

# Summary

---

暑假开始准备转移博客，试了几个都不怎么满意（我还去试了下LineBlog 不知道那时候在想什么。。）

现在暂时转移至WordPress，不过还在完善中，预计。。算了不瞎预计的好。。

课上说最好做个代码集，嗯嗯 我也觉得挺有必要的  
毕竟现在我连Floyd怎么写都忘了 无脑SPFA\_(:3] <)\_

反正有用没用都稍微写一下，暂定是目录这些，有些还在找例题、整理代码什么的，所以还是空的。

GitHub上还欠了几题，之后会补上来。

我做的二级目录到博客园就被无视了，，将就看看吧  
感觉实在简陋了些啊。。

- STL
  - stack
  - queue
  - priority\_queue
  - sort
  - map
  - set
- 功能函数
  - MAX
  - MIN
  - 最大公约数
- 基础算法与数据结构
  - 快速排序
  - 归并排序
  - 表达式求值（调度场算法）
  - 线段树求区间和
  - AVL树（不包含删除操作）

- k叉哈夫曼树 ( 求合并n个数的最小代价 )
- 并查集 ( 求图的连通性 )
- SPFA求负权环
- SPFA求多源点最短路径 ( 可直接作单源点用 )
- Dijkstra
- Floyd
- 字典树
- 哈希表
- 堆
- 优先队列
- 深搜
- 广搜
- 双向广搜
- 红黑树
- Prim
- Kruskal
- KMP
- ac自动机
- 快速幂
- 其他

题号以作业次数为准

---

## STL

- stack
  - queue
  - priority\_queue
  - sort
  - map
  - set
-

# stack

## 头文件

```
1. #include<stack>
2. using namespace std;
```

## 声明

```
1. stack<数据类型> 变量名;
```

```
1. a.empty() 判断栈是否为空
2. a.pop() 移除栈顶元素
3. a.push(b) 将元素b压入栈中
4. a.size() 返回栈中元素个数
5. a.top() 返回栈顶元素
```

---

# queue

## 头文件

```
1. #include<queue>
2. using namespace std;
```

## 声明

```
1. queue<数据类型> 变量名;
```

```
1. a.empty() 判断队列是否为空
2. a.pop() 将队头元素出队
3. a.push(b) 将元素b入队
4. a.size() 返回队列中元素个数
```

5. `a.front()` 返回队头元素
6. `a.back()` 返回队尾元素

---

## priority\_queue

### 头文件

1. `#include<queue>`
2. `using namespace std;`

### 声明

1. `priority_queue<数据类型> 变量名;`

1. `a.empty()` 判断队列是否为空
2. `a.pop()` 移除队头元素
3. `a.push(b)` 将元素b入队
4. `a.size()` 返回队列中元素个数
5. `a.top()` 返回队头元素
- 6.
7. `//默认从大到小`
8. `//从小到大&&多关键字`
9. `struct t`
10. `{`
11. `int p, q;`
12. `};`
13. `priority_queue<t> a[n];`
14. `bool operator < (t x, t y)`
15. `{`
16. `return x.p < y.p;`
17. `}`

---

## sort

## 头文件

```
1. #include<algorithm>
2. using namespace std;
```

```
1. //从小到大
2. int a[n];
3. sort(a, a+n);
4.
5. //从大到小
6. int compare(int x, int y)
7. {
8.     return x > y;
9. }
10. sort(a, a + 3, compare);
11.
12. //多关键字
13. struct t
14. {
15.     int p, q;
16. };
17. t a[n];
18. int compare(t x, t y)
19. {
20.     if (x.p == y.p) return x.q > y.q;
21.     else return x.p > y.p;
22. }
23. sort(a, a+n, compare);
```

---

## 功能函数

- MAX
  - MIN
  - 最大公约数
-

## MAX

```
1. int max(int x, int y)
2. {
3.     return x > y ? x : y;
4. }
```

---

## MIN

```
1. int min(int x, int y)
2. {
3.     return x < y ? x : y;
4. }
```

---

## 最大公约数

```
1. int gcd(int x, int y)
2. {
3.     if (y == 0) return x;
4.     else return gcd(y, x%y);
5. }
```

---

## 基础算法与数据结构

- 快速排序
- 归并排序
- 表达式求值 (调度场算法)
- 线段树求区间和
- AVL树 (不包含删除操作)
- k叉哈夫曼树 (求合并n个数的最小代价)

