

7.19 课堂笔记 20

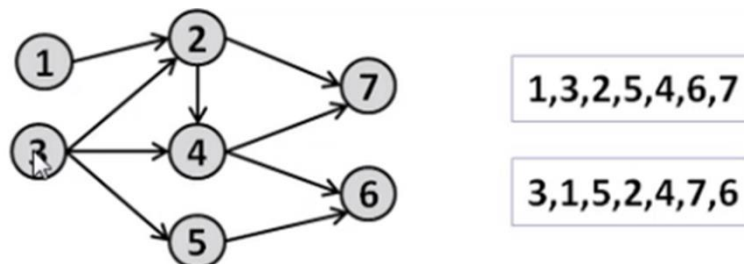
课堂内容

1. 顶点活动网络(AOV 网)

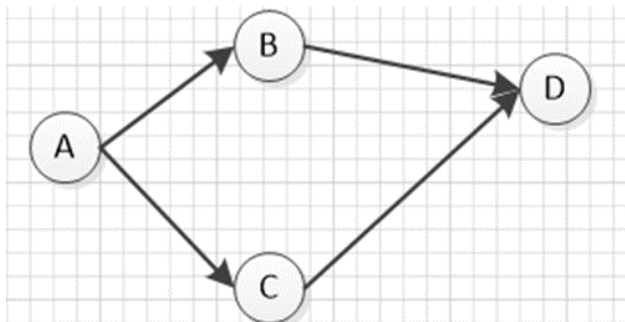
- 一项大的工程可以看作是由若干个子工程组成的集合,这些子工程之间必定存在一定的先后顺序,即某些子工程必须在其他的一些子工程完成后才能开始。
- 在一个表示工程的有向图中,用顶点表示活动,用弧表示活动之间的优先关系。这样的有向图为顶点表示活动的网,我们称为 AOV 网(Activity On Vertex Network)
- 拓扑排序:把 AOV 网中所有活动排成一个序列
- 拓扑排序可以帮助我们合理安排工程进度,由 AOV 网构造拓扑序列具有很高的应用价值

2. 基本概念:

- 活动:子工程组成的集合,每个子工程即为一个活动。
- 前驱活动:有向边起点的活动称为终点的前驱活动(只有当一个活动的前驱全部完成后,这个活动才能进行)。
- 后继活动:有向边终点的活动称为起点后继活动。
- 拓扑排序:将 AOV 网中所有活动排成一个序列,使得每个活动的前驱活动都排在该活动的前面。
- 拓扑序列:经过拓扑排序后得到的活动序列(一个 AOV 网的拓扑序列不是唯一的)。
- 一个 AOV 网一定是一个有向无环图,不存在回路
- 拓扑排序可以用来判断一个有向图是否存在环

