[18] 图的基本应用

深入浅出程序设计竞赛
第 3 部分 – 简单数据结构
V 2021-02

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图


图
图的概念和建立
例 1：凡华中学的地图
顶点、边与权、度数
有向图与无向图
例 2：图的存储
邻接矩阵
邻接表
复杂度分析
图的遍历
例 3：阅读博客
深度优先遍历
广度优先遍历
例 4：图的遍历
图遍历的应用
DAG 与拓扑排序
例 5：杂务
有向无环图
例 6：最大食物链计数
拓扑排序
洛谷

第 18 章 图的基本应用

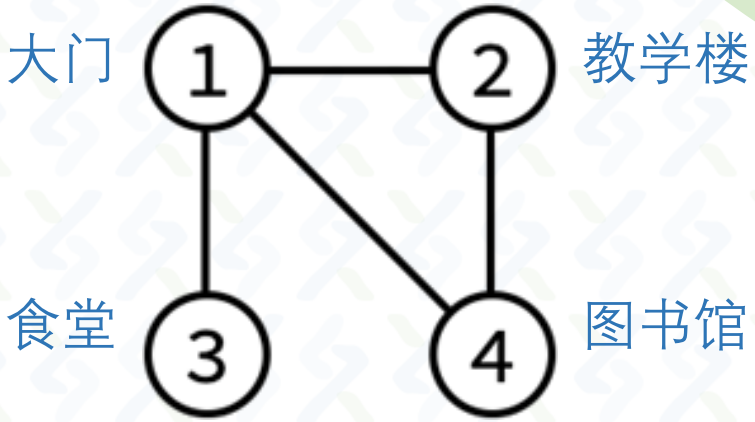
图的概念和建立

学校里有很多建筑，有道路连接这些建筑。由这些建筑和道路组成的集合，就可以认为是图。

请翻至课本 P252

凡华中学的地图

这里是凡华中学的地图，数字代表建筑物，黑色的线是道路。
把每个建筑物称作顶点，每条直接连通的道路称作边。
一个图就是由这些顶点和边构成的集合。
顶点连边的数量就是顶点的度数。



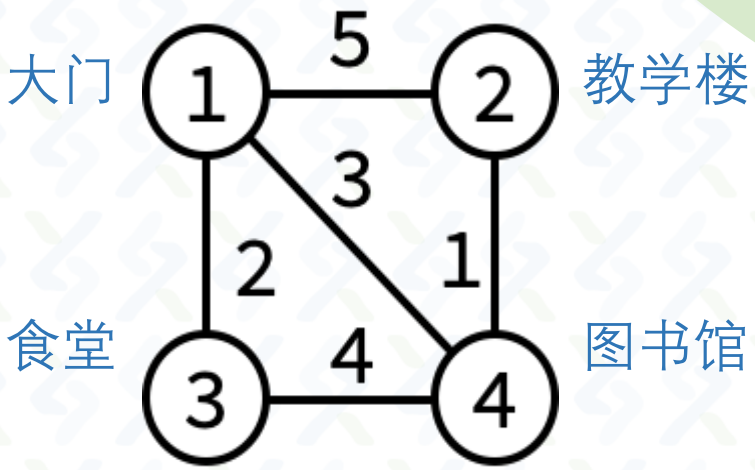
凡华中学的地图

两个顶点中间有不止一条边，这被称为重边。

有时也会有自己到自己的边，被称为自环。

而下图标注了走过这段道路所需时间，这个针对每一条边的属性值被称为边权。

类似地，顶点也可以有点权。



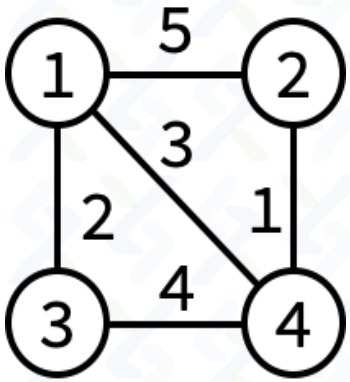
凡华中学的地图

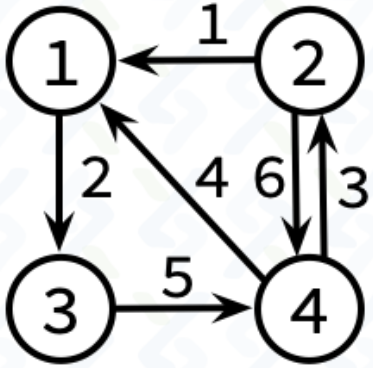
对于右下图，每条道路只能单向通行，即边是单向的，被称为有向图。而左下图则是无向图，可以双向通行。

