ACM 模板

Wajov

September 8, 2017

目录

1 字符串算法

1.1 最小表示

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, x, y, t, ans;
char s[N + 10];
int main()
{
    scanf("%s", s + 1);
    n = strlen(s + 1);
    x = 1;
    y = 2;
    for (int i = 0; x <= n && y <= n && i <= n; )
    {
        t = s[(x + i - 1) % n + 1] - s[(y + i - 1) % n + 1];
        if (!t)
            i++;
        else
        {
            t > 0 ? x += i + 1 : y += i + 1;
            if (x == y)
                y++;
            i = 0;
        }
    }
    ans = min(x, y);
    for (int i = ans; i <= n; i++)
        putchar(s[i]);
    for (int i = 1; i < ans; i++)
        putchar(s[i]);
    puts("");
    return 0;
}
```

1.2 Manacher

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, ans, p[N << 1];
char c, s[N << 1];
int main()
{
    s[0] = '$';
    while ((c = getchar()) != '\n')
    {
        s[++n] = '#';
        s[++n] = c;
    }
    s[++n] = '#';
    for (int i = 1, j = 0; i <= n; i++)
    {
        p[i] = i < j + p[j] ? min(p[(j << 1) - i], j + p[j] - i) : 1;
        while (s[i + p[i]] == s[i - p[i]])
            p[i]++;
        if (i + p[i] > j + p[j])
            j = i;
        ans = max(ans, p[i] - 1);
    }
    printf("%d\n", ans);
    return 0;
}
```

1.3 Knuth-Morris-Pratt

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, m, num, p[N], ans[N];
char a[N + 10], b[N + 10];
int main()
{
    scanf("%s%s", a + 1, b + 1);
    n = strlen(a + 1);
    m = strlen(b + 1);
    for (int i = 2, j = 0; i <= m; i++)
    {
        for (; j > 0 && b[j + 1] != b[i]; j = p[j]);
        if (b[j + 1] == b[i])
            j++;
        p[i] = j;
    }
    for (int i = 1, j = 0; i <= n; i++)
    {
        for (; j > 0 && b[j + 1] != a[i]; j = p[j]);
        if (b[j + 1] == a[i])
            j++;
        if (j == m)
        {
            ans[++num] = i - j + 1;
            j = p[j];
        }
    }
    for (int i = 1; i < num; i++)
        printf("%d␣", ans[i]);
    printf("%d\n", ans[num]);
    return 0;
}
```

1.4 扩展 Knuth-Morris-Pratt

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, m, p[N], ex[N];
char a[N + 10], b[N + 10];
int main()
{
    scanf("%s%s", a + 1, b + 1);
    n = strlen(a + 1);
    m = strlen(b + 1);
    for (int i = 2, j = 0; i <= m; i++)
        {
            p[i] = i < j + p[j] ? min(p[i - j + 1], j + p[j] - i) : 0;
            for (; i + p[i] <= m && b[i + p[i]] == b[p[i] + 1]; p[i]++);
            if (i + p[i] > j + p[j])
                j = i;
        }
        for (int i = 1, j = 0; i <= n; i++)
        {
            ex[i] = i <= j + ex[j] ? min(p[i - j + 1], j + ex[j] - i) : 0;
            for (; i + ex[i] <= n && ex[i] < m && a[i + ex[i]] == b[ex[i] + 1]; ex[i]++);
            if (i + ex[i] > j + ex[j])
                j = i;
        }
    for (int i = 1; i < n; i++)
        printf("%d␣", ex[i]);
    printf("%d\n", ex[n]);
    return 0;
}
```

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, 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 }
```

1.6 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  {
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];
```

```cpp
    for (int i = n; i; i−−)
        tmp[id[i]] = sum[a[id[i]]]−−;
    for (int i = 1; i <= n; i++)
        id[tmp[i]] = i;
}
int main()
{
    scanf("%s", s + 1);
    n = strlen(s + 1);
    for (int i = 1; i <= n; i++)
        a[id[i] = i] = s[i] − 97;
    Sort(a, 25);
    for (int i = 1; i <= n; i <<= 1)
    {
        for (int j = 1, t = 0; j <= n; j++)
            rk[id[j]] = a[id[j]] == a[id[j − 1]] && b[id[j]] == b[id[j − 1]] ? t : ++t;
        for (int j = 1; j <= n; j++)
        {
            a[j] = rk[j];
            b[j] = rk[min(i + j, n + 1)];
        }
        Sort(b, n);
        Sort(a, n);
    }
    for (int i = 1; i < n; i++)
        printf("%d␣", rk[i]);
    printf("%d\n", rk[n]);
    return 0;
}
```

