

Lesson1

DFS (执着)

数据结构：栈 空间：O(h) 在空间上比BFS有优势

不具有最短性

两个特性

- 回溯
回溯完之后注意恢复现场
- 剪枝

顺序与搜索树

- e1: 全排列问题

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 10;
6
7  int n;
8  int path[N];
9  bool st[N];
10
11 void dfs(int u) {
12     if (u == n) { //退出条件
13         for (int i=0; i<n; i++) printf("%d ", path[i]);
14         puts("");
15         return ;
16     }
17
18     for (int i=1; i<=n; i++) {
19         if (!st[i]) {
20             path[u] = i;
21             st[i] = true;
22             dfs(u+1);
23             st[i] = false; //恢复现场
24         }
25     }
26 }
27
28
29 int main(void) {
30     scanf("%d", &n);
31
32     dfs(0);
33
34     return 0;
35 }
```

- e2: n皇后问题

第一种搜索顺序:

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 20;
6
7  int n;
8  char g[N][N];
9  bool col[N], dg[N], udg[N]; //dg对角线, udg反对角线, dg数量是n的二倍, 所以N=20
10
11 //全排列搜索顺序
12 void dfs(int u) {
13     if (u == n) {
14         for (int i=0; i<n; i++) puts(g[i]);
15         puts("");
16         return ;
17     }
18
19     for (int i=0; i<n; i++) {
20         if (!col[i] && !dg[i+u] && !udg[i-u+n]) {
21             g[u][i] = 'Q';
```

```
22         col[i] = dg[i+u] = udg[i-u+n] = true;
23         dfs(u+1);
24         //恢复现场
25         g[u][i] = '.';
26         col[i] = dg[i+u] = udg[i-u+n] = false;
27     }
28 }
29 }
30
31 int main(void) {
32     scanf("%d", &n);
33
34     for (int i=0; i<n; i++)
35         for (int j=0; j<n; j++)
36             g[i][j] = '.';
37
38     dfs(0);
39
40     return 0;
41 }
```

第二种搜索顺序：

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 20;
6
7  int n;
8  char g[N][N];
9  bool row[N], col[N], dg[N], udg[N];
10
11 void dfs(int x, int y, int s) {
12     if (y == n) y = 0, x++;
13
14     if (x == n) {
15         if (s == n) {
16             for (int i=0; i<n; i++) puts(g[i]);
17             puts("");
18         }
19
20         return ;
21     }
22
23     //不放皇后
24     dfs(x, y+1, s);
25
26     //放皇后
27     if (!row[x] && !col[y] && !dg[x+y] && !udg[y-x+n]) {
28         g[x][y] = 'Q';
29         row[x] = col[y] = dg[x+y] = udg[y-x+n] = true;
30         dfs(x, y+1, s+1);
31         g[x][y] = '.';
32         row[x] = col[y] = dg[x+y] = udg[y-x+n] = false;
33     }
34 }
35
36 int main(void) {
37     scanf("%d", &n);
38
39     for (int i=0; i<n; i++)
40         for (int j=0; j<n; j++)
41             g[i][j] = '.';
42
43     dfs(0, 0, 0);
44
45     return 0;
46 }
```

BFS (稳重, 层层遍历)

数据结构：队列 空间：O(2^h)

当每条边权重相同时，能找到最短路 (DFS不具备)

```

1 queue <- 初始
2 while queue不空 {
3     t <- 对头
4     扩展 t 所有邻点
5 }

```

- e1: 走迷宫