对于有向图：

- 一个顶点向别的顶点连边的条数称作这个结点的出度，
- 别的顶点连边到一个顶点的条数称作这个结点的入度。

思考：右下图的每个顶点的入度和出度是多少呢？





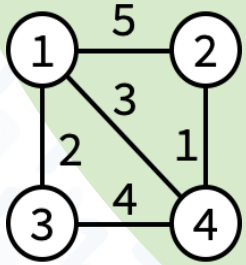
图的存储

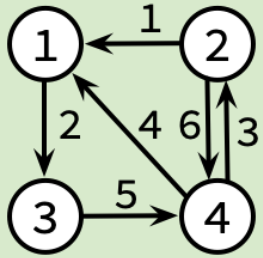
例 18.2

现希望将第二张图和第三张图存进计算机。

可以使用邻接矩阵存储一张图：记 v[i][j] 表示从 i 到 j 的边权。如果不通可以设置为 0 或者 inf（一个很大的数字）。

无向图的邻接矩阵是对称的，有向图的邻接矩阵则不对称。





i\j	1	2	3	4
1	0	5	2	3
2	5	0	0	1
3	2	0	0	4
4	3	1	4	0

i\j	1	2	3	4
1	0	0	2	0
2	1	0	0	6
3	0	0	0	5
4	4	3	0	0

图的存储

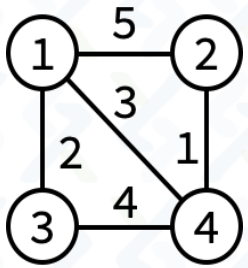
对于一个 n 个点 m 条边的图，在使用邻接矩阵时，需要开一个 $n \times n$ 的数组，即空间复杂度 $O(n^2)$，代码如下：

```
cin >> n;
for (int i = 1; i <= n; i++)
  for (int j = 1; j <= n; j++)
    cin >> v[i][j]; // 存入每一对点之间的边权
for (int i = 1; i <= n; i++)
  for (int j = 1; j <= n; j++)
    if (v[i][j] > 0)
      cout << "edge from point " << i << " to point " << j
           << " with length " << v[i][j] << '\n';
```

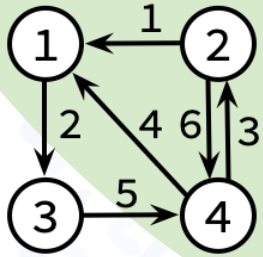
如需要得到图上所有的边，就需要遍历整个 $n \times n$ 数组，即遍历边时间复杂度 $O(n^2)$。如何优化其效率呢？

图的存储

开一个二维数组 p[i][j]，第一维 i 表示起点，第二维 j 表示 i 的第 j 条边，p[i][j] 表示这条边的终点。



p[1][j]	to: 2 cost: 5	to: 3 cost: 2	to: 4 cost: 3
p[2][j]	to: 1 cost: 5	to: 4 cost: 1	
p[3][j]	to: 1 cost: 2	to: 4 cost: 4	
p[4][j]	to: 1 cost: 3	to: 2 cost: 1	to: 3 cost: 4



p[1][j]	to: 3 cost: 2	
p[2][j]	to: 1 cost: 1	to: 4 cost: 6
p[3][j]	to: 4 cost: 5	
p[4][j]	to: 1 cost: 4	to: 2 cost: 3

图的存储

可采用 vector 代替二维数组，使用 vector 存储第二维，从而减少空间占用。这种做法被称为邻接表。

为了同时存储边的终点与边权，可以采用结构体。

```cpp
#include <iostream>
#include <vector>
#define MAXN 1005
using namespace std;

struct edge {
// 记录边的终点，边权的结构体
    int to, cost;
};

int n, m; // 图有 n 个点 m 条边

vector <edge> p[MAXN]; // 邻接表

int v[MAXN][MAXN]; // 邻接矩阵
```