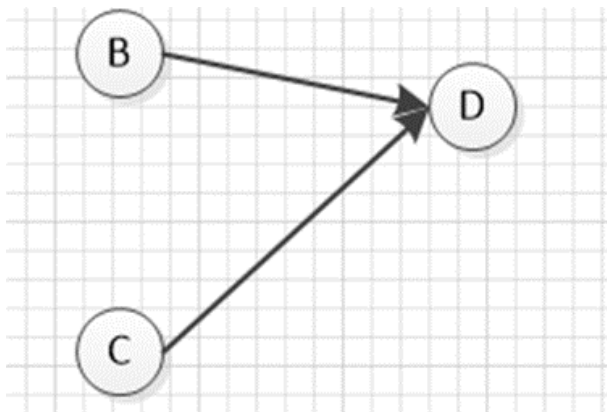


3. 拓扑排序

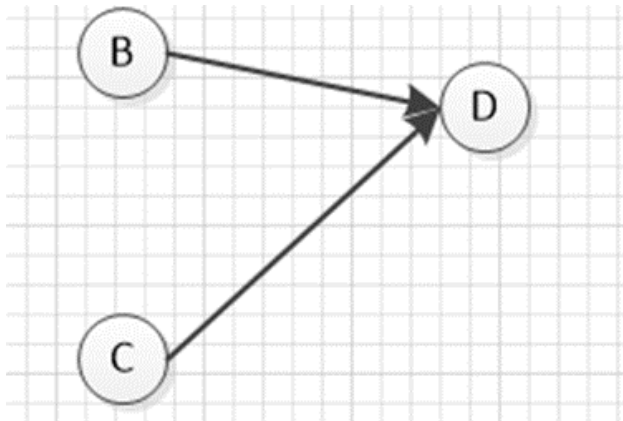


- A 的入度为 0, A 入栈 (队)
- 拓扑序列: 空

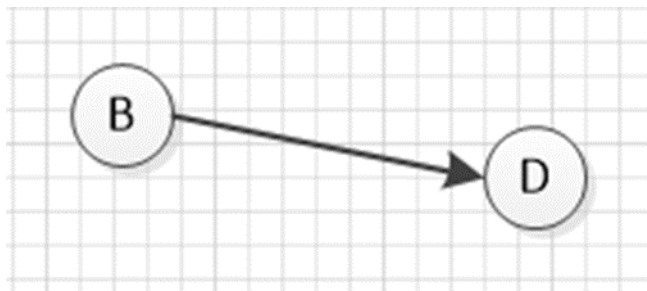
- 栈（队）内元素：A



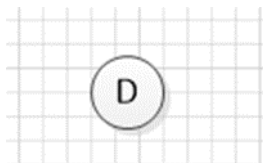
- 栈 A 出栈（队列），输出 A，B、C 入度减 1，等价于删除 A 的关联边
- 拓扑序列：A
- 栈（队列）内元素：空



- B、C 入度为 0，B、C 进栈（队）
- 拓扑序列：A
- 栈（队）内元素：BC（顺序可换）

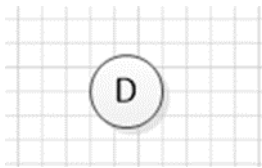


- C 出栈，输出 C，D 入度减 1
- 拓扑序列：AC
- 栈内元素：B



- B 出栈，输出 B，D 入度减 1，D 入栈
- 拓扑序列：ACB

- 栈内元素：D



- D 出栈，输出 D
- 拓扑序列：ACBD
- 栈内元素：空
- 结束

4. 拓扑排序模板

```

1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int numEdge, n, m;
5  int head[110];
6  int rudu[110];
7  queue<int> que;
8  struct edge{
9      int next;
10     int to;
11 } e[10010];
12 void AddEdge(int from, int to){
13     numEdge++;
14     e[numEdge].next = head[from];
15     e[numEdge].to = to;
16     head[from] = numEdge;
17 }
18
19 void TuoPu(){
20     for(int i = 1; i <= n; i++){
21         if(rudu[i] == 0){
22             que.push(i);
23         }
24     }
25     while(!que.empty()){
26         int x = que.front();
27         cout << x << " ";
28         que.pop();
29         for(int i = head[x]; i != 0; i = e[i].next){
30             int y = e[i].to;
31             rudu[y]--;
32             if(rudu[y] == 0){
33                 que.push(y);
34             }
35         }
36     }
37 }
38
39 int main(){
40     cin >> n >> m;
41     int u = 0, v = 0;
42     for(int i = 1; i <= m; i++){
43         cin >> u >> v;
44         rudu[v]++;
45         AddEdge(u, v);
46     }
47     TuoPu();
48 }

```

总结：

- 选择一个入度为 0 的顶点并输出
- 从 AOV 网中删除此顶点和以此顶点为起点的所有关联边
- 重复上述两步，直到不存在入度为 0 的顶点为止
- 若输出的顶点数小于 AOV 网中的顶点数，则说明图中有回路，否则输出序列为一种拓扑排序

5. Activity On Edge Network(AOE 网)

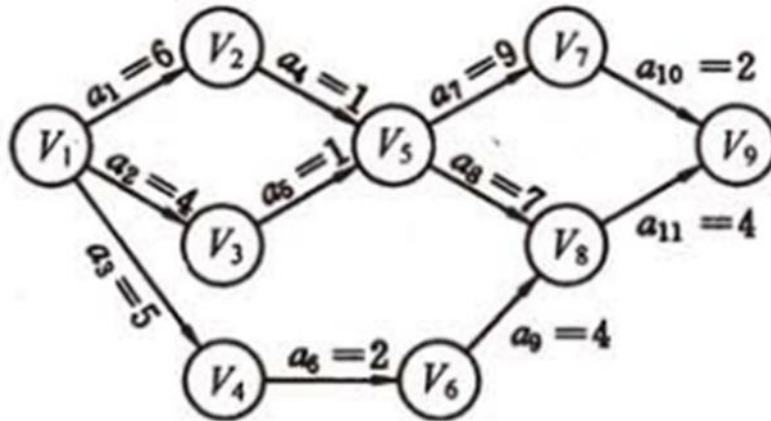
在带权有向图中若以顶点表示事件，有向边表示活动，边上的权值表示该活动持续的时间，这样的图简称为 AOE 网。