2 图算法

2.1 拓扑排序

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001, M = 1000001;
int n, m, u, v, tot, num, Head[N], Next[M], Link[M], ans[N];
bool flag[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS(int x)
{
    flag[x] = true;
    for (int i = Head[x], j; i; i = Next[i])
        if (!flag[j = Link[i]])
            DFS(j);
    ans[++num] = x;
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
    }
    for (int i = 1; i <= n; i++)
        if (!flag[i])
            DFS(i);
    for (int i = n; i > 1; i−−)
        printf("%d␣", ans[i]);
    printf("%d\n", ans[1]);
    return 0;
}
```

2.2 Floyd-Warshall

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 101;
int n, m, u, v, c, d[N][N];
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            d[i][j] = i == j ? 0 : INT_MAX >> 1;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        d[u][v] = d[v][u] = min(d[u][v], c);
    }
    for (int k = 1; k <= n; k++)
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j < n; j++)
            printf("%d␣", d[i][j] == INT_MAX >> 1 ? -1 : d[i][j]);
        printf("%d\n", d[i][n] == INT_MAX >> 1 ? -1 : d[i][n]);
    }
    return 0;
}
```

2.3 Floyd-Warshall（最小环）

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 101;
int n, m, u, v, c, t, num, Min, a[N][N], d[N][N], p[N][N], ans[N];
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
        {
            a[i][j] = i == j ? 0 : INT_MAX / 3;
            p[i][j] = i;
        }
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        a[u][v] = a[v][u] = min(a[u][v], c);
    }
    memcpy(d, a, sizeof(d));
    Min = INT_MAX / 3;
    for (int k = 1; k <= n; k++)
    {
        for (int i = 1; i < k; i++)
            for (int j = 1; j < i; j++)
                if (d[i][j] + a[i][k] + a[k][j] < Min)
                {
                    Min = d[i][j] + a[i][k] + a[k][j];
                    for (num = 0, t = j; t != i; t = p[i][t])
                        ans[++num] = t;
                    ans[++num] = i;
                    ans[++num] = k;
                }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                if (d[i][k] + d[k][j] < d[i][j])
                {
                    d[i][j] = d[i][k] + d[k][j];
                    p[i][j] = p[k][j];
                }
```

```
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], 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 (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          {
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, t; 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

```cpp
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 Dinic（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, tot, ans1, ans2;
5  int Head[N], cur[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                                 if (!flag[j])
33                                 {
34                                         q.push(j);
```

```
35                                         flag[j] = true;
36                                 }
37                         }
38                 }
39             return d[T] < INT_MAX;
40 }
41 bool DFS(int x)
42 {
43     if (x == T)
44     {
45         int tmp = INT_MAX, sum = 0;
46         for (int i = T; i != S; i = From[i])
47         {
48             tmp = min(tmp, Rest[Edge[i]]);
49             sum += Cost[Edge[i]];
50         }
51         for (int i = T; i != S; i = From[i])
52         {
53             Rest[Edge[i]] -= tmp;
54             Rest[Edge[i] ^ 1] += tmp;
55         }
56         ans1 += tmp;
57         ans2 += tmp * sum;
58         return true;
59     }
60     flag[x] = true;
61     for (int &i = cur[x], j; i; i = Next[i])
62     {
63         j = Link[i];
64         if (Rest[i] > 0 && !flag[j] && d[x] + Cost[i] == d[j])
65         {
66             From[j] = x;
67             Edge[j] = i;
68             if (DFS(j))
69             {
70                 flag[x] = false;
71                 return true;
72             }
73         }
74     }
75     flag[x] = false;
76     return false;
77 }
78 int main()
79 {
80     scanf("%d%d%d%d", &n, &m, &S, &T);
81     tot = 1;
82     for (int i = 1; i <= m; i++)
83     {
84         scanf("%d%d%d%d", &u, &v, &r, &c);
85         AddEdge(u, v, r, c);
86         AddEdge(v, u, 0, -c);
87     }
88     while (BFS())
89     {
90         memcpy(cur, Head, sizeof(cur));
91         while (DFS(S));
92     }
93     printf("%d %d\n", ans1, ans2);
94     return 0;
95 }
```

