (int i = 1; i <= n; i++)
29            for (int j = 1; j <= n; j++)
30                if (d[i][k] + d[k][j] < d[i][j]) {
31                    d[i][j] = d[i][k] + d[k][j];
32                    p[i][j] = p[k][j];
33                }
34    }
35    printf("%d\n", Min);
36    for (int i = 1; i < num; i++)
37        printf("%d_", ans[i]);
38    printf("%d\n", ans[num]);
39    return 0;
40 }
```

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     Next[++tot] = Head[u];
9     Link[tot] = v;
10    Cost[tot] = c;
11    Head[u] = tot;
12 }
13 int main() {
14     scanf("%d%d%d", &n, &m, &s);
15     for (int i = 1; i <= m; i++) {
16         scanf("%d%d%d", &u, &v, &c);
17         AddEdge(u, v, c);
18         AddEdge(v, u, c);
19     }
20     for (int i = 1; i <= n; i++)
21         d[i] = INT_MAX;
22     d[s] = 0;
23     q.push(s);
24     flag[s] = true;
25     while (!q.empty()) {
26         now = q.front();
```

```

27     q.pop();
28     flag[now] = false;
29     for (int i = Head[now], j; i; i = Next[i])
30         if (d[now] + Cost[i] < d[j = Link[i]]) {
31             d[j] = d[now] + Cost[i];
32             if (!flag[j]) {
33                 q.push(j);
34                 flag[j] = true;
35             }
36         }
37     }
38     for (int i = 1; i < n; i++)
39         printf("%d□", d[i]);
40     printf("%d\n", d[n]);
41     return 0;
42 }

```

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      Next[++tot] = Head[u];
9      Link[tot] = v;
10     Cost[tot] = c;
11     Head[u] = tot;
12 }
13 int main() {
14     scanf("%d%d%d", &n, &m, &s);
15     for (int i = 1; i <= m; i++) {
16         scanf("%d%d%d", &u, &v, &c);
17         AddEdge(u, v, c);
18         AddEdge(v, u, c);
19     }
20     for (int i = 1; i <= n; i++)
21         d[i] = INT_MAX;
22     q.push(make_pair(d[s] = 0, s));
23     while (!q.empty()) {
24         now = q.top().second;
25         q.pop();
26         if (flag[now])
27             continue;
28         flag[now] = true;
29         for (int i = Head[now], j; i; i = Next[i])
30             if (d[now] + Cost[i] < d[j = Link[i]]) {
31                 d[j] = d[now] + Cost[i];
32                 q.push(make_pair(-d[j], j));
33             }
34     }
35     for (int i = 1; i < n; i++)
36         printf("%d□", d[i] == INT_MAX ? -1 : d[i]);
37     printf("%d\n", d[n] == INT_MAX ? -1 : d[n]);
38     return 0;
39 }

```

2.6 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      Next[++tot] = Head[u];

```



```

9     Link[tot] = v;
10    Cost[tot] = c;
11    Head[u] = tot;
12 }
13 int main() {
14     scanf("%d%d", &n, &m);
15     for (int i = 1; i <= m; i++) {
16         scanf("%d%d%d", &u, &v, &c);
17         AddEdge(u, v, c);
18         AddEdge(v, u, c);
19     }
20     for (int i = 1; i <= n; i++)
21         d[i] = INT_MAX;
22     q.push(make_pair(d[1], 0, 1));
23     while (!q.empty()) {
24         now = q.top().second;
25         q.pop();
26         if (flag[now])
27             continue;
28         ans += d[now];
29         flag[now] = true;
30         for (int i = Head[now], j; i; i = Next[i])
31             if (Cost[i] < d[j = Link[i]]) {
32                 d[j] = Cost[i];
33                 q.push(make_pair(-d[j], j));
34             }
35     }
36     printf("%d\n", ans);
37     return 0;
38 }

```

2.7 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      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Head[u] = tot;
12 }
13 void DFS(int x) {
14     s.push(x);
15     flag[x] = true;
16     low[x] = dfn[x] = ++idx;
17     for (int i = Head[x], j; i; i = Next[i])
18         if (!dfn[j = Link[i]]) {
19             DFS(j);
20             low[x] = min(low[x], low[j]);
21         } else if (flag[j])
22             low[x] = min(low[x], dfn[j]);
23     if (low[x] == dfn[x]) {
24         int t;
25         num++;
26         do {
27             t = s.top();
28             s.pop();
29             flag[t] = false;
30             sub[num].push_back(t);
31         } while (t != x);
32     }
33 }
34 int main() {
35     scanf("%d%d", &n, &m);
36     for (int i = 1; i <= m; i++) {
37         scanf("%d%d", &u, &v);
38         AddEdge(u, v);

```