- 并查集 ( 求图的连通性 )
  - SPFA求负权环
  - SPFA求多源点最短路径 ( 可直接作单源点用 )
  - Dijkstra
  - Floyd
  - 字典树
  - 哈希表
  - 堆
  - 优先队列
  - 深搜
  - 广搜
  - 双向广搜
  - 红黑树
  - Prim
  - Kruskal
  - KMP
  - ac自动机
  - 快速幂
- 

## 快速排序

```
1.  #include<iostream>
2.  using namespace std;
3.
4.  int i, j, k, n, m, s, t, a[1000];
5.
6.  void q(int l, int r)
7.  {
8.      int i, j, x, t;
9.      i = l;
10.     j = r;
11.     x = a[(i + j) / 2];
12.     do
13.     {
14.         while (a[i] < x) i++;
```

```

15.     while (a[j] > x) j--;
16.     if (i <= j)
17.     {
18.         t = a[i];
19.         a[i] = a[j];
20.         a[j] = t;
21.         i++;
22.         j--;
23.     }
24.     } while (i <= j);
25.     if (j > l) q(l, j);
26.     if (i < r) q(i, r);
27. }
28.
29. int main()
30. {
31.     cin >> n;
32.     for (i = 1; i <= n; i++)
33.         cin >> a[i];
34.     q(1, n);
35.     for (i = 1; i <= n; i++)
36.         cout << a[i] << ' ';
37.     return 0;
38. }

```

## 归并排序

### 2.1 nxd

给定  $n$  个数  $a_1, a_2, \dots, a_n$ ，求满足条件的  $(i, j)$  数量： $i < j$  且  $a[i] < a[j]$

```

1.     #include<iostream>
2.     #include<cstdio>
3.     #include<cstring>
4.     using namespace std;
5.
6.     int a[200000], b[200000];
7.     __int64 s;
8.

```



```

9. void p(int l, int m, int r)
10. {
11.     int i = l;
12.     int j = m + 1;
13.     int k = l;
14.     while (i <= m && j <= r)
15.     {
16.         if (a[i] < a[j])
17.         {
18.             b[k++] = a[j++];
19.             s += m - i + 1;
20.         }
21.         else
22.         {
23.             b[k++] = a[i++];
24.         }
25.     }
26.     while (i <= m) b[k++] = a[i++];
27.     while (j <= r) b[k++] = a[j++];
28.     for (i = l; i <= r; i++)
29.         a[i] = b[i];
30. }
31.
32. void q(int l, int r)
33. {
34.     if (l < r)
35.     {
36.         int m = (l + r) >> 1;
37.         q(l, m);
38.         q(m + 1, r);
39.         p(l, m, r);
40.     }
41. }
42.
43. int main()
44. {
45.     int n;
46.     scanf("%d", &n);
47.     for (int i = 0; i < n; i++)
48.         scanf("%d", &a[i]);
49.     s = 0;
50.     q(0, n - 1);
51.     printf("%I64d", s);
52.     return 0;
53. }

```

---

## 表达式求值 ( 调度场算法 )

### 3.2 calculator

```
1.  #include<stdio.h>
2.  #include<string.h>
3.
4.  int i, j, k, n, m, s, t, a[1000];
5.  char b[2000], c[2000], d[2000];
6.
7.  int main()
8.  {
9.      scanf("%s", &b);
10.     i = 0;
11.     j = 0;
12.     k = 0;
13.     n = strlen(b);
14.     //中缀转后缀
15.     while (i < n)
16.     {
17.         if ((b[i] >= '0') && (b[i] <= '9'))
18.         {
19.             while ((b[i] >= '0') && (b[i] <= '9'))
20.             {
21.                 c[j++] = b[i++];
22.             }
23.             c[j++] = '!';
24.         }
25.         if ((b[i] == '+') || (b[i] == '-'))
26.         {
27.             while ((k > 0) && (d[k - 1] != '('))
28.             {
29.                 c[j++] = d[k - 1];
30.                 k--;
31.             }
32.             d[k++] = b[i];
33.         }
34.         if ((b[i] == '*') || (b[i] == '/'))
35.         {
36.             while ((k > 0) && (d[k - 1] != '(') && ((d[k - 1] == '*') ||
(d[k - 1] == '/'))
```

```

37.         {
38.             c[j++] = d[k - 1];
39.             k--;
40.         }
41.         d[k++] = b[i];
42.     }
43.     if (b[i] == '(')
44.     {
45.         d[k++] = b[i];
46.     }
47.     if (b[i] == ')')
48.     {
49.         while ((k > 0) && (d[k - 1] != '('))
50.         {
51.             c[j++] = d[k - 1];
52.             k--;
53.         }
54.         if (k > 0) k--;
55.     }
56.     i++;
57. }
58. while (k > 0)
59. {
60.     c[j++] = d[k - 1];
61.     k--;
62. }
63. //计算后缀
64. c[j] = '\0';
65. i = 0;
66. j = -1;
67. while (c[i] != '\0')
68. {
69.     if ((c[i] >= '0') && (c[i] <= '9'))
70.     {
71.         double x = 0;
72.         while ((c[i] >= '0') && (c[i] <= '9'))
73.         {
74.             x = 10 * x + c[i] - '0';
75.             i++;
76.         }
77.         j++;
78.         a[j] = x;
79.     }
80.     else
81.     {