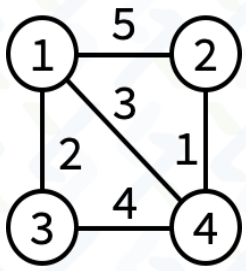
```cpp
int main() {
  cin >> n >> m;
  for (int i = 1; i <= m; i++) {
    int u, v, l;
    cin >> u >> v >> l;
    p[u].push_back((edge) {v, l});
    // p[v].push_back((edge){u, l});
    // 无向图邻接表要加一条反方向的边
  }
}
```

```cpp
// 遍历邻接表，把邻接表转换为邻接矩阵
for (int i = 1; i <= n; i++)
  for (int j = 0; j < p[i].size(); j++)
    v[i][p[i][j].to] = p[i][j].cost;
```

图的存储

邻接表优点：总的空间复杂度是 $O(m)$ 的，且遍历某个点相邻的节点的时间复杂度为 $O(p)$，其中 p 为该点的出度。

邻接表缺点：如果需要指定查询或修改边<i,j>的边权，需要的时间复杂度为 $O(p)$，不如邻接矩阵的 $O(1)$。



i\j	1	2	3	4
1	0	5	2	3
2	5	0	0	1
3	2	0	0	4
4	3	1	4	0

p[1][j]	to: 2 cost: 5	to: 3 cost: 2	to: 4 cost: 3
p[2][j]	to: 1 cost: 5	to: 4 cost: 1	
p[3][j]	to: 1 cost: 2	to: 4 cost: 4	
p[4][j]	to: 1 cost: 3	to: 2 cost: 1	to: 3 cost: 4

邻接矩阵 适用于点较少（几百）、边较多（稠密图）的情况；

邻接表 适用于点较多、或者可能出现重边的情况。

图的遍历

假设你把校园抽象成了一张图，你要从大门开始，参观所有的建筑物处各一次，你会怎么走呢？

请翻至课本 P256

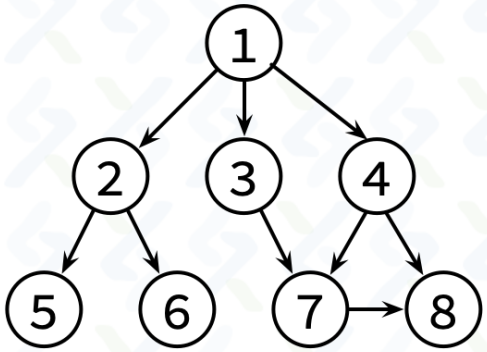
图的遍历

例 18.3 查找文献（洛谷 P5318）

每篇文章有若干个参考文献。若每看一篇文章就会去看它的参考文献（看过就不用看了）。

目前看的是编号 1 的文章。请设计一种看的方法，可以不重不漏地看完所有文章。

如图，其中，文献 X → Y 表示文章 X 有参考文献 Y。

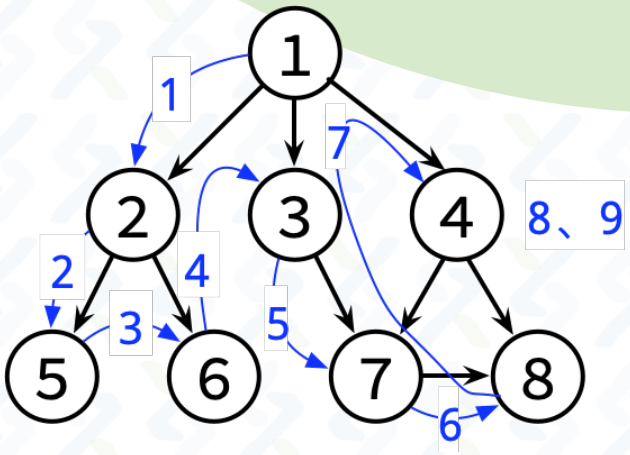


图的遍历

方法1：对于每篇文章，把它和全部参考文献看完再看别的文章。

顺序：1 2 5 6 3 7 8 4

1. 先看博客 1 的参考文献 2；
2. 深入看文献 2 的参考文献 5；
3. 文献 5 没有参考文献，于是继续看文献 2 的参考文献 6；
4. 文献 2 看完了所有参考文献，继续看博客 1 的参考文献 3
5. 深入看文献 3 的参考文献 7；
6. 深入看文献 7 的参考文献 8；
7. 看完了文献 3，继续看文献 1 的参考文献 4；
8. 发现文献 7 已经经被看过了，就不再看了；
9. 文献 8 也读过了。