```

39     }
40     for (int i = 1; i <= n; i++)
41         if (!dfn[i])
42             DFS(i);
43     printf("%d\n", num);
44     for (int i = 1; i <= num; i++) {
45         for (int j = 0; j < sub[i].size() - 1; j++)
46             printf("%d_", sub[i][j]);
47         printf("%d\n", sub[i][sub[i].size() - 1]);
48     }
49     return 0;
50 }

```

2.8 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      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Head[u] = tot;
12 }
13 void DFS(int x, int y) {
14     s.push(x);
15     flag[x] = true;
16     low[x] = dfn[x] = ++idx;
17     for (int i = Head[x], j; i; i = Next[i]) {
18         if ((j = Link[i]) == y)
19             continue;
20         if (!dfn[j]) {
21             DFS(j, x);
22             low[x] = min(low[x], low[j]);
23         } else if (flag[j])
24             low[x] = min(low[x], dfn[j]);
25     }
26     if (x != y && low[x] >= dfn[y]) {
27         int t;
28         num++;
29         do {
30             t = s.top();
31             s.pop();
32             flag[t] = false;
33             sub[num].push_back(t);
34         } while (t != y);
35         s.push(y);
36         flag[y] = true;
37     }
38 }
39 int main() {
40     scanf("%d%d", &n, &m);
41     for (int i = 1; i <= m; i++) {
42         scanf("%d%d", &u, &v);
43         AddEdge(u, v);
44         AddEdge(v, u);
45     }
46     for (int i = 1; i <= n; i++)
47         if (!dfn[i]) {
48             DFS(i, i);
49             s.pop();
50             flag[i] = false;
51         }
52     printf("%d\n", num);
53     for (int i = 1; i <= num; i++) {
54         for (int j = 0; j < sub[i].size() - 1; j++)
55             printf("%d_", sub[i][j]);
56         printf("%d\n", sub[i][sub[i].size() - 1]);

```

```

57     }
58     return 0;
59 }

```

2.9 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      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Head[u] = tot;
12 }
13 void DFS(int x, int y) {
14     s.push(x);
15     flag[x] = true;
16     low[x] = dfn[x] = ++idx;
17     for (int i = Head[x], j; i; i = Next[i]) {
18         if ((j = Link[i]) == y)
19             continue;
20         if (!dfn[j]) {
21             DFS(j, x);
22             low[x] = min(low[x], low[j]);
23         } else if (flag[j])
24             low[x] = min(low[x], dfn[j]);
25     }
26     if (low[x] > dfn[y]) {
27         int t;
28         num++;
29         do {
30             t = s.top();
31             s.pop();
32             flag[t] = false;
33             sub[num].push_back(t);
34         } while (t != x);
35     }
36 }
37 int main() {
38     scanf("%d%d", &n, &m);
39     for (int i = 1; i <= m; i++) {
40         scanf("%d%d", &u, &v);
41         AddEdge(u, v);
42         AddEdge(v, u);
43     }
44     for (int i = 1; i <= n; i++)
45         if (!dfn[i]) {
46             DFS(i, i);
47             num++;
48             while (!s.empty()) {
49                 flag[s.top()] = false;
50                 sub[num].push_back(s.top());
51                 s.pop();
52             }
53         }
54     printf("%d\n", num);
55     for (int i = 1; i <= num; i++) {
56         for (int j = 0; j < sub[i].size() - 1; j++)
57             printf("%d□", sub[i][j]);
58         printf("%d\n", sub[i][sub[i].size() - 1]);
59     }
60     return 0;
61 }

```

2.10 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      Next[++tot] = Head[u];
8      Link[tot] = v;
9      Head[u] = tot;
10 }
11 bool DFS(int x) {
12     for (int i = Head[x], j; i; i = Next[i])
13         if (!flag[j = Link[i]]) {
14             flag[j] = true;
15             if (p[j] == 0 || DFS(p[j])) {
16                 p[j] = x;
17                 return true;
18             }
19         }
20     return false;
21 }
22 int main() {
23     scanf("%d%d%d", &n, &m, &k);
24     for (int i = 1; i <= k; i++) {
25         scanf("%d%d", &u, &v);
26         AddEdge(u, v);
27     }
28     for (int i = 1; i <= n; i++) {
29         memset(flag, false, sizeof(flag));
30         if (DFS(i))
31             ans++;
32     }
33     printf("%d\n", ans);
34     return 0;
35 }

```

2.11 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      fx[x] = true;
8      for (int i = 1, t; i <= m; i++)
9          if (!fy[i]) {
10             t = lx[x] + ly[i] - a[x][i];
11             if (!t) {
12                 fy[i] = true;
13                 if (p[i] == 0 || DFS(p[i])) {
14                     p[i] = x;
15                     return true;
16                 }
17             } else
18                 slack[i] = min(slack[i], t);
19         }
20     return false;
21 }
22 bool Find(int x) {
23     memset(fx, false, sizeof(fx));
24     memset(fy, false, sizeof(fy));
25     return DFS(x);
26 }
27 int main() {
28     scanf("%d%d", &n, &m);
29     for (int i = 1; i <= n; i++)
30         for (int j = 1; j <= m; j++)
31             scanf("%d", &a[i][j]);
32     for (int i = 1; i <= n; i++) {

```