```

```
82.         j--;
83.         switch (c[i])
84.         {
85.         case '+':
86.         {
87.             a[j] += a[j + 1];
88.             break;
89.         }
90.         case '-':
91.         {
92.             a[j] -= a[j + 1];
93.             break;
94.         }
95.         case '*':
96.         {
97.             a[j] *= a[j + 1];
98.             break;
99.         }
100.        case '/':
101.        {
102.            a[j] /= a[j + 1];
103.            break;
104.        }
105.        }
106.    }
107.    i++;
108. }
109. printf("%d", a[j]);
110. return 0;
111. }
```

---

## 线段树求区间和

### 5.2 bubble\_sort

```
1.     #include<stdio.h>
2.
3.     int i, j, k, n, m, s, t, a[300001], b[100001], c[100001];
4.
5.     int min(int x, int y)
```

```

6.  {
7.      return x < y ? x : y;
8.  }
9.  int max(int x, int y)
10. {
11.     return x > y ? x : y;
12. }
13. int p(int l, int r)
14. {
15.     int s;
16.     s = 0;
17.     l += m - 1;
18.     r += m + 1;
19.     while ((l^r != 1) && (l != r))
20.     {
21.         if (l & 1 == 0) s += a[l ^ 1];
22.         if (r & 1 == 1) s += a[r ^ 1];
23.         l >>= 1;
24.         r >>= 1;
25.     }
26.     return s;
27. }
28.
29. void q(int k)
30. {
31.     k >>= 1;
32.     while (k > 1)
33.     {
34.         a[k] = a[k << 1] + a[(k << 1) + 1];
35.         k >>= 1;
36.     }
37. }
38.
39. int main()
40. {
41.     scanf("%d", &n);
42.     for (i = 1; i <= n; i++)
43.         scanf("%d", &b[i]);
44.     m = 1;
45.     while (m <= n) m <<= 1;
46.     for (i = m + 1; i <= m + n; i++)
47.         a[i] = 1;
48.     for (i = m - 1; i >= 1; i--)
49.         a[i] = a[i << 1] + a[(i << 1) + 1];
50.     for (i = 1; i <= n; i++)

```

```

51.     {
52.         t = p(1, b[i] - 1) + i;
53.         c[b[i]] = max(b[i], max(t, i)) - min(b[i], min(t, i));
54.         a[m + b[i]] = 0;
55.         q(m + b[i]);
56.     }
57.     printf("%d", c[1]);
58.     for (i = 2; i <= n; i++)
59.         printf(" %d", c[i]);
60.     return 0;
61. }

```

## AVL树 ( 不包含删除操作 )

### 8.1 wbhavl

```

1.     #include<stdio.h>
2.     #include<stdlib.h>
3.
4.     int i, j, k, n, m, s, t, a[100001];
5.
6.     struct node
7.     {
8.         int dep;
9.         int val;
10.        node *p;
11.        node *l;
12.        node *r;
13.    };
14.
15.    node* insert(node *tree, int value);
16.    void updata(node *tree);
17.    int depth(node *tree);
18.    node* aaaavl(node *tree, node *newp);
19.    int papa(node *tree);
20.    node* leftSingle(node *tree);
21.    node* rightSingle(node *tree);
22.    node* leftDouble(node *tree);
23.    node* rightDouble(node *tree);
24.    int haha(node *tree, int pp);