性质：

- (1) 只有在某顶点所代表的事件发生后，从该顶点出发的各有向边所代表的活动才能开始。
- (2) 只有在进入某点的各有向边所代表的活动都已结束，该顶点所代表的事件才能

发生。

- 关键路径：在 AOE 网络中从源点到汇点（结束顶点）的最长路径。关键路径上的活动为关键活动。
- 路径长度：路径上各活动持续时间的总和（即路径上所有权之和）。
- 完成工程的最短时间：从工程开始点（源点）到完成点（汇点）的最长路径称为完成工程的最短时间。



关键活动

- 事件最早发生时间 $_{ve}[k]$ ：从起始点到 k 点的最大路径长度，顶点 k 的所有前驱活动均已完成的时

$$_{ve}[1]=0, \quad _{ve}[k]=\max(_{ve}[i] + w[i][k])$$

- 事件最迟发生时间 $_{vl}[k]$ ：事件 k 允许的最晚发生时间，按拓扑逆序求各顶点 $_{vl}$ 值即可

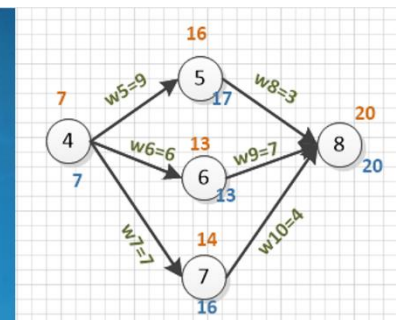
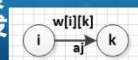
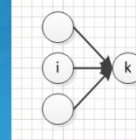
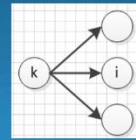
$$_{vl}[n]=_{ve}[n], \quad _{vl}[k]=\min(_{vl}[i] - w[k][i])$$

- 活动最早发生时间 $_{ae}[k]$ ：若活动 a_j 由边 $\langle i, k \rangle$ 表示，则 a_j 的最早开始时间等于事件 i 的最早发生时间。

$$_{ae}[j] = _{ve}[i]$$

- 活动最迟发生时间 $_{al}[k]$ ：若活动 a_j 由边 $\langle i, k \rangle$ 表示，则 a_j 的最晚开始时间要保证事件 k 的最迟发生时间不拖后。

$$_{al}[j] = _{vl}[k] - w[i][k]$$



橙色：事件最早发生时间 $_{ve}[k]$
绿色：事件最迟发生时间 $_{vl}[k]$

有个人的后代很多，辈分关系很混乱，请你帮整理一下这种关系。
给出每个人的孩子的信息。
输出一个序列，使得每个人的后辈都比那个人后列出。

【输入格式】

第1行一个整数N ($1 \leq N \leq 100$)，表示家族的人数。
接下来N行，第i行描述第i个人的儿子。
每行最后是0表示描述完毕。

【输出格式】

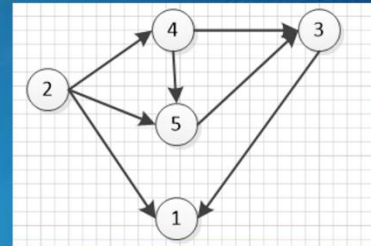
输出一个序列，使得每个人的后辈都比那个人后列出。
如果有多解输出任意一解。

【输入样例】

```
5
0
4 5 1 0
1 0
5 3 0
3 0
```

【输出样例】

```
2 4 5 3 1
```



```
int main(){
    cin >> n;
    for(int i=1; i<=n; i++){
        while(cin >> v2){
            if(v2==0){
                break;
            }
            addEdge(i,v2);
            rudu[v2]++;
        }
    }
    Tuopu();
    return 0;
}
```

7. 洛谷 P4017

你知道食物链吗？Delia 生物考试的时候，数食物链条数的题目全都错了，因为她总是重复数了几条或漏掉了几条。于是她来就来求助你，然而你也不会啊！写一个程序来帮她吧。