```

33     lx[i] = INT_MIN;
34     for (int j = 1; j <= m; j++)
35         lx[i] = max(lx[i], a[i][j]);
36 }
37 for (int i = 1; i <= n; i++) {
38     for (int j = 1; j <= m; j++)
39         slack[j] = INT_MAX;
40     while (!Find(i)) {
41         t = INT_MAX;
42         for (int j = 1; j <= m; j++)
43             if (!fy[j])
44                 t = min(t, slack[j]);
45         for (int j = 1; j <= n; j++)
46             if (fx[j])
47                 lx[j] -= t;
48         for (int j = 1; j <= m; j++)
49             if (fy[j])
50                 ly[j] += t;
51         else
52             slack[j] -= t;
53     }
54 }
55 for (int i = 1; i <= m; i++)
56     if (p[i])
57         ans += a[p[i]][i];
58 printf("%d\n", ans);
59 return 0;
60 }

```

2.12 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      Next[++tot] = Head[u];
9      Link[tot] = v;
10     Rest[tot] = r;
11     Head[u] = tot;
12 }
13 bool BFS() {
14     for (int i = 1; i <= n; i++)
15         d[i] = INT_MAX;
16     d[S] = 0;
17     q.push(S);
18     while (!q.empty()) {
19         int now = q.front();
20         q.pop();
21         for (int i = Head[now], j; i; i = Next[i])
22             if (Rest[i] > 0 && d[now] + 1 < d[j = Link[i]]) {
23                 d[j] = d[now] + 1;
24                 q.push(j);
25             }
26     }
27     return d[T] < INT_MAX;
28 }
29 bool DFS(int x) {
30     if (x == T) {
31         int tmp = INT_MAX;
32         for (int i = T; i != S; i = From[i])
33             tmp = min(tmp, Rest[Edge[i]]);
34         for (int i = T; i != S; i = From[i]) {
35             Rest[Edge[i]] -= tmp;
36             Rest[Edge[i] ^ 1] += tmp;
37         }
38         ans += tmp;
39         return true;
40     }

```

```

41     for (int &i = cur[x], j; i; i = Next[i])
42         if (Rest[i] > 0 && d[x] + 1 == d[j = Link[i]]) {
43             From[j] = x;
44             Edge[j] = i;
45             if (DFS(j))
46                 return true;
47         }
48     return false;
49 }
50 int main() {
51     scanf("%d%d%d%d", &n, &m, &S, &T);
52     tot = 1;
53     for (int i = 1; i <= m; i++) {
54         scanf("%d%d%d", &u, &v, &r);
55         AddEdge(u, v, r);
56         AddEdge(v, u, 0);
57     }
58     while (BFS()) {
59         memcpy(cur, Head, sizeof(cur));
60         while (DFS(S));
61     }
62     printf("%d\n", ans);
63     return 0;
64 }

```

2.13 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      Next[++tot] = Head[u];
10     Link[tot] = v;
11     Rest[tot] = r;
12     Cost[tot] = c;
13     Head[u] = tot;
14 }
15 bool BFS() {
16     for (int i = 1; i <= n; i++)
17         d[i] = INT_MAX;
18     d[S] = 0;
19     q.push(S);
20     flag[S] = true;
21     while (!q.empty()) {
22         int now = q.front();
23         q.pop();
24         flag[now] = false;
25         for (int i = Head[now], j; i; i = Next[i])
26             if (Rest[i] > 0 && d[now] + Cost[i] < d[j = Link[i]]) {
27                 d[j] = d[now] + Cost[i];
28                 From[j] = now;
29                 Edge[j] = i;
30                 if (!flag[j]) {
31                     q.push(j);
32                     flag[j] = true;
33                 }
34             }
35     }
36     return d[T] < INT_MAX;
37 }
38 int main() {
39     scanf("%d%d%d%d", &n, &m, &S, &T);
40     tot = 1;
41     for (int i = 1; i <= m; i++) {
42         scanf("%d%d%d%d", &u, &v, &r, &c);
43         AddEdge(u, v, r, c);
44         AddEdge(v, u, 0, -c);

```

```

45     }
46     while (BFS()) {
47         tmp = INT_MAX;
48         sum = 0;
49         for (int i = T; i != S; i = From[i]) {
50             tmp = min(tmp, Rest[Edge[i]]);
51             sum += Cost[Edge[i]];
52         }
53         for (int i = T; i != S; i = From[i]) {
54             Rest[Edge[i]] -= tmp;
55             Rest[Edge[i] ^ 1] += tmp;
56         }
57         ans1 += tmp;
58         ans2 += tmp * sum;
59     }
60     printf("%d_ %d\n", ans1, ans2);
61     return 0;
62 }

```

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     Next[++tot] = Head[u];
11     Link[tot] = v;
12     Head[u] = tot;
13 }
14 int Get(int x) {
15     if (a[x] != x)
16         a[x] = Get(a[x]);
17     return a[x];
18 }
19 void DFS(int x) {
20     flag[x] = true;
21     a[x] = x;
22     for (int i = 0; i < Q[x].size(); i++)
23         ans[Q[x][i].se] = Get(a[Q[x][i].fi]);
24     for (int i = Head[x], j; i; i = Next[i])
25         if (!flag[j = Link[i]]) {
26             DFS(j);
27             a[j] = x;
28         }
29 }
30 int main() {
31     scanf("%d", &n);
32     for (int i = 1; i < n; i++) {
33         scanf("%d%d", &u, &v);
34         AddEdge(u, v);
35         AddEdge(v, u);
36     }
37     scanf("%d", &m);
38     for (int i = 1; i <= m; i++) {
39         scanf("%d%d", &u, &v);
40         Q[u].push_back({v, i});
41         Q[v].push_back({u, i});
42     }
43     DFS(1);
44     for (int i = 1; i <= m; i++)
45         printf("%d\n", ans[i]);
46     return 0;
47 }

