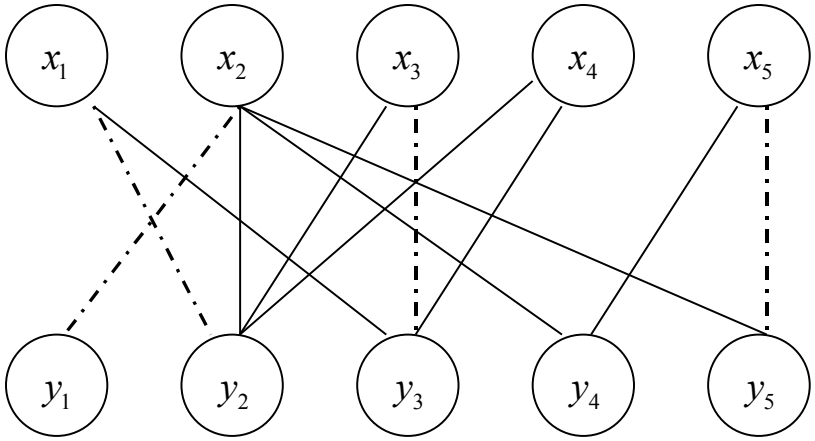
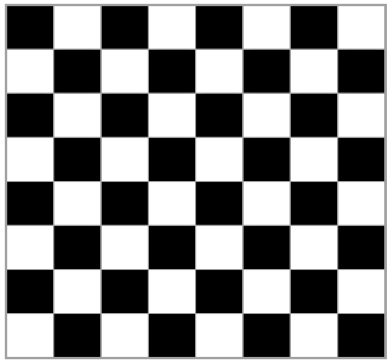
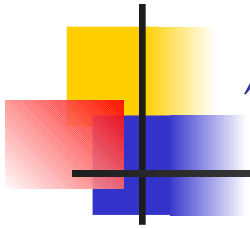
# 二分图匹配及其应用



# 例题 1　棋盘的覆盖



- 一张普通的国际象棋棋盘
- 8*8 = 64　格中被删除了一些格子
- 使用 1*2　的多米诺骨牌进行覆盖
- 求最大覆盖的格数和方案

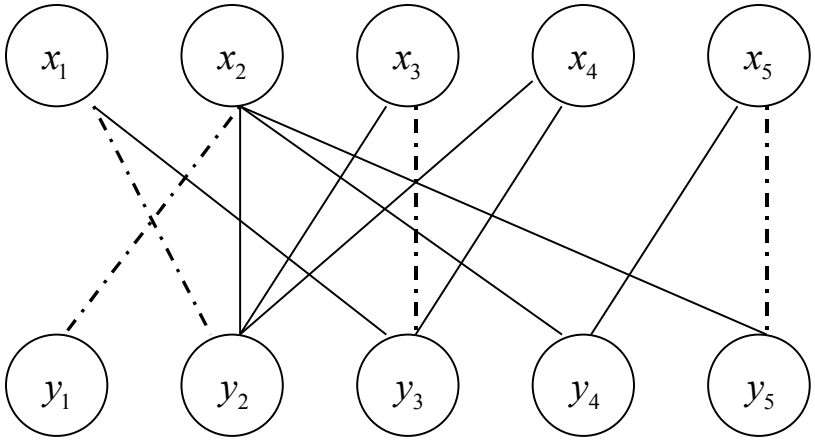


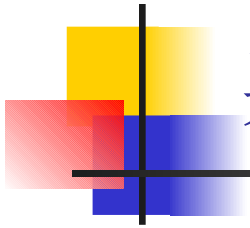
# 例题 1　思路

- 染色
- 建图
- 性质
- 最大匹配和完美匹配

# 基本概念

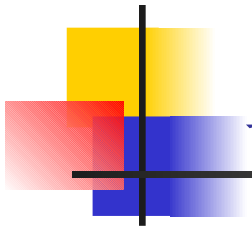
- 点集分为互补的两个集合





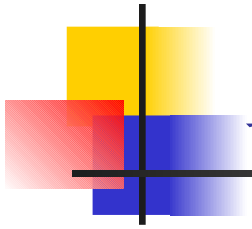
# 基本定理

- 判定定理：
  - 一个图为二分图的充要条件是其所有回路均为偶数长。
  - 如何判断一张图是否为二分图，并对节点进行正确的划分呢？



# 二分图的判断问题

- 染色法
- 将节点用黑白两种颜色染
- 实现：深度优先搜索



# 二分图判断问题解答 1

- var
- visited: array[1..100] of integer;
- data: array[1..100][1..100] of integer;
- n: integer;
- success: boolean;

# 二分图判断问题解答 2

```
procedure dfs(which, color:integer);
var
    i: integer;
begin
    if not success then exit;
    if visited[which] <> 0 then
    begin
        if visited[which] <> color then success = false;
        exit;
    end;
    visited[which] := color;
    for i:=1 to n do
        if data[which][i] <> 0 then dfs(i, 3-color);
end;
```