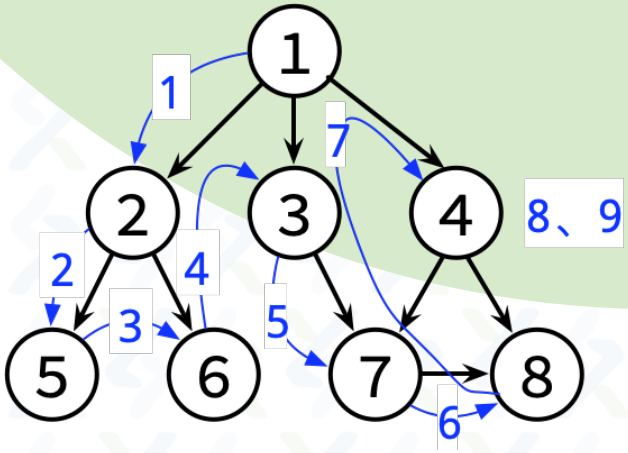


图的遍历

这种方法类似深度优先搜索，被称为深度优先遍历。

特点是优先阅读最先被发现的文章，指导找不到就退回去，需要使用栈维护阅读的层级（一般用递归来实现）。

```
void solve(int x) {
    for (int i = 0; i < p[x].size(); i++)
    if (!u[p[x][i]]) {
        u[p[x][i]] = true;
        solve(p[x][i]);
    }
}
```

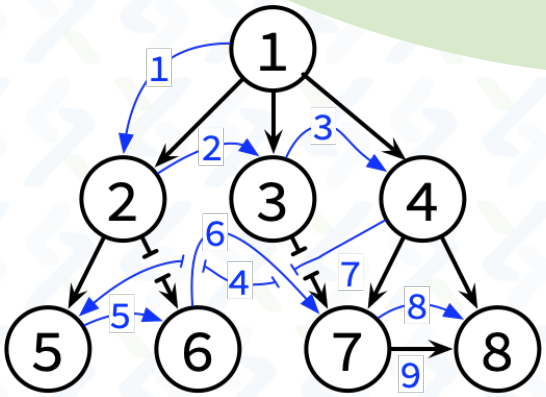


图的遍历

方法 2：把同一引用深度的文章看完，再看下一引用深度的文章。

顺序：1 2 3 4 5 6 7 8

1. 先看博客 1 的参考文献 2；
2. 继续看 1 的参考文献 3；
3. 继续看 1 的参考文献 4；
4. 看完 1 的参考文献，开始看它的参考文献 2 的参考文献 5；
5. 继续看 2 的参考文献 6；
6. 看完 2 的参考文献，开始看 3 的参考文献 7；
7. 发现 4 的参考文献 7 已经被看过了；
8. 继续看 4 的参考文献 8；
9. 发现 7 的参考文献 8 已经被看过了；

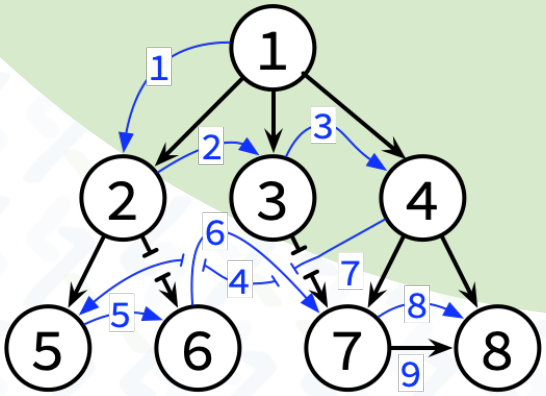


图的遍历

这种方法类似广度优先搜索，被称为深度优先遍历。

特点是优先阅读离初始论文距离更近的文章。需要使用队列来维护等待阅读的文章。

```
while (!q.empty()) {
    int x = q.front();
    q.pop();
    cout << x << ' ';
    for (int i = 0; i < p[x].size(); i++)
        if (!u[p[x][i]]) {
            u[p[x][i]] = true;
            q.push(p[x][i]);
        }
}
```