```

3.2 树链剖分

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 100001;
4 int n, m, u, v, tot, num;
5 int d[N], f[N], s[N], son[N], top[N], idx[N], key[N], Head[N], Next[N << 1], Link[N << 1];
6 inline void AddEdge(int u, int v) {
7     Next[++tot] = Head[u];
8     Link[tot] = v;
9     Head[u] = tot;
10 }
11 void DFS1(int x) {
12     d[x] = d[f[x]] + 1;
13     s[x] = 1;
14     for (int i = Head[x], j; i; i = Next[i])
15         if (!d[j = Link[i]]) {
16             f[j] = x;
17             DFS1(j);
18             s[x] += s[j];
19             if (s[j] > s[son[x]])
20                 son[x] = j;
21         }
22 }
23 void DFS2(int x) {
24     top[x] = x == son[f[x]] ? top[f[x]] : x;
25     key[idx[x] = ++num] = x;
26     if (son[x])
27         DFS2(son[x]);
28     for (int i = Head[x], j; i; i = Next[i]) {
29         j = Link[i];
30         if (f[j] == x && j != son[x])
31             DFS2(j);
32     }
33 }
34 int LCA(int x, int y) {
35     int u, v;
36     while ((u = top[x]) != (v = top[y]))
37         if (d[u] > d[v])
38             x = f[u];
39         else
40             y = f[v];
41     if (d[x] > d[y])
42         swap(x, y);
43     return x;
44 }
45 int main() {
46     scanf("%d", &n);
47     for (int i = 1; i < n; i++) {
48         scanf("%d%d", &u, &v);
49         AddEdge(u, v);
50         AddEdge(v, u);
51     }
52     DFS1(1);
53     DFS2(1);
54     scanf("%d", &m);
55     for (int i = 1; i <= m; i++) {
56         scanf("%d%d", &u, &v);
57         printf("%d\n", LCA(u, v));
58     }
59     return 0;
60 }
```

4 数据结构

4.1 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     if (a[x] != x)
7         a[x] = Find(a[x]);
8     return a[x];
9 }
10 void Merge(int x, int y) {
11     if ((x = Find(x)) == (y = Find(y)))
12         return;
13     b[x] < b[y] ? a[x] = y : a[y] = x;
14     if (b[x] == b[y])
15         b[x]++;
16 }
17 int main() {
18     for (int i = 1; i <= n; i++)
19         a[i] = i;
20     return 0;
21 }

```

4.2 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     int p = 1, t;
7     for (int i = 0; s[i]; i++) {
8         t = s[i] - 97;
9         if (!son[p][t])
10             son[p][t] = ++pos;
11         p = son[p][t];
12     }
13     num[p]++;
14 }
15 int Find(char s[]) {
16     int p = 1, t;
17     for (int i = 0; s[i]; i++) {
18         t = s[i] - 97;
19         if (!son[p][t])
20             return 0;
21         p = son[p][t];
22     }
23     return num[p];
24 }
25 int main() {
26     pos = 1;
27     return 0;
28 }

```

4.3 左偏树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 100001;
4 int num, l[N], r[N], d[N], key[N];
5 int Merge(int p, int q) {
6     if (!p)
7         return q;
8     if (!q)
9         return p;
10    if (key[p] > key[q])
11        swap(p, q);
12    r[p] = Merge(r[p], q);
13    if (d[l[p]] < d[r[p]])
14        swap(l[p], r[p]);
15    d[p] = d[r[p]] + 1;
16    return p;
17 }

```

```

18 void Push(int &p, int x) {
19     key[++num] = x;
20     p = Merge(p, num);
21 }
22 void Pop(int &p) {
23     p = Merge(l[p], r[p]);
24 }
25 int Top(int p) {
26     return key[p];
27 }
28 int main() {
29     d[0] = -1;
30     return 0;
31 }