题目描述

给你一个食物网，你要求出这个食物网中最大食物链的数量。

（这里的“最大食物链”，指的是生物学意义上的食物链，即最左端是不会捕食其他生物的生产者，最右端是不会被其他生物捕食的消费者。）

Delia 非常急，所以你只有 1 秒的时间。

由于这个结果可能过大，你只需要输出总数模上 80112002 的结果。

输入格式

第一行，两个正整数 n、m，表示生物种类 n 和吃与被吃的关系数 m。

接下来 m 行，每行两个正整数，表示被吃的生物 A 和吃 A 的生物 B。

输出格式

一行一个整数，为最大食物链数量模上 80112002 的结果。

输入 #1

5 7

1 2

1 3

2 3

3 5

2 5

4 5

3 4

输出 #1

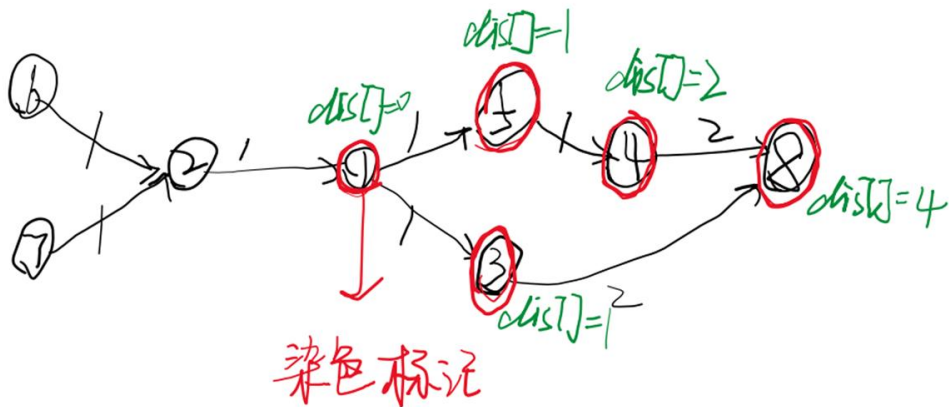
5

n<=5000,m<=500000

保证数据中无环

```
1 #include <iostream>
2 #include <queue>
3 using namespace std;
4 int numEdge, n, m, ans;
5 int head[5010];
6 int rudu[5010];
7 int chudu[5010];
8 int sum[5010];
9 queue<int> que;
10 struct edge{
11     int next;
12     int to;
13 } e[500010];
14
15 void AddEdge(int from, int to){
16     numEdge++;
17     e[numEdge].next = head[from];
18     e[numEdge].to = to;
19     head[from] = numEdge;
20 }
```

```
21 int main(){
22     cin >> n >> m;
23     int u = 0, v = 0;
24     for(int i = 1; i <= m; i++){
25         cin >> u >> v;
26         chudu[u]++;
27         rudu[v]++;
28         AddEdge(u, v);
29     }
30     for(int i = 1; i <= n; i++){
31         if(rudu[i] == 0){
32             que.push(i);
33             sum[i] = 1;
34         }
35     }
36     int x = 0;
37     while(!que.empty()){
38         x = que.front();
39         que.pop();
40         for(int i = head[x]; i != 0; i = e[i].next){
41             int y = e[i].to;
42             sum[y] += sum[x];
43             sum[y] %= 80112002;
44             rudu[y]--;
45             if(rudu[y] == 0){
46                 if(chudu[y] == 0){
47                     ans += sum[y];
48                     ans %= 80112002;
49                 }
50                 que.push(y);
51             }
52         }
53     }
54     cout << ans;
55     return 0;
56 }
```



```

4   int numEdge, n, m;
5   int head[1510];
6   int rdu[1510];
7   int dis[1510];
8   bool flag[1510];
9   int ans;
10  struct edge{
11      int next;
12      int to;
13      int w;
14  } e[100010];
15  void AddEdge(int from, int to, int w){
16      numEdge++;
17      e[numEdge].next = head[from];
18      e[numEdge].to = to;
19      e[numEdge].w = w;
20      head[from] = numEdge;
21  }
22
23  void TuoPu(){
24      flag[1] = true; // 染色初始化
25      queue<int> que;
26      for(int i = 1; i <= n; i++){
27          if(rdu[i] == 0){
28              que.push(i);
29          }
30      }
31      // 1 4 2 6 5 3 7
32      while(!que.empty()){
33          int x = que.front();
34          que.pop();
35          for(int i = head[x]; i != 0; i = e[i].next){
36              int y = e[i].to;
37              int w = e[i].w;
38              if(flag[x]){
39                  flag[y] = true;
40                  dis[y] = max(dis[y], dis[x] + w);
41              }
42              rdu[y]--;
43              if(rdu[y] == 0){
44                  que.push(y);
45              }
46          }
47      }
48  }