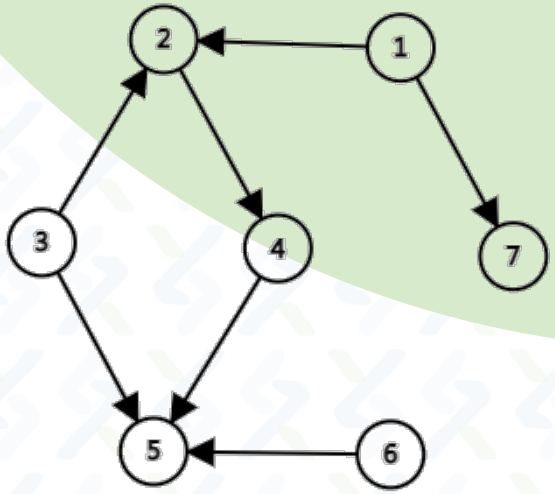
图的遍历

例 18.4 图的遍历（洛谷 P3916）

给出 N 个点，M 条边的有向图，对于每个点 v，求 $A(v)$ 表示从点 v 出发，能到达的编号最大的点。$(N, M \leq 10^5)$

思考：依次处理每个点可以遍历到的点？

别忘记数据范围，可能会超时！

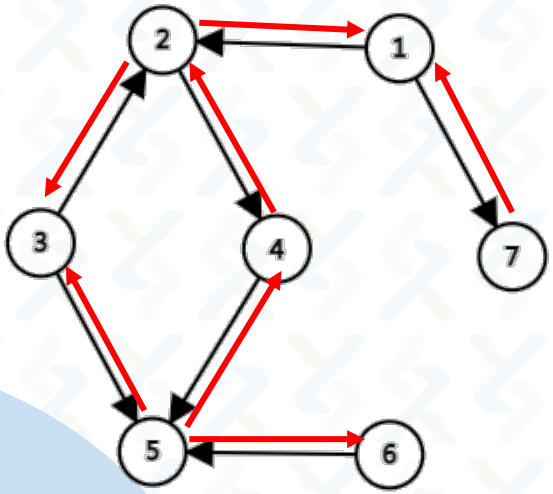


v	1	2	3	4	5	6	7
A(v)	7	5	5	5	5	6	7

图的遍历

思路：让最大的点去“告诉”哪些点能到达它，用反向边建图。

枚举点时从 n 枚举到 1。然后从当前枚举的点 u 出发，让能遍历到的且没有被更新过的点 v 的 A(v) = u。




```
void solve(int x, int v) {
  a[x] = v;
  for (int i = 0; i < p[x].size(); i++)
    if (a[p[x][i]] == 0)
      solve(p[x][i], v);
}
```

```
for (int i = n; i >= 0; i--)
  if (a[i] == 0)
    solve(i, i);
```