```

4.4 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      for (; x <= n; x += x & -x)
7          sum[x] += y;
8  }
9  int Sum(int x) {
10     int ans = 0;
11     for (; x; x -= x & -x)
12         ans += sum[x];
13     return ans;
14 }
15 int main() {
16     return 0;
17 }

```

4.5 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      for (SIZE = 1; SIZE < n + 2; SIZE <= 1);
7      for (int i = 1; i <= n; i++)
8          sum[SIZE + i] = a[i];
9      for (int i = SIZE - 1; i; i--)
10         sum[i] = sum[i << 1] + sum[(i << 1) + 1];
11 }
12 void Add(int x, int y) {
13     for (x += SIZE; x; x >>= 1)
14         sum[x] += y;
15 }
16 int Sum(int x, int y) {
17     int ans = 0;
18     for (x += SIZE - 1, y += SIZE + 1; x ^ y ^ 1; x >>= 1, y >>= 1) {
19         if ((x & 1) == 0)
20             ans += sum[x ^ 1];
21         if ((y & 1) == 1)
22             ans += sum[y ^ 1];
23     }
24     return ans;
25 }
26 int main() {
27     return 0;
28 }

```

4.6 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], ll[N << 1], rr[N << 1], sum[N << 1], lab[N << 1];
5 inline void Label(int p, int x) {
6     sum[p] += (rr[p] - ll[p] + 1) * x;
7     lab[p] += x;
8 }
9 inline void Down(int p) {
10     if (ll[p] < rr[p]) {
11         Label(l[p], lab[p]);
12         Label(r[p], lab[p]);
13     }
14     lab[p] = 0;
15 }
16 inline void Up(int p) {
17     sum[p] = sum[l[p]] + sum[r[p]];
18 }
19 void Build(int p, int x, int y) {
20     ll[p] = x;
21     rr[p] = y;
22     if (x == y) {
23         sum[p] = a[x];
24         return;
25     }
26     int z = x + y >> 1;
27     Build(l[p] = ++num, x, z);
28     Build(r[p] = ++num, z + 1, y);
29     Up(p);
30 }
31 void Add(int p, int x, int y, int z) {
32     Down(p);
33     if (ll[p] == x && rr[p] == y) {
34         Label(p, z);
35         return;
36     }
37     if (y < ll[r[p]])
38         Add(l[p], x, y, z);
39     else if (x > rr[l[p]])
40         Add(r[p], x, y, z);
41     else {
42         Add(l[p], x, rr[l[p]], z);
43         Add(r[p], ll[r[p]], y, z);
44     }
45     Up(p);
46 }
47 int Sum(int p, int x, int y) {
48     Down(p);
49     if (ll[p] == x && rr[p] == y)
50         return sum[p];
51     if (y < ll[r[p]])
52         return Sum(l[p], x, y);
53     else if (x > rr[l[p]])
54         return Sum(r[p], x, y);
55     else
56         return Sum(l[p], x, rr[l[p]]) + Sum(r[p], ll[r[p]], y);
57 }
58 int main() {
59     num = 1;
60     return 0;
61 }
```

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      if (l[p]) {
8          key[l[p]] += lab[p];
9          lab[l[p]] += lab[p];
10         sum[l[p]] += s[l[p]] * lab[p];
11         if (flag[p]) {
12             flag[l[p]] = !flag[l[p]];
13             swap(l[l[p]], r[l[p]]);
14         }
15     }
16     if (r[p]) {
17         key[r[p]] += lab[p];
18         lab[r[p]] += lab[p];
19         sum[r[p]] += s[r[p]] * lab[p];
20         if (flag[p]) {
21             flag[r[p]] = !flag[r[p]];
22             swap(l[r[p]], r[r[p]]);
23         }
24     }
25     lab[p] = 0;
26     flag[p] = false;
27 }
28 inline void Up(int p) {
29     s[p] = s[l[p]] + s[r[p]] + 1;
30     sum[p] = sum[l[p]] + sum[r[p]] + key[p];
31 }
32 inline void L(int p) {
33     int t = f[p];
34     if (r[t] = l[p])
35         f[l[p]] = t;
36     if (f[p] = f[t])
37         t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
38     f[t] = p;
39     l[p] = t;
40 }
41 inline void R(int p) {
42     int t = f[p];
43     if (l[t] = r[p])
44         f[r[p]] = t;
45     if (f[p] = f[t])
46         t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
47     f[t] = p;
48     r[p] = t;
49 }
50 void Splay(int p, int T) {
51     for (int q, t; (q = f[p]) != T; )
52         if (f[q] == T) {
53             p == l[q] ? R(p) : L(p);
54             Up(q), Up(p);
55         } else {
56             t = f[q];
57             if (p == l[q])
58                 q == l[t] ? (R(q), R(p)) : (R(p), L(p));
59             else
60                 q == r[t] ? (L(q), L(p)) : (L(p), R(p));
61             Up(t), Up(q), Up(p);
62         }
63     if (!T)
64         root = p;
65 }
66 int Select(int x) {
67     int p = root, t = s[l[root]];
68     Down(p);
69     while (x != t + 1) {
70         if (x < t + 1)
71             t -= s[r[p = l[p]]] + 1;
72         else
73             t += s[l[p = r[p]]] + 1;
74         Down(p);
75     }
76     return p;
77 }
78 void Insert(int x, int y) {