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;
```

```cpp
const int N = 1000001, M = 1000001;
int n, m, u, v, tot, Head[N], Next[N << 1], Link[N << 1], a[N], ans[M];
bool flag[N];
vector<pair<int, int> > Q[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
int Get(int x)
{
    if (a[x] != x)
        a[x] = Get(a[x]);
    return a[x];
}
void DFS(int x)
{
    flag[x] = true;
    a[x] = x;
    for (int i = 0; i < Q[x].size(); i++)
        ans[Q[x][i].se] = Get(a[Q[x][i].fi]);
    for (int i = Head[x], j; i; i = Next[i])
        if (!flag[j = Link[i]])
        {
            DFS(j);
            a[j] = x;
        }
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i < n; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
        AddEdge(v, u);
    }
    scanf("%d", &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        Q[u].push_back({v, i});
        Q[v].push_back({u, i});
    }
    DFS(1);
    for (int i = 1; i <= m; i++)
        printf("%d\n", ans[i]);
    return 0;
}
```

3.2 树链剖分

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int n, m, u, v, tot, num, d[N], f[N], s[N], son[N], top[N], idx[N], key[N], Head[N], Next[N << 1], Link[N << 1
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS1(int x)
{
    d[x] = d[f[x]] + 1;
    s[x] = 1;
    for (int i = Head[x], j; i; i = Next[i])
        if (!d[j = Link[i]])
        {
            f[j] = x;
```

```
19              DFS1(j);
20              s[x] += s[j];
21              if (s[j] > s[son[x]])
22                  son[x] = j;
23          }
24  }
25  void DFS2(int x)
26  {
27      top[x] = x == son[f[x]] ? top[f[x]] : x;
28      key[idx[x] = ++num] = x;
29      if (son[x])
30          DFS2(son[x]);
31      for (int i = Head[x], j; i; i = Next[i])
32      {
33          j = Link[i];
34          if (f[j] == x && j != son[x])
35              DFS2(j);
36      }
37  }
38  int LCA(int x, int y)
39  {
40      int u, v;
41      while ((u = top[x]) != (v = top[y]))
42          if (d[u] > d[v])
43              x = f[u];
44          else
45              y = f[v];
46      if (d[x] > d[y])
47          swap(x, y);
48      return x;
49  }
50  int main()
51  {
52      scanf("%d", &n);
53      for (int i = 1; i < n; i++)
54      {
55          scanf("%d%d", &u, &v);
56          AddEdge(u, v);
57          AddEdge(v, u);
58      }
59      DFS1(1);
60      DFS2(1);
61      scanf("%d", &m);
62      for (int i = 1; i <= m; i++)
63      {
64          scanf("%d%d", &u, &v);
65          printf("%d\n", LCA(u, v));
66      }
67      return 0;
68  }
```

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

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int n, SIZE, a[N], sum[N << 2];
void Build()
{
    for (SIZE = 1; SIZE < n + 2; SIZE <<= 1);
    for (int i = 1; i <= n; i++)
        sum[SIZE + i] = a[i];
    for (int i = SIZE - 1; i; i--)
        sum[i] = sum[i << 1] + sum[(i << 1) + 1];
}
void Add(int x, int y)
{
    for (x += SIZE; x; x >>= 1)
        sum[x] += y;
}
int Sum(int x, int y)
{
    int ans = 0;
    for (x += SIZE - 1, y += SIZE + 1; x ^ y ^ 1; x >>= 1, y >>= 1)
    {
        if ((x & 1) == 0)
            ans += sum[x ^ 1];
        if ((y & 1) == 1)
            ans += sum[y ^ 1];
    }
    return ans;
}
int main()
{
    return 0;
}
```

4.5 线段树

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int num, a[N], l[N << 1], r[N << 1], sum[N << 1], lab[N << 1];
inline void Label(int p, int x, int y, int z)
{
    sum[p] += (y - x + 1) * z;
    lab[p] += z;
}
inline void Down(int p, int x, int y)
{
    if (x < y)
    {
        int z = x + y >> 1;
        Label(l[p], x, z, lab[p]);
        Label(r[p], z + 1, y, lab[p]);
    }
    lab[p] = 0;
}
inline void Up(int p)
{
    sum[p] = sum[l[p]] + sum[r[p]];
}
void Build(int p, int x, int y)
{
    if (x == y)
    {
        sum[p] = a[x];
        return;
    }
    int z = x + y >> 1;
    Build(l[p] = ++num, x, z);
    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（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  }
```