洛谷

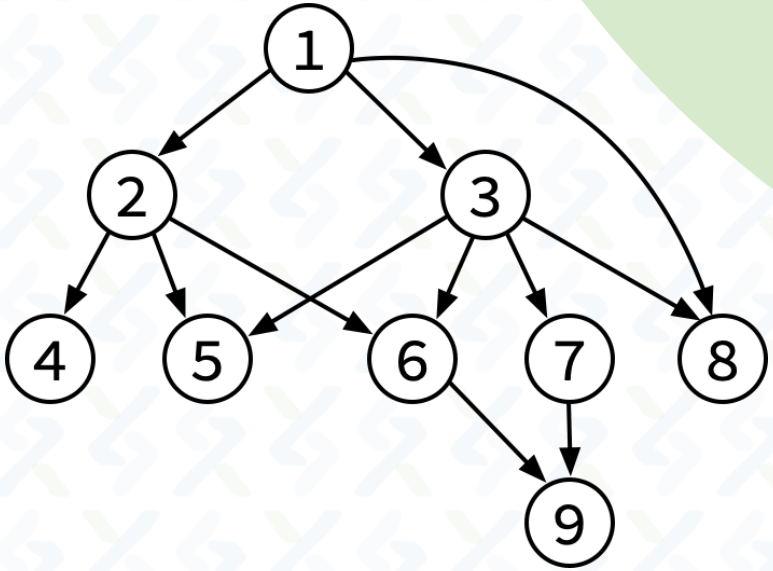
DAG 与拓扑排序

食物链是生物学中的一个重要概念，它表示了物种之间的单向的捕食关系。
由食物链组合能够产生一张食物网，就能直观表现出生态系统的营养基础了。

请翻至课本 P260

有向无环图

对于一张有向图，如果它没有环，则称为有向无环图，简称 DAG。
下面的图就是一个 DAG 的例子。



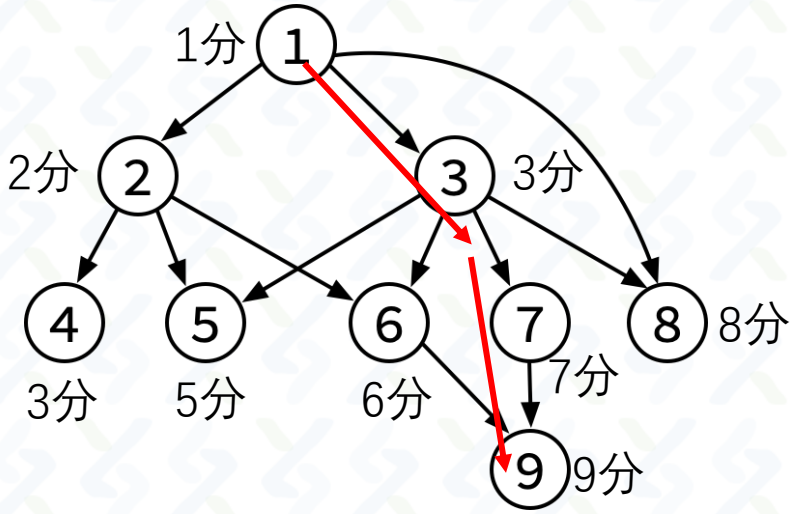
杂务

例 18.5 杂务（洛谷 P1113，USACO2002）

有 n 项杂务要完成，完成第 i 项杂务需要一定的时间 len[i]。

有的杂务需要另一些杂务完成后才能做，互相没有关系的杂务可以同时进行。

请求出完成所有杂务的最短时间。如图 1→3→7→9 是任务路径。



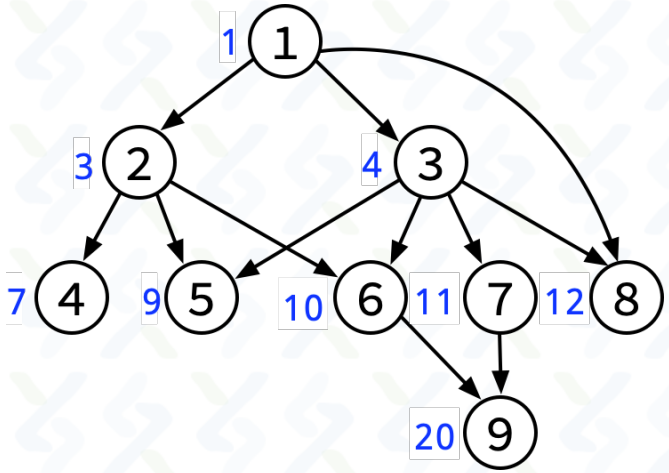
杂务

思路：

如果 x 是 y 的准备工作，那么在 x 和 y 之间连一条有向边。

对于每个任务，使用 vis 数组记下来完成这个任务所需要的最短时间，可以在 dfs 的过程中完成。

完成每个任务所需要的最短时间就是其所有准备任务里面最晚完成的时间加上完成这个任务需要的时间。



杂务

使用记忆化搜索，优化效率完成本题。实际上建图的时候，是后面的任务指向前面的任务，然后去溯源。

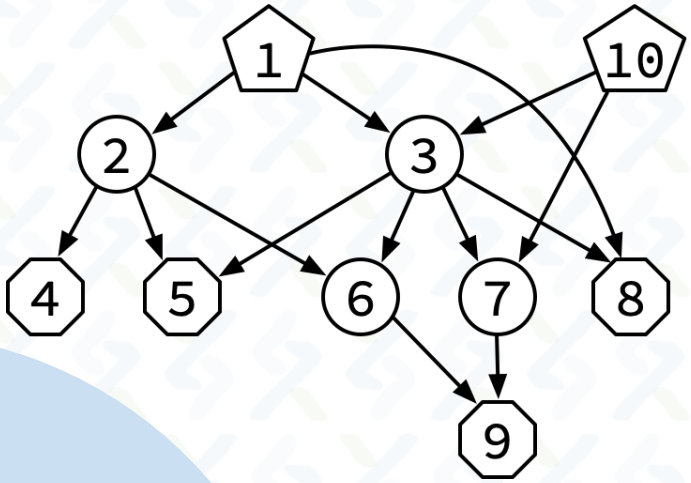
```
int dfs(int x) {
    if (vis[x]) return vis[x];
    for (int i = 0; i < linker[x].size(); i++)
        vis[x] = max(vis[x], dfs(linker[x][i]));
    vis[x] += len[x];
    return vis[x];
}
```

```
cin >> n;
for (int i = 1; i <= n; i++) {