```

1 #include <iostream>
2 #include <cstring>
3
4 using namespace std;
5
6 typedef pair<int, int> PII;
7
8 const int N = 110;
9
10 int n, m;
11 int g[N][N], d[N][N]; //d[i][j]记录起点到点(i, j)的距离
12 PII q[N*N], pre[N][N]; //pre记录路径, 逆序
13 int hh, tt = -1;
14
15 int bfs() {
16     memset(d, -1, sizeof d);
17
18     d[0][0] = 0;
19     q[++tt] = {0, 0};
20
21     int dx[4] = {-1, 0, 1, 0}, dy[4] = {0, 1, 0, -1};
22
23     while (hh <= tt) {
24         auto t = q[hh++];
25
26         for (int i=0; i<4; i++) {
27             int x = t.first+dx[i], y = t.second+dy[i];
28
29             if (x>=0 && x<n && y>=0 && y<m && !g[x][y] && d[x][y] == -1) {
30                 d[x][y] = d[t.first][t.second]+1;
31                 pre[x][y] = t;
32                 q[++tt] = {x, y};
33             }
34         }
35     }
36
37     //输出路径
38     // int x = n-1, y = m-1;
39     // while (x || y) {
40     //     cout << x << " " << y << endl;
41     //     auto t = pre[x][y];
42     //     x = t.first, y = t.second;
43     // }
44
45     return d[n-1][m-1];
46 }
47
48 int main(void) {
49     scanf("%d%d", &n, &m);
50
51     for (int i=0; i<n; i++)
52         for (int j=0; j<m; j++)
53             scanf("%d", &g[i][j]);
54
55     cout << bfs() << endl;
56
57     return 0;
58 }
59

```

- e2: 八数码问题

```

1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 #include <queue>
5 #include <unordered_map>
6
7 using namespace std;
8
9 queue<string> q;

```

```
10 unordered_map<string, int> d;
11
12 int bfs(string state) {
13     d[state] = 0;
14     q.push(state);
15
16     int dx[4] = {-1, 0, 1, 0}, dy[4] = {0, 1, 0, -1};
17
18     while (q.size()) {
19         auto t = q.front(); q.pop();
20
21         if (t == "12345678x") return d[t];
22
23         int dist = d[t];
24         int k = t.find('x');
25         int x = k/3, y = k%3;
26
27         for (int i=0; i<4; i++) {
28             int a = x+dx[i], b = y+dy[i];
29
30             if (a>=0 && a<3 && b>=0 && b<3) {
31                 swap(t[a*3+b], t[k]);
32
33                 if (!d.count(t)) {
34                     d[t] = dist+1;
35                     q.push(t);
36                 }
37
38                 swap(t[a*3+b], t[k]);    //恢复现场
39             }
40         }
41     }
42
43     return -1;
44 }
45
46 int main(void) {
47     char s[2];
48     string state;
49
50     for (int i=0; i<9; i++) {
51         scanf("%s", s);
52         state += *s;
53     }
54
55     cout << bfs(state);
56
57     return 0;
58 }
59
```

树与图的存储

树是一种特殊的图（无环连通图）

图的类型

- 有向图

存储方式

- 邻接矩阵，使用二维数组 $g[a, b]$ （不能保存重边）

空间复杂度 n^2 ，适用于稠密图

- 邻接表（每个结点开一个单链表，与拉链法哈希表一直）

适用于稀疏图

```
1  const int N = 100010, M = N*2;    //N代表结点数，M代表边数
2
3  int h[N], e[M], ne[M], idx;
4
5  void add(int a, int b) {
6      e[idx] = b, ne[idx] = h[a], h[a] = idx++;
7  }
```

- 无向图

对于一条无向边，建两条有向边

树与图的深度优先遍历

- 遍历方式

```
1  const int N = 100010, M = N*2;
2
3  int h[N], e[M], ne[M], idx;
4  bool st[N];
5
6  void add(int a, int b) {
7      e[idx] = b, ne[idx] = h[a], h[a] = idx++;
8  }
9
10 void dfs(int u) {
11     st[u] = true;    //已经被遍历
12
13     for (int i=h[u]; i!=-1; i=ne[i]) {
14         int j = ne[i];
15         if (!st[j]) dfs(j);
16     }
17 }
```

- e1: 树的重心