4.7 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   }
```

4.8 节点大小平衡树

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int root, pos, l[N], r[N], f[N],  s[N], num[N], key[N];
inline void L(int p)
{
    int t = r[p];
    if (r[p] = l[t])
        f[l[t]] = p;
    l[t] = p;
    if (f[t] = f[p])
        p == l[f[p]] ? l[f[p]] = t : r[f[p]] = t;
    f[p] = t;
    s[t] = s[p];
    s[p] = s[l[p]] + s[r[p]] + num[p];
    if (p == root)
        root = t;
}
inline void R(int p)
{
    int t = l[p];
    if (l[p] = r[t])
        f[r[t]] = p;
    r[t] = p;
    if (f[t] = f[p])
        p == l[f[p]] ? l[f[p]] = t : r[f[p]] = t;
    f[p] = t;
    s[t] = s[p];
    s[p] = s[l[p]] + s[r[p]] + num[p];
    if (p == root)
        root = t;
}
void Fix(int p, bool flag)
{
    if (flag)
        if (s[l[r[p]]] > s[l[p]])
            R(r[p]), L(p);
        else
            if (s[r[r[p]]] > s[l[p]])
                L(p);
            else
                return;
    else
        if (s[r[l[p]]] > s[r[p]])
            L(l[p]), R(p);
        else
            if (s[l[l[p]]] > s[r[p]])
                R(p);
            else
                return;
    Fix(l[p], 0);
    Fix(r[p], 1);
    Fix(p, 0);
    Fix(p, 1);
}
void Insert(int p, int q, int x)
{
    if (!p)
    {
        p = ++pos;
        if (q)
            x < key[q] ? l[q] = p : r[q] = p;
        else
            root = p;
        key[p] = x;
        f[p] = q;
        s[p] = num[p] = 1;
    }
    else
    {
        s[p]++;
        if (x == key[p])
```

```c
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

```cpp
#include <bits/stdc++.h>
using namespace std;
int a, b, x, y, t;
int gcd(int a, int b, int &x, int &y)
{
    if (b)
    {
        int t, xt, yt;
        t = gcd(b, a % b, xt, yt);
        x = yt;
        y = xt − a / b * yt;
        return t;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
int main()
{
    scanf("%d%d", &a, &b);
    t = gcd(a, b, x, y);
    printf("%d␣%d␣%d\n", x, y, t);
    return 0;
}
```

5.4 Euler 筛

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, num, p[N], fai[N], miu[N];
bool flag[N];
int main()
{
    scanf("%d", &n);
    fai[1] = miu[1] = 1;
    for (int i = 2; i <= n; i++)
    {
        if (!flag[i])
        {
            p[++num] = i;
            fai[i] = i − 1;
            miu[i] = −1;
        }
        for (int j = 1; j <= num; j++)
        {
            if (i * p[j] > n)
                break;
            flag[i * p[j]] = true;
            if (i % p[j] == 0)
            {
                fai[i * p[j]] = fai[i] * p[j];
                miu[i * p[j]] = 0;
                break;
            }
            else
            {
                fai[i * p[j]] = fai[i] * (p[j] − 1);
                miu[i * p[j]] = −miu[i];
            }
        }
    }
    printf("%d\n", num);
    for (int i = 1; i < num; i++)
        printf("%d␣", p[i]);
    printf("%d\n", p[num]);
    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

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;
typedef pair<double, double> Point;
const int N = 100001;
int n, top;
Point p[N], s[N];
inline double Sqr(double x)
{
    return x * x;
}
inline double Dist(Point a, Point b)
{
    return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
}
inline double Cross(Point a, Point b, Point c)
{
    return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
}
inline bool cmp(Point a, Point b)
{
    return Cross(p[0], a, b) > 0 || Cross(p[0], a, b) == 0 && Dist(p[0], a) < Dist(p[0], b);
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%lf%lf", &p[i].x, &p[i].y);
        if (p[i].y < p[0].y || p[i].y == p[0].y && p[i].x < p[0].x)
            swap(p[0], p[i]);
    }
    sort(p + 1, p + n, cmp);
    s[top = 1] = p[0];
    for (int i = 1; i < n; i++)
    {
        for (; top > 1 && Cross(s[top - 1], s[top], p[i]) < 0; top--);
        s[++top] = p[i];
    }
    for (; top > 2 && Cross(s[top - 1], s[top], s[1]) < 0; top--);
    printf("%d\n", top);
    for (int i = 1; i <= top; i++)
        printf("%.5f %.5f\n", s[i].x, s[i].y);
    return 0;
}
```

6.4 最小圆覆盖

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;
typedef pair<double, double> Point;
const int N = 1000001;
int x, y, n;
double r;
Point O, p[N];
inline double Sqr(double x)
{
    return x * x;
}
inline double Dist(Point a, Point b)
{
    return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
}
inline Point Calc(Point a, Point b, Point c)
{
    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  }
```