    cin >> x >> len[i];
    while (cin >> y)
        if (!y) break; else linker[y].push_back(x);
}
for (int i = 1; i <= n; i++)
    ans = max(ans, dfs(i));
```

拓扑排序

对于 DAG，在有的时候，对于一个节点的信息的计算，需要确保与之相连的点的信息全部被计算。这时就需要用到拓扑排序。

本质是确保 DAG 上的点的计算顺序而非对数列排序。



STEP 1：将入度为 0 的点插入队列。
STEP 2：取出队头 x，遍历所有x能到达的点y。
STEP 3：对于每一个点 y，维护其节点信息，
同时使它入度 -1 以完成删边操作。
STEP 4：当 y 入度为 0 时，插入队列。
STEP 5：跳转到 STEP 2。

最大食物链计数

例 18.6 最大食物链计数（洛谷 P4017）

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。对 80112002 取模。不超过 5000 个点，500000 条边。

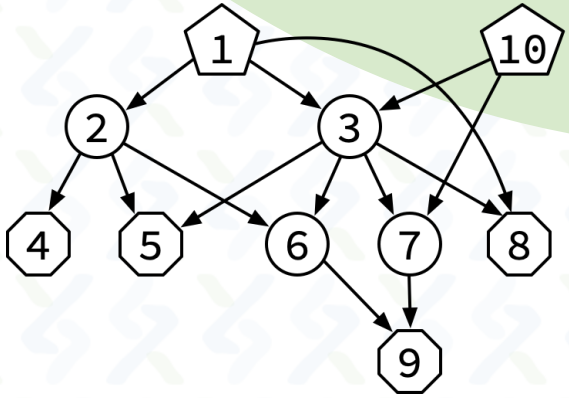
（食物链开头是不会捕食其他生物的生产者，结尾是不会被其他生物捕食的消费者。）

例如：图中食物网就是一个 DAG

五边形点入度为 0，八边形点出度为 0。

图中 1→3→7→9 是一条最大食物链；

而 2→6, 3→7→9 和 1→3→7 都不是。



最大食物链计数

思路：

食物网中的关系是单向且无环的，因而可以抽象为 DAG。

题意自然转化为从入度为 0 的点到出度为 0 的点的链的条数。

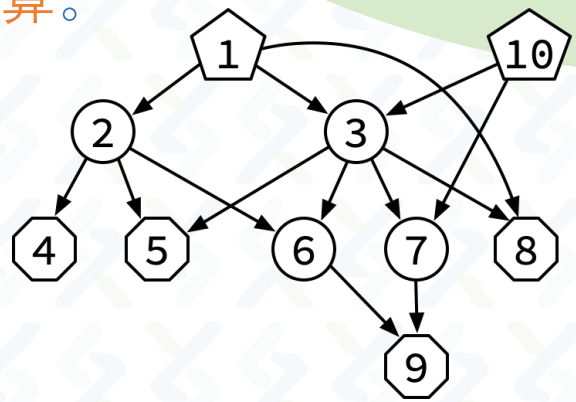
用 f[x] 表示从任意一个入度为 0 的点到点 x 的链的条数之和。则 f[x] 为能到达 x 的所有点 u 的 f[u] 之和。