```

```

79     int p = Select(x + 1);
80     Splay(p, 0);
81     Down(p);
82     for (p = r[p]; l[p]; p = l[p])
83         Down(p);
84     Down(p);
85     l[p] = ++pos;
86     f[pos] = p;
87     sum[pos] = key[pos] = y;
88     Splay(pos, 0);
89 }
90 void Delete(int x) {
91     int p = Select(x + 1);
92     Splay(p, 0);
93     Down(p);
94     for (p = l[p]; r[p]; p = r[p])
95         Down(p);
96     Down(p);
97     f[r[root]] = p;
98     r[p] = r[root];
99     f[l[root]] = 0;
100    Splay(p, 0);
101 }
102 void Add(int x, int y, int z) {
103     Splay(Select(x), 0);
104     Splay(Select(y + 2), root);
105     key[l[r[root]]] += z;
106     lab[l[r[root]]] += z;
107     sum[l[r[root]]] += s[l[r[root]]] * z;
108     Up(r[root]), Up(root);
109 }
110 void Reverse(int x, int y) {
111     Splay(Select(x), 0);
112     Splay(Select(y + 2), root);
113     flag[l[r[root]]] = !flag[l[r[root]]];
114     swap(l[l[r[root]]], r[l[r[root]]]);
115     Up(r[root]), Up(root);
116 }
117 int Sum(int x, int y) {
118     Splay(Select(x), 0);
119     Splay(Select(y + 2), root);
120     return sum[l[r[root]]];
121 }
122 int main() {
123     root = 1;
124     pos = 2;
125     r[1] = s[1] = 2;
126     f[2] = s[2] = 1;
127     return 0;
128 }

```

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], s[N], num[N], key[N];
5  inline void L(int &p) {
6      int t = r[p];
7      r[p] = l[t];
8      l[t] = p;
9      s[t] = s[p];
10     s[p] = s[l[p]] + s[r[p]] + num[p];
11     p = t;
12 }
13 inline void R(int &p) {
14     int t = l[p];
15     l[p] = r[t];
16     r[t] = p;
17     s[t] = s[p];
18     s[p] = s[l[p]] + s[r[p]] + num[p];

```

```

19     p = t;
20 }
21 void Fix(int &p, bool flag) {
22     if (flag) {
23         if (s[l[r[p]]] > s[l[p]])
24             R(r[p]), L(p);
25         else if (s[r[r[p]]] > s[l[p]])
26             L(p);
27         else
28             return;
29     } else {
30         if (s[r[l[p]]] > s[r[p]])
31             L(l[p]), R(p);
32         else if (s[l[l[p]]] > s[r[p]])
33             R(p);
34         else
35             return;
36     }
37     Fix(l[p], 0);
38     Fix(r[p], 1);
39     Fix(p, 0);
40     Fix(p, 1);
41 }
42 void Insert(int &p, int x) {
43     if (p) {
44         s[p]++;
45         if (x == key[p])
46             num[p]++;
47         else {
48             Insert(x < key[p] ? l[p] : r[p], x);
49             Fix(p, x > key[p]);
50         }
51     } else {
52         p = ++pos;
53         key[p] = x;
54         s[p] = num[p] = 1;
55     }
56 }
57 void Delete(int x) {
58     int p, q, t, tmp;
59     for (p = root, t = 0; x != key[p]; p = x < key[p] ? l[p] : r[p]) {
60         s[p]--;
61         t = p;
62     }
63     s[p]--;
64     if (!(--num[p]))
65         if (l[p]) {
66             for (q = l[p], t = p; r[q]; q = r[q])
67                 t = q;
68             for (tmp = l[p]; r[tmp]; tmp = r[tmp])
69                 s[tmp] -= num[q];
70             key[p] = key[q];
71             num[p] = num[q];
72             q == l[t] ? l[t] = l[q] : r[t] = l[q];
73         } else if (t)
74             p == l[t] ? l[t] = r[p] : r[t] = r[p];
75         else
76             root = r[p];
77 }
78 int Rank(int x) {
79     int p = root, t = s[l[root]];
80     while (key[p] != x)
81         if (x < key[p]) {
82             p = l[p];
83             t -= s[r[p]] + num[p];
84         } else {
85             t += num[p];
86             p = r[p];
87             t += s[l[p]];
88         }
89     return t + 1;
90 }
91 int Select(int x) {

```

```

92     int p = root, t = s[l[root]];
93     while (x < t + 1 || x > t + num[p])
94         if (x < t + 1) {
95             p = l[p];
96             t -= s[r[p]] + num[p];
97         } else {
98             t += num[p];
99             p = r[p];
100            t += s[l[p]];
101        }
102    return key[p];
103 }
104 int Pred(int x) {
105     int p = root, t;
106     while (p)
107         if (x > key[p]) {
108             t = p;
109             p = r[p];
110         } else
111             p = l[p];
112     return key[t];
113 }
114 int Succ(int x) {
115     int p = root, t;
116     while (p)
117         if (x < key[p]) {
118             t = p;
119             p = l[p];
120         } else
121             p = r[p];
122     return key[t];
123 }
124 int main() {
125     return 0;
126 }

```

5 数学

5.1 快速幂