```

```
25.
26. node* insert(node *tree, int value)
27. {
28.     node *newp, *nowp;
29.     newp = new node;
30.     newp->val = value;
31.     newp->p = NULL;
32.     newp->l = NULL;
33.     newp->r = NULL;
34.     if (tree == NULL)
35.     {
36.         newp->dep = 1;
37.         tree = newp;
38.     }
39.     else
40.     {
41.         nowp = tree;
42.         while (1 > 0)
43.         {
44.             if (newp->val <= nowp->val)
45.             {
46.                 if (nowp->l == NULL)
47.                 {
48.                     nowp->l = newp;
49.                     newp->p = nowp;
50.                     break;
51.                 }
52.                 else
53.                 {
54.                     nowp = nowp->l;
55.                     continue;
56.                 }
57.             }
58.             else
59.             {
60.                 if (nowp->r == NULL)
61.                 {
62.                     nowp->r = newp;
63.                     newp->p = nowp;
64.                     break;
65.                 }
66.                 else
67.                 {
68.                     nowp = nowp->r;
69.                     continue;
```

```

70.         }
71.     }
72. }
73.     updata(newp);
74.     tree = aaaavl(tree, newp);
75. }
76. return tree;
77. }
78.
79. void updata(node *tree)
80. {
81.     if (tree == NULL) return;
82.     else
83.     {
84.         int l, r;
85.         l = depth(tree->l);
86.         r = depth(tree->r);
87.         tree->dep = 1 + (l > r ? l : r);
88.     }
89. }
90.
91. int depth(node *tree)
92. {
93.     if (tree == NULL) return 0;
94.     else return tree->dep;
95. }
96.
97. node* aaaavl(node *tree, node *newp)
98. {
99.     int pa;
100.    while (newp != NULL)
101.    {
102.        updata(newp);
103.        pa = papa(newp);
104.        if ((pa < -1) || (pa > 1))
105.        {
106.            if (pa > 1)
107.            {
108.                if (papa(newp->r) > 0)
109.                {
110.                    newp = leftSingle(newp);
111.                }
112.            } else
113.            {
114.                newp = leftDouble(newp);

```



```

115.         }
116.     }
117.     if (pa < -1)
118.     {
119.         if (papa(newp->l) < 0)
120.         {
121.             newp = rightSingle(newp);
122.         }
123.         else
124.         {
125.             newp = rightDouble(newp);
126.         }
127.     }
128.     if (newp->p == NULL) tree = newp;
129.     break;
130. }
131. newp = newp->p;
132. }
133. return tree;
134. }
135.
136. int papa(node *tree)
137. {
138.     if (tree == NULL) return 0;
139.     else return depth(tree->r) - depth(tree->l);
140. }
141.
142. node* leftSingle(node *tree)
143. {
144.     node *newroot, *mature;
145.     mature = tree->p;
146.     newroot = tree->r;
147.     if (newroot->l != NULL)
148.     {
149.         newroot->l->p = tree;
150.     }
151.     tree->r = newroot->l;
152.     updata(tree);
153.     newroot->l = tree;
154.     newroot->p = mature;
155.     if (mature != NULL)
156.     {
157.         if (mature->l == tree)
158.         {
159.             mature->l = newroot;

```

```

160.     }
161.     else
162.     {
163.         mature->r = newroot;
164.     }
165. }
166. tree->p = newroot;
167. updata(newroot);
168. return newroot;
169. }
170.
171. node* rightSingle(node *tree)
172. {
173.     node *newroot, *mature, *naive;
174.     mature = tree->p;
175.     newroot = tree->l;
176.     if (newroot->r != NULL)
177.     {
178.         newroot->r->p = tree;
179.     }
180.     tree->l = newroot->r;
181.     updata(tree);
182.     newroot->r = tree;
183.     newroot->p = mature;
184.     if (mature != NULL)
185.     {
186.         if (mature->l == tree)
187.         {
188.             mature->l = newroot;
189.         }
190.         else
191.         {
192.             mature->r = newroot;
193.         }
194.     }
195.     tree->p = newroot;
196.     updata(newroot);
197.     return newroot;
198. }
199.
200. node* leftDouble(node *tree)
201. {
202.     rightSingle(tree->r);
203.     return leftSingle(tree);
204. }