对于入度为 0 的点，f[x]=1

满足需要确保与之相连的点的信息全部被计算。

使用拓扑排序。只需 STEP 3 中的维护，

变成 f[y] = (f[x] + f[y]) % 80112002 即可。



最大食物链计数

预处理：

```
for (int i=1; i<=m; i++) {
    int x, y;
    cin >> x >> y;
    outd[x]++; // 出度
    ind[y]++;  // 入度
    p[x].push_back(y);
}
memset(f, 0, sizeof(f));
for (int i = 1; i <= n; i++)
    if (ind[i] == 0) {
        q.push(i);
        f[i] = 1;
    }
```

拓扑排序：

```
while (!q.empty()) {
  int x = q.front();
  q.pop();
  for (int i=0; i<p[x].size(); i++) {
    int y = p[x][i];
    f[y] = (f[x] + f[y]) % MOD;
    ind[y]--;
    if (ind[y] == 0)
      q.push(y);
    }
}
```

答案统计：

```
for (int i = 1; i <= n; i++)
    if (outd[i] == 0)
        ans = (ans + f[i]) % MOD;
```

小提示：何处取模

本题，对于 f[x] 的计算，以及最后答案的统计，需要一边运算一边取模，防止运算的中途数据超过了数据类型从而答案错误。

例如：

```
(a + b + c + d) % MOD = a % MOD + b % MOD + c % MOD + d % MOD;
(a * b * c * d) % MOD = a % MOD * b % MOD * c % MOD * d % MOD;
```

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P263

总结

图的定义

由顶点与边组成的集合。

分类：有向图、无向图

存储：邻接矩阵、邻接表

图的遍历

深度优先遍历与广度优先遍历

有向无环图

拓扑排序确保 DAG 上的点的计算顺序

作业

替换练习：

1. 使用广度优先遍历，完成例 18.4。
2. 使用拓扑排序，完成例 18.5。

习题 18.1

请链表来代替 vector 实现邻接表。这种实现方法也被称为“链式前向星”。

作业

习题 18.2 信息传递（洛谷 P2661，NOIP 2015 提高组）

求有向图的最大的环。

习题 18.3 最长路（洛谷 P1807）

设 G 为有 $n(n \leq 1500)$ 个顶点的有向无环图，G 中各顶点的编号为 1 到 n，且当为 G 中的一条边时有 $i < j$。设 $w(i,j)$ 为边的长度，计算图 G 中 <1,n> 间的最长路径。

作业

习题 18.5 奶牛野餐（洛谷 P2853，USACO 2006）

$K(1 \leq K \leq 100)$ 只奶牛分散在 $N(1 \leq N \leq 1000)$ 个牧场。现在她们要集中起来进餐。

牧场之间有$M(1 \leq M \leq 10000)$条有向路连接，而且不存在起点和终点相同的有向路。她们进餐的地点必须是所有奶牛都可到达的地方。那么，有多少这样的牧场呢？

作业

习题 18.6 封锁阳光大学（洛谷 P1330）

阳光大学的校园是一张由 n 个点构成的无向图，n(n≤10000)个点之间由 m(m≤100000) 条道路连接。

每只路障可以对一个点进行封锁，当某个点被封锁后，与这个点相连的道路就被封锁了，无法在与这些道路上走了。

当两只路障封锁了相邻的两个点时，就会产生冲突。

最少需要多少个路障，可以封锁所有道路并且不发生冲突。

作业

习题 18.8 排序（洛谷 P1347）

一个不同的值的升序排序数列指的是一个从左到右元素依次增大的序列，例如，一个有序的数列 A,B,C,D 表示A<B,B<C,C<D。

在这道题中，我们将给你一系列形如 A<B 的关系，并要求你判断是否能够根据这些关系确定这个数列的顺序。

元素数量 n(n≤26)。

参考阅读

以下的内容限于课件篇幅未能详细阐述。如果学有余力，可自行翻阅课本作为扩展学习。

- P263 习题 18.1：如何去使用链表实现邻接表
- 习题 18.4,18.7,18.9。