```

```

1   int main(){
2       cin >> n >> m;
3       int u = 0, v = 0, w = 0;
4       for(int i = 1; i <= m; i++){
5           cin >> u >> v >> w;
6           AddEdge(u, v, w);
7           rdu[v]++;
8       }
9
10      TuoPu();
11      if(flag[n]){
12          cout << dis[n];
13      }
14      else
15          cout << -1;
16
17      return 0;
18  }

```

9. 洛谷 P1113

```

3  using namespace std;
4  int numEdge, n, m;
5  int head[10010];
6  int rdu[10010];
7  int w[10010];
8  int dp[10010];
9  int ans;
10
11 struct edge{
12     int next;
13     int to;
14 } e[100010];
15 void AddEdge(int from, int to){
16     numEdge++;
17     e[numEdge].next = head[from];
18     e[numEdge].to = to;
19     head[from] = numEdge;
20 }
21 void TuoPu(){
22     queue<int> que;
23     for(int i = 1; i <= n; i++){
24         if(rdu[i] == 0){
25             que.push(i);
26             dp[i] = w[i];
27         }
28     }
29     //1 4 2 6 5 3 7
30     while(!que.empty()){
31         int x = que.front();
32         que.pop();
33         for(int i = head[x]; i != 0; i = e[i].next){
34             int y = e[i].to;
35             //cout << y << " "<<dp[y] <<" "<< dp[x] <
36             dp[y] = max(dp[y], dp[x] + w[y]);
37             rdu[y]--;
38             if(rdu[y] == 0){
39                 que.push(y);
40             }
41         }
42     }
43 }
44
1  int main(){
2     cin >> n;
3     int u = 0, v = 0, maxi = 0;
4     for(int i = 1; i <= n; i++){
5         cin >> u >> w[i];
6         while(cin >> v && v){
7             rdu[u]++;
8             AddEdge(v, u);
9         }
10    }
11
12    TuoPu();
13    for(int i = 1; i <= n; i++){
14        ans = max(ans, dp[i]);
15    }
16
17    cout << ans;
18
19 }
20

```

10. 洛谷 P2419

方法 1: Floyd 解法：求解两点间是否可以联通，三层循环后可构建如下的数组

	1	2	3	4	5
1		1			1
2					1
3		1			1
4		1	1		1
5					

可采用下面的核心代码来判断是否能确定排名

```
for(int i = 1; i <= n; i++){
    int cur = 1;
    for(int j = 1; j <= n; j++){
        if(i != j){
            cur = cur && (f[i][j] || f[j][i]);
        }
    }
    ans += cur;
}
```

以 4 为例：

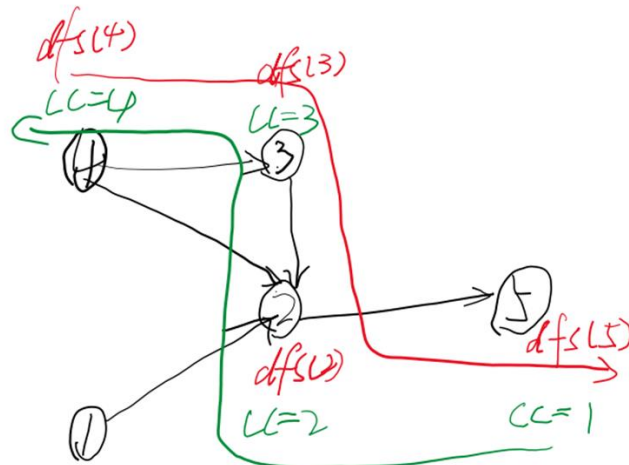
$cur = cur \&\& (f[4][1] || f[1][4])$ 为 0 即无法确定 1 和 4 之间的能力，因此 4 是无法确定的。

```

1 //P2419
2 #include <iostream>
3 using namespace std;
4 int n = 0;
5 int m = 0;
6 bool f[1010][1010];
7 int ans = 0;
8 int x = 0;
9 int y = 0;
10 void floyd(){
11     for(int k = 1; k <= n; k++){
12         for(int i = 1; i <= n; i++){
13             for(int j = 1; j <= n; j++){
14                 f[i][j] = f[i][j] || (f[i][k] && f[k][j]);
15             }
16         }
17     }
18 }
19 int main(){
20     cin >> n >> m;
21     for(int i = 1; i <= m; i++){
22         cin >> x >> y;
23         f[x][y] = true;
24     }
25     floyd();
26     for(int i = 1; i <= n; i++){
27         int cur = 1;
28         for(int j = 1; j <= n; j++){
29             if(i != j){
30                 cur = cur && (f[i][j] || f[j][i]);
31             }
32         }
33         ans += cur;
34     }
35     cout << ans;
36     return 0;
37 }