```

```
205.
206. node* rightDouble(node *tree)
207. {
208.     leftSingle(tree->l);
209.     return rightSingle(tree);
210. }
211.
212. int haha(node *tree, int pp)
213. {
214.     node *nowp;
215.     int qq;
216.     qq = 1;
217.     nowp = tree;
218.     while (nowp)
219.     {
220.         if (nowp->val > pp)
221.         {
222.             nowp = nowp->l;
223.             qq++;
224.         }
225.         else
226.         {
227.             if (nowp->val < pp)
228.             {
229.                 nowp = nowp->r;
230.                 qq++;
231.             }
232.             else break;
233.         }
234.     }
235.     return qq;
236. }
237.
238. int main()
239. {
240.     node *tree, *now;
241.     int val;
242.     tree = NULL;
243.     scanf("%d", &n);
244.     for (i = 0; i < n; i++)
245.     {
246.         scanf("%d", &a[i]);
247.         tree = insert(tree, a[i]);
248.     }
249.     printf("%d", haha(tree, a[0]));
```

```
250.     for (i = 1; i < n; i++)
251.         printf(" %d", haha(tree, a[i]));
252.     return 0;
253. }
```

---

## k叉哈夫曼树 ( 求合并n个数的最小代价 )

也可用堆或优先队列

### 9.1 hbsz

```
1.     #include<stdio.h>
2.     #include<algorithm>
3.     using namespace std;
4.
5.     int i, j, k, n, m, s, t, b[100002];
6.     short int a[100002];
7.
8.     int main()
9.     {
10.        scanf("%d", &n);
11.        for (i = 0; i < n; i++)
12.            scanf("%d", &a[i]);
13.        sort(a, a + n);
14.        t = 0;
15.        i = 0;
16.        j = 0;
17.        s = 0;
18.        while (n - i + t - j > 1)
19.        {
20.            m = 0;
21.            for (k = 0; k < 2; k++)
22.            {
23.                if (i == n)
24.                {
25.                    m += b[j];
26.                    j++;
27.                }
28.                else
29.                    if (j == t)
```

```

30.         {
31.             m += a[i];
32.             i++;
33.         }
34.         else
35.             if (a[i] < b[j])
36.             {
37.                 m += a[i];
38.                 i++;
39.             }
40.             else
41.             {
42.                 m += b[j];
43.                 j++;
44.             }
45.         }
46.         s += m;
47.         b[t] = m;
48.         t++;
49.     }
50.     printf("%d", s);
51.     return 0;
52. }

```

---

## 并查集 (求图的连通性)

### 10.2 friends

```

1.     #include<stdio.h>
2.
3.     struct node
4.     {
5.         int x, y;
6.     };
7.
8.     node e[50010];
9.
10.    int i, j, k, n, m, s, t, x, y, d, l, a[50010], b[50010], f[50010], c[5
11.    0010], p[50010], q[50010];

```

```

12. int aaaa(int x)
13. {
14.     return f[x] == x ? x : f[x] = aaaa(f[x]);
15. }
16.
17. void qqq(int x)
18. {
19.     int i, pp, qq;
20.     pp = aaaa(x);
21.     i = a[x];
22.     while (i != 0)
23.     {
24.         if (p[e[i].y])
25.         {
26.             qq = aaaa(e[i].y);
27.             if (pp != qq)
28.             {
29.                 t--;
30.                 f[qq] = pp;
31.             }
32.         }
33.         i = e[i].x;
34.     }
35. }
36.
37. int main()
38. {
39.     scanf("%d%d", &n, &m);
40.     for (i = 0; i < n; i++)
41.     {
42.         f[i] = i;
43.     }
44.     l = 0;
45.     for (i = 0; i < m; i++)
46.     {
47.         scanf("%d%d", &x, &y);
48.         l++;
49.         e[l].x = a[x];
50.         a[x] = l;
51.         e[l].y = y;
52.         l++;
53.         e[l].x = a[y];
54.         a[y] = l;
55.         e[l].y = x;
56.     }

```

```

57.     scanf("%d", &d);
58.     for (i = 1; i <= d; i++)
59.     {
60.         scanf("%d", &b[i]);
61.         c[b[i]] = 1;
62.     }
63.     t = 0;
64.     for (i = 0; i < n; i++)
65.     {
66.         if (!c[i])
67.         {
68.             t++;
69.             qq(i);
70.             p[i] = 1;
71.         }
72.     }
73.     q[d + 1] = t;
74.     for (i = d; i >= 1; i--)
75.     {
76.         t++;
77.         qq(b[i]);
78.         p[b[i]] = 1;
79.         q[i] = t;
80.     }
81.     for (i = 1; i <= d + 1; i++)
82.     {
83.         printf("%d\n", q[i]);
84.     }
85.     return 0;
86. }

```

---

## SPFA求负权环

### 11.1 CrazyScientist

```

1.     #include<stdio.h>
2.
3.     int i, j, k, n, m, s, t, p, a[2010], b[80010][3], c[2010];
4.     bool d[2010];
5.