```

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

```

5.2 Euclid

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int a, b;
4  int gcd(int a, int b) {
5      return b ? gcd(b, a % b) : a;
6  }
7  int main() {
8      scanf("%d%d", &a, &b);
9      printf("%d\n", gcd(a, b));
10     return 0;

```

```
11 }
```

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      if (b) {
6          int t, xt, yt;
7          t = gcd(b, a % b, xt, yt);
8          x = yt;
9          y = xt - a / b * yt;
10         return t;
11     } else {
12         x = 1;
13         y = 0;
14         return a;
15     }
16 }
17 int main() {
18     scanf("%d%d", &a, &b);
19     t = gcd(a, b, x, y);
20     printf("%d□%d□%d\n", x, y, t);
21     return 0;
22 }
```

5.4 Miller-Rabin 测试

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  ll n;
5  ll Mul(ll a, ll b, ll MOD) {
6      ll ans = 0;
7      while (b) {
8          if (b & 1)
9              ans = (ans + a) % MOD;
10         a = (a << 1) % MOD;
11         b >>= 1;
12     }
13     return ans;
14 }
15 ll Pow(ll a, ll b, ll MOD) {
16     ll ans = 1;
17     while (b) {
18         if (b & 1)
19             ans = Mul(ans, a, MOD);
20         a = Mul(a, a, MOD);
21         b >>= 1;
22     }
23     return ans;
24 }
25 bool Judge(ll p) {
26     if (p < 2)
27         return false;
28     int num = 0;
29     ll t = p - 1, t1, t2;
30     for (; !(t & 1); t >>= 1)
31         num++;
32     for (int i = 0; i < 5; i++) {
33         t1 = Pow(rand() % (p - 1) + 1, t, p);
34         for (int j = 0; t1 != 1 && j < num; j++) {
35             t2 = Mul(t1, t1, p);
36             if (t1 != 1 && t1 != p - 1 && t2 == 1)
37                 return false;
38             t1 = t2;
39         }
40         if (t1 != 1)
```



```

41         return false;
42     }
43     return true;
44 }
45 int main() {
46     srand(time(NULL));
47     scanf("%lld", &n);
48     puts(Judge(n) ? "YES" : "NO");
49     return 0;
50 }

```

5.5 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      scanf("%d", &n);
8      fai[1] = miu[1] = 1;
9      for (int i = 2; i <= n; i++) {
10         if (!flag[i]) {
11             p[++num] = i;
12             fai[i] = i - 1;
13             miu[i] = -1;
14         }
15         for (int j = 1; j <= num; j++) {
16             if (i * p[j] > n)
17                 break;
18             flag[i * p[j]] = true;
19             if (i % p[j] == 0) {
20                 fai[i * p[j]] = fai[i] * p[j];
21                 miu[i * p[j]] = 0;
22                 break;
23             } else {
24                 fai[i * p[j]] = fai[i] * (p[j] - 1);
25                 miu[i * p[j]] = -miu[i];
26             }
27         }
28     }
29     printf("%d\n", num);
30     for (int i = 1; i < num; i++)
31         printf("%d□", p[i]);
32     printf("%d\n", p[num]);
33     for (int i = 1; i < n; i++)
34         printf("%d□", fai[i]);
35     printf("%d\n", fai[n]);
36     for (int i = 1; i < n; i++)
37         printf("%d□", miu[i]);
38     printf("%d\n", miu[n]);
39     return 0;
40 }

```

5.6 Gauss 消元

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 101, M = 101;
4  const double EPS = 1e-5;
5  int n, m, tmp;
6  double a[M][N + 1], t[M][N + 1], temp[N + 1];
7  bool flag;
8  int main() {
9      scanf("%d%d", &n, &m);
10     for (int i = 1; i <= m; i++)
11         for (int j = 1; j <= n + 1; j++)
12             scanf("%lf", &t[i][j]);
13     for (int i = 1; i <= n; i++) {

```

```

14     tmp = 0;
15     for (int j = 1; j <= m; j++) {
16         flag = false;
17         for (int k = 1; !flag && k <= n; k++)
18             if (fabs(t[j][k]) > EPS)
19                 flag = true;
20         if (flag)
21             memcpy(a[++tmp], t[j], sizeof(t[j]));
22         else if (fabs(t[j][n + 1]) > EPS) {
23             puts("No Solution");
24             return 0;
25         }
26     }
27     if ((m = tmp) < n) {
28         puts("Infinite Solutions");
29         return 0;
30     }
31     flag = false;
32     for (int j = i; !flag && j <= m; j++)
33         if (fabs(a[j][i]) > EPS) {
34             memcpy(temp, a[i], sizeof(temp));
35             memcpy(a[i], a[j], sizeof(temp));
36             memcpy(a[j], temp, sizeof(temp));
37             flag = true;
38         }
39     if (!flag) {
40         puts("Infinite Solutions");
41         return 0;
42     }
43     for (int j = i + 1; j <= n + 1; j++)
44         a[i][j] /= a[i][i];
45     a[i][i] = 1;
46     for (int j = i + 1; j <= m; j++) {
47         for (int k = i + 1; k <= n + 1; k++)
48             a[j][k] -= a[i][k] * a[j][i];
49         a[j][i] = 0;
50     }
51     memcpy(t, a, sizeof(a));
52 }
53 for (int i = n - 1; i; i--)
54     for (int j = i + 1; j <= n; j++)
55         a[i][n + 1] -= a[i][j] * a[j][n + 1];
56 for (int i = 1; i < n; i++)
57     printf("%f ", a[i][n + 1]);
58 printf("%f\n", a[n][n + 1]);
59 return 0;
60 }