```

方法二：DFS



以 4 为例：

递归的过程：dfs(4)→dfs(3)→dfs(2)→dfs(1)

返回的过程：4 ← 3 ← 2 ← 1

能得到 4 赢得次数是 4，

判断赢的次数：用正图存储

判断输的次数：用反图存储

判断是否能确定排名的核心：若赢得次数（算自己本身）+输得次数（算自己本身）== N+1
则能确定排名，否则不确定

```

1  #include <iostream>
2  #include <queue>
3  #include <cstring>
4  using namespace std;
5  #define ll long long
6  const int N = 110;
7  const int M = 5000;
8  ll head[N];
9  ll head1[N];
10 ll vis[N];
11 ll vis1[N];
12 ll numEdge;
13 ll numEdge1;
14 ll n, m;
15 struct edge {
16     int to;
17     int next;
18 } e[2*M], e1[2*M];
19 void AddEdge(int from, int to) {
20     numEdge++;
21     e[numEdge].next = head[from];
22     e[numEdge].to = to;
23     head[from] = numEdge;
24 }
25 void AddEdge1(int from, int to) { //反图
26     numEdge1++;
27     e1[numEdge1].next = head1[from];
28     e1[numEdge1].to = to;
29     head1[from] = numEdge1;
30 }
31 int dfs(int x){
32     int cc = 1;
33     vis[x] = true;
34     for(int i = head[x]; i != 0; i = e[i].next){
35         int y = e[i].to;
36         if(!vis[y]){
37             cc += dfs(y);
38         }
39     }
40     return cc;
41 }

```

```

1  int dfs1(int x){
2      int cc = 1;
3      vis1[x] = true;
4      for(int i = head1[x]; i != 0; i = e1[i].next){
5          int y = e1[i].to;
6          if(!vis1[y]){
7              cc += dfs1(y);
8          }
9      }
10     return cc;
11 }
12
13 int main(){
14     int u = 0, v = 0, res = 0;
15     cin >> n >> m;
16     for(int i = 1; i <= m; i++){
17         cin >> u >> v;
18         AddEdge(u, v);
19         AddEdge1(v, u);
20     }
21     for(int i = 1; i <= n; i++){
22         memset(vis, false, sizeof(vis));
23         memset(vis1, false, sizeof(vis1));
24         int cc1 = dfs(i);
25         int cc2 = dfs1(i);
26         if(cc1 + cc2 == n + 1){
27             res++;
28         }
29     }
30     cout << res;
31     return 0;
32 }

```

11. 初赛测试答案:

提高级 CSP-S 第 5 套初赛模拟试题答案及解析

一、单项选择题

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	A	D	B	D	C	B	D	C	B	C	D	B	B	D