```
1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  const int N = 100010, M = 2*N;
7
8  int n;
9  int h[N], e[M], ne[M], idx;
10 bool st[N];
11 int ans = N;
12
13 void add(int a, int b) {
14     e[idx] = b, ne[idx] = h[a], h[a] = idx++;
15 }
16
17 //返回以u为根的子树的结点个数（包含u）
18 int dfs(int u) {
19     st[u] = true;
20
21     int sum = 1, res = 0;
22     for (int i=h[u]; i!=-1; i=ne[i]) {
23         int j = e[i];
24         if (!st[j]) {
25             int s = dfs(j);
26             sum += s;
27             res = max(res, s);
28         }
29     }
30
31     res = max(res, n-sum);
32     ans = min(ans, res);
33
34     return sum;
35 }
36
37
38 int main(void) {
39     scanf("%d", &n);
40
41     memset(h, -1, sizeof h);
42
43     for (int i=0; i<n; i++) {
44         int a, b;
45         scanf("%d%d", &a, &b);
46         add(a, b), add(b, a);
47     }
48
49     dfs(1);
50
51     cout << ans;
52 }
```

```
53     return 0;
54 }
```

树与图的宽度优先遍历

- e1: 图中点的层次

```
1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  const int N = 100010, M = N;
7
8  int n, m;
9  int h[N], e[M], ne[M], idx;
10 int q[M], hh, tt = -1;
11 int d[N];    //每个点到1号点的距离
12
13 void add(int a, int b) {
14     e[idx] = b, ne[idx] = h[a], h[a] = idx++;
15 }
16
17 int bfs() {
18     memset(d, -1, sizeof d);
19
20     d[1] = 0;
21     q[++tt] = 1;
22
23     while (hh <= tt) {
24         int t = q[hh++];
25
26         for (int i=h[t]; i!=-1; i=ne[i]) {
27             int j = e[i];
28             if (d[j] == -1) {
29                 d[j] = d[t]+1;
30                 q[++tt] = j;
31             }
32         }
33     }
34
35     return d[n];
36 }
37
38 int main(void) {
39     scanf("%d%d", &n, &m);
40
41     //注意初始化的位置
42     memset(h, -1, sizeof h);
43
44     for (int i=0; i<m; i++) {
45         int a, b;
46         scanf("%d%d", &a, &b);
47         add(a, b);
48     }
49
50     cout << bfs() << endl;
51
52
53     return 0;
54 }
55
```

拓扑排序（有向图宽搜的应用）

有环图不存在拓扑排序。可以证明，有向无环图一定存在拓扑序列

度数

- 入度：有多少边指入，**入度为0**的点可以作为**拓扑序列的起点**
- 出度：有多少边指出

步骤

```

1 queue <- 所有入队为0的点
2 while queue不空 {
3     t <- 队头
4     枚举t的所有出边 t -> j
5         删掉 t -> j, d[j]--;
6         if (d[j] == 0) queue <- j;
7 }

```

- e1: 有向图的拓扑序列

```

1 #include <iostream>
2 #include <cstring>
3
4 using namespace std;
5
6 const int N = 100010, M = 100010;
7
8 int n, m;
9 int h[N], e[M], ne[M], idx;
10 int q[N], hh, tt = -1;
11 int d[N];    //d[i]表示第i个结点的入度
12
13 void add(int a, int b) {
14     e[idx] = b, ne[idx] = h[a], h[a] = idx++;
15 }
16
17 bool toposort() {
18     //将所有入度为0的点入队
19     for (int i=1; i<=n; i++)
20         if (!d[i]) q[++tt] = i;
21
22     while (hh <= tt) {
23         int t = q[hh++];
24
25         for (int i=h[t]; i!=-1; i=ne[i]) {
26             int j = e[i];
27             d[j]--;
28             if (d[j] == 0) q[++tt] = j;
29         }
30     }
31
32     return tt == n-1;
33 }
34
35 int main(void) {
36     scanf("%d%d", &n, &m);
37
38     memset(h, -1, sizeof h);
39
40     for (int i=0; i<m; i++) {
41         int a, b;
42         scanf("%d%d", &a, &b);
43         add(a, b);
44         d[b]++;
45     }
46
47     if (toposort())
48         for (int i=0; i<n; i++) printf("%d ", q[i]);
49     else puts("-1");
50
51     return 0;
52 }

```