```

5.7 快速 Fourier 变换

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100001;
4  const double PI = acos(-1);
5  int n, m, LENG, SIZE;
6  double t;
7  complex<double> a[N << 2], b[N << 2], c[N << 2];
8  complex<double> ya[N << 2], yb[N << 2], yc[N << 2], yt[N << 2];
9  void DFT(complex<double> a[], complex<double> y[], bool flag) {
10     for (int i = 0; i < SIZE; i++) {
11         int t = 0;
12         for (int j = 0; j < LENG; j++)
13             t += (i >> j & 1) << LENG - j - 1;
14         y[i] = a[t];
15     }
16     for (int t = 1; t < SIZE; t <= 1) {
17         double tmp = (flag ? -1 : 1) * PI / t;
18         for (int i = 0; i < SIZE; i += t << 1)
19             for (int j = 0; j < t; j++) {
20                 yt[i + j] = y[i + j] + polar(1.0, tmp * j) * y[i + t + j];
21                 yt[i + t + j] = y[i + j] + polar(1.0, tmp * (t + j)) * y[i + t + j];

```

```

22     }
23     memcpy(y, yt, sizeof(yt));
24 }
25 if (flag)
26     for (int i = 0; i < SIZE; i++)
27         y[i] /= SIZE;
28 }
29 int main() {
30     scanf("%d%d", &n, &m);
31     for (int i = 0; i < n; i++) {
32         scanf("%lf", &t);
33         a[i] = {t, 0};
34     }
35     for (int i = 0; i < m; i++) {
36         scanf("%lf", &t);
37         b[i] = {t, 0};
38     }
39     for (LENG = 0, SIZE = 1; SIZE < n + m - 1; LENG++, SIZE <= 1);
40     DFT(a, ya, false);
41     DFT(b, yb, false);
42     for (int i = 0; i < SIZE; i++)
43         yc[i] = ya[i] * yb[i];
44     DFT(yc, c, true);
45     for (int i = 0; i < n + m - 2; i++)
46         printf("%f□", c[i].real());
47     printf("%f\n", c[n + m - 2].real());
48     return 0;
49 }

```

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     return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
14 }
15 inline bool Judge(Segment a, Segment b) {
16     if (max(a.x1, a.x2) < min(b.x1, b.x2) || max(b.x1, b.x2) < min(a.x1, a.x2))
17         return false;
18     if (max(a.y1, a.y2) < min(b.y1, b.y2) || max(b.y1, b.y2) < min(a.y1, a.y2))
19         return false;
20     return Cross(a.x, a.y, b.x) * Cross(a.x, a.y, b.y) <= 0 &&
21         Cross(b.x, b.y, a.x) * Cross(b.x, b.y, a.y) <= 0;
22 }
23 int main() {
24     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);
25     puts(Judge(a, b) ? "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     return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
12 }
13 int main() {
14     scanf("%d", &n);
15     for (int i = 1; i <= n; i++)
16         scanf("%lf%lf", &p[i].x, &p[i].y);
17     for (int i = 3; i <= n; i++)
18         ans += Cross(p[1], p[i - 1], p[i]);
19     printf("%f\n", ans / 2);
20     return 0;
21 }

```

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     return x * x;
11 }
12 inline double Dist(Point a, Point b) {
13     return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
14 }
15 inline double Cross(Point a, Point b, Point c) {
16     return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
17 }
18 inline bool cmp(Point a, Point b) {
19     return Cross(p[0], a, b) > 0 || Cross(p[0], a, b) == 0 && Dist(p[0], a) < Dist(p[0], b);
20 }
21 int main() {
22     scanf("%d", &n);
23     for (int i = 0; i < n; i++) {
24         scanf("%lf%lf", &p[i].x, &p[i].y);
25         if (p[i].y < p[0].y || p[i].y == p[0].y && p[i].x < p[0].x)
26             swap(p[0], p[i]);
27     }
28     sort(p + 1, p + n, cmp);
29     s[top = 1] = p[0];
30     for (int i = 1; i < n; i++) {
31         for (; top > 1 && Cross(s[top - 1], s[top], p[i]) < 0; top--);
32         s[++top] = p[i];
33     }
34     for (; top > 2 && Cross(s[top - 1], s[top], s[1]) < 0; top--);
35     printf("%d\n", top);
36     for (int i = 1; i <= top; i++)
37         printf("%f_ %f\n", s[i].x, s[i].y);
38     return 0;
39 }

```

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  const double EPS = 1e-5;
8  int n;

```

```

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

```