【解析】

1. 处理后缀表达式, 需要将两个参与运算的参数前置压入栈中, 然后遇到一个运算符, 就将栈顶的两个元素弹出来运算, 并将得到的结果压入栈中。只有 B 符合题意。
2. 字典序需要从第一个字符开始依次比较大小。
3. 直接计算可以得到结果是 15。
4. 将每两个数分成一组, 先在组内比较一次, 然后组内较小的更新最小值, 组内较大的更新最大值。次数为 $\frac{8}{2} \times 3 - 2 = 10$ 。
5. 在 TCP/IP 协议族中, 最核心的网络协议是 IP 协议。
6. 求第 K 大数, 可以在序列中先随机一个数, 然后将比这个数小的移到这个数的左边, 其余的移到右边, 然后可以判断出第 K 大的数在哪一侧, 递归处理即可。每次规模期望减少一半, 时间复杂度为 $T(1) = O(1), T(n) = T\left(\frac{n}{2}\right) + n = O(n)$ 。
7. 这个数据结构应当满足先进先出原则, 所以选 B。
8. 通过计算可知答案选择 D。
9. 依次用排除法即可, 发现 C 不满足条件。
10. 依次用排除法即可, 发现 B 不满足条件。
11. 本题考查关于操作系统的知识, Windows 使用的不是 Linux 内核。
12. 本题考查计算机算法的基本知识, 只有选项 D 是正确的。
13. 有且仅有描述 B 是正确的。
14. 快速排序是不稳定的。
15. 所有翻转和旋转组合后, 可以得到本题的置换群的置换一共有以下几个: $(1, 2, 3, 4), (2, 3, 4, 1), (3, 4, 1, 2), (4, 1, 2, 3), (1, 4, 3, 2), (2, 1, 4, 3), (3, 2, 1, 4), (4, 3, 2, 1)$, 分解后的循环数分别为 4, 1, 2, 1, 3, 2, 3, 2。根据 Polya 定理, 答案为: $\frac{2 \times 5^1 + 3 \times 5^2 + 2 \times 5^3 + 5^4}{8} = 120$ 。

二、阅读程序

1.

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	√	B	D

【解析】

- (1) 注意递归到 $n=1$ 时, 如果移动的话 $++tot$ 是不会被执行的。
- (2) 模拟可得, $n=2$ 时, 输出的第一行应当是 $A \rightarrow B/A \rightarrow C/B \rightarrow C/$ 。
- (3) 模拟可得, $n=3$ 时, 输出的第二行应当是 7。
- (4) 根据程序可以得知符合题意。
- (5) 输出的第一行是 $A \rightarrow C/A \rightarrow B/C \rightarrow B/A \rightarrow C/B \rightarrow A/B \rightarrow C/A \rightarrow C/$, 其中第 21 个字符是 B。
- (6) 本应得到 $2^n - 1 = 2^{17} - 1 = 131071$, 但是因为 tot 的类型是 `unsigned short` (储存范围是 $0 \sim 2^{16} - 1$), 在发生上溢的时候相当于对 2^{16} 取模, 因此得到 65535。

题号	(1)	(2)	(3)	(4)	(5)
答案	√	×	×	B	D

【解析】

- (1) 全局变量的初值是 0, 因此删去不影响结果。
- (2) 程序会进入死循环。
- (3) 注意本题下标从 0 开始, 因此 $n=4$ 时, 输出的 $a[3][2]=5$ 。
- (4) 显然每个位置只会在值为 0 的时候被访问一次并赋值为非 0 数, 因此时间复杂度为 $O(n^2)$ 。
- (5) 首先程序所求的应当是从 $(0, n-1)$ 开始, 按照逆时针顺序填数。可以知道第 i 圈一共有 $4(n-2i+1)$ 个数。而 $a[33][66]$ 所在的恰好是第 34 圈的第一个数, 1~33 圈共有 $\frac{33 \times (2n-66)}{2} \times 4 = 8844$, 因此 $a[33][66] = 8845$ 。

3.

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	√	B	C

【解析】

- (1) $s[st]$ 应当要赋成最后一个非字母的值。
- (2) 阅读程序可以发现 s 不含大写字母的位置。
- (3) 如果不知道 k , 是不一定能推出输入结果的。如果知道 k , 是一定可以推出输入结果的。
- (4) 可以发现, 当 k 确定时, 一个输出是可以唯一确定一个输入的。
- (5) 手算可得。
- (6) 手算可得。

三、完善程序

1.

题号	(1)	(2)	(3)	(4)	(5)
答案	A	B	C	B	C

【解析】

- (1) 拓扑排序开始应当将入度为 0 的点加入队列中。
- (2) 将当前队首的结点插入到当前拓扑序的末尾。
- (3) 判断 cur 和 i 之间有没有连边。
- (4) i 的入度减一。
- (5) 有一条有向边 (u, v) , 所以要连接 (u, v) 。

2.

题号	(1)	(2)	(3)	(4)	(5)
答案	D	C	B	B	C

【解析】

- (1) 判断皇后是否在一条斜线上。
- (2) 将第 k 行的皇后放在 i 的位置。
- (3) 当 $k=n-1$ 时就停止搜索。
- (4) 继续搜索下一行。
- (5) 从第 0 行开始搜索。