```

```

6. void q(int k)
7. {
8.     int i, j;
9.     d[k] = true;
10.    i = a[k];
11.    while (i != 0)
12.    {
13.        j = b[i][0];
14.        if (c[k] + b[i][1] < c[j])
15.        {
16.            c[j] = c[k] + b[i][1];
17.            if ((d[j] == true) || (p == 1))
18.            {
19.                p = 1;
20.                if (d[s] == true)
21.                {
22.                    t = 1;
23.                }
24.                break;
25.            }
26.            q(j);
27.        }
28.        i = b[i][2];
29.    }
30.    d[k] = false;
31. }
32.
33. int main()
34. {
35.     scanf("%d%d", &n, &m);
36.     for (i = 1; i <= n; i++)
37.     {
38.         a[i] = 0;
39.         c[i] = 0;
40.         d[i] = false;
41.     }
42.     s = 0;
43.     for (i = 1; i <= m; i++)
44.     {
45.         scanf("%d%d%d", &j, &k, &t);
46.         s++;
47.         b[s][0] = k;
48.         b[s][1] = t;
49.         b[s][2] = a[j];
50.         a[j] = s;

```



```

51.     }
52.     scanf("%d", &s);
53.     t = 0;
54.     for (i = 1; i <= n; i++)
55.     {
56.         p = 0;
57.         q(i);
58.         if (t == 1) break;
59.     }
60.     if (t == 1)
61.         printf("EL PSY CONGROO");
62.     else
63.         printf("ttt");
64.     return 0;
65. }

```

## SPFA求多源点最短路径（可直接作单源点用）

### 11.2 FuYihao

```

1.     #include<stdio.h>
2.     #include<string.h>
3.
4.     int i, j, k, n, m, s, t, q, a[410][410] = { 0 }, b[410][410] = { 0 }, c
5.     [410], d[200010], e[410][410];
6.     bool f[410];
7.     void sasasa(int k)
8.     {
9.         int i, j, h, t;
10.        if (k > 1)
11.        {
12.            j = 1;
13.            for (i = 2; i < k; i++)
14.                if (e[i][k] < e[j][k]) j = i;
15.            for (i = 1; i <= n; i++)
16.                e[k][i] = e[j][k] + e[j][i];
17.        }
18.        e[k][k] = 0;
19.        f[k] = true;

```

```

20.     d[1] = k;
21.     h = 0;
22.     t = 1;
23.     while (h < t)
24.     {
25.         h++;
26.         j = d[h];
27.         f[j] = false;
28.         for (i = 1; i <= n; i++)
29.         {
30.             if (e[k][i] > e[k][j] + a[j][i])
31.             {
32.                 e[k][i] = e[k][j] + a[j][i];
33.                 if (f[i] == false)
34.                 {
35.                     t++;
36.                     d[t] = i;
37.                     f[i] = true;
38.                 }
39.             }
40.         }
41.     }
42. }
43.
44. int main()
45. {
46.     memset(a, 0x3f, sizeof(a));
47.     memset(e, 0x3f, sizeof(e));
48.     scanf("%d%d", &n, &m);
49.     for (i = 0; i < m; i++)
50.     {
51.         scanf("%d%d%d", &j, &k, &t);
52.         if ((a[j][k] != 0) && (t > a[j][k])) continue;
53.         a[j][k] = t;
54.         a[k][j] = t;
55.     }
56.     scanf("%d", &q);
57.     for (i = 1; i <= n; i++)
58.     {
59.         memset(f, 0, sizeof(f));
60.         sasasa(i);
61.     }
62.     while (q--)
63.     {
64.         scanf("%d%d", &j, &k);

```

```
65.     if (e[j][k] != 0x3f3f3f3f)
66.     {
67.         if (q > 0) printf("%d\n", e[j][k]);
68.         else printf("%d", e[j][k]);
69.     }
70.     else
71.     {
72.         if (q > 0) printf("-1\n");
73.         else printf("-1");
74.     }
75. }
76. return 0;
77. }
```

---

## Dijkstra