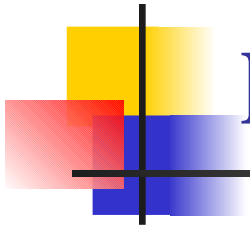
# 二分图判断问题解答 3

```
function judge:boolean;
var
    i: integer;
begin
    success := true;
    for i:=1 to n do
        if visited[i] = 0 then dfs(i,1);
    judge := success;
end.
```

# HALL 定理

- 二分图，存在完美匹配的充分必要条件是，对于任意一个顶点集合的子集 A，它的相邻点构成的邻集 X(A)，都满足以下条件：

$$|X(A)| \geq |A|$$

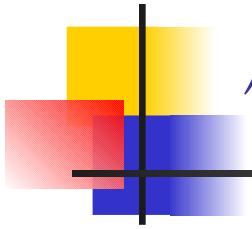


# HALL 定理 证明

- 必要性
- 充分性

# 例题 2 HALL 定理的应用

- 构造 N*N 的矩阵，使得每行都有 1 到 n 每个数字出现一次且仅一次，每列都有 1 到 n 每个数字出现一次且仅一次



# 例题 2　思路

- 每次构造一行
- 循环 n 次构造 n 行
- 建图

- 如何证明？

# 例题 3　思考题

- N 为奇数， M=(N-1)/2 ， 由组合数学知：

$$C_n^m = C_n^{m+1}$$

- 因为 M+M+1 ＝ N

# 例题 3　思考题

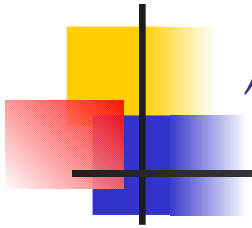
现将所有的从 n 个数里取 m 个数的组合构成一个集合 A

将所有的从 n 个数里取 m+1 个数的组合构成另一个集合 B

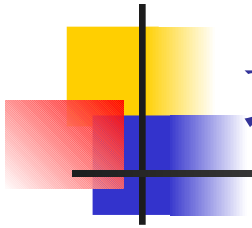
这两个集合是否存在着一一映射的关系？

使得 A 中的每个元素 a 都对应到 B 中的元素 b，且 a 为 b 的一个子集？

# 例题 3　思考题解答



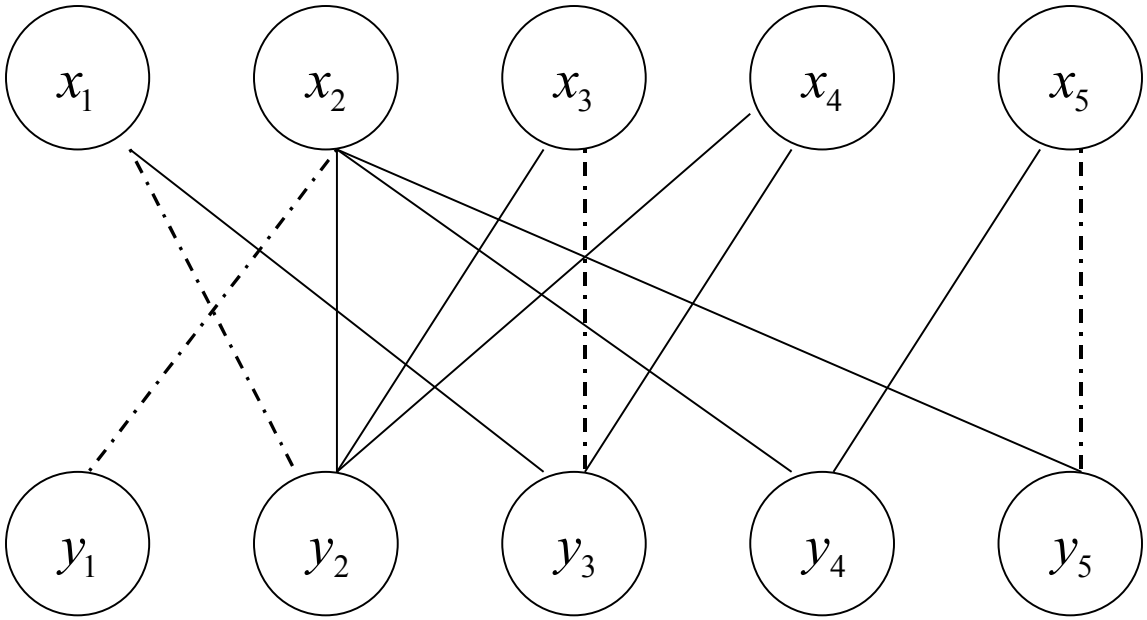
- 建图
- 找完美匹配
- 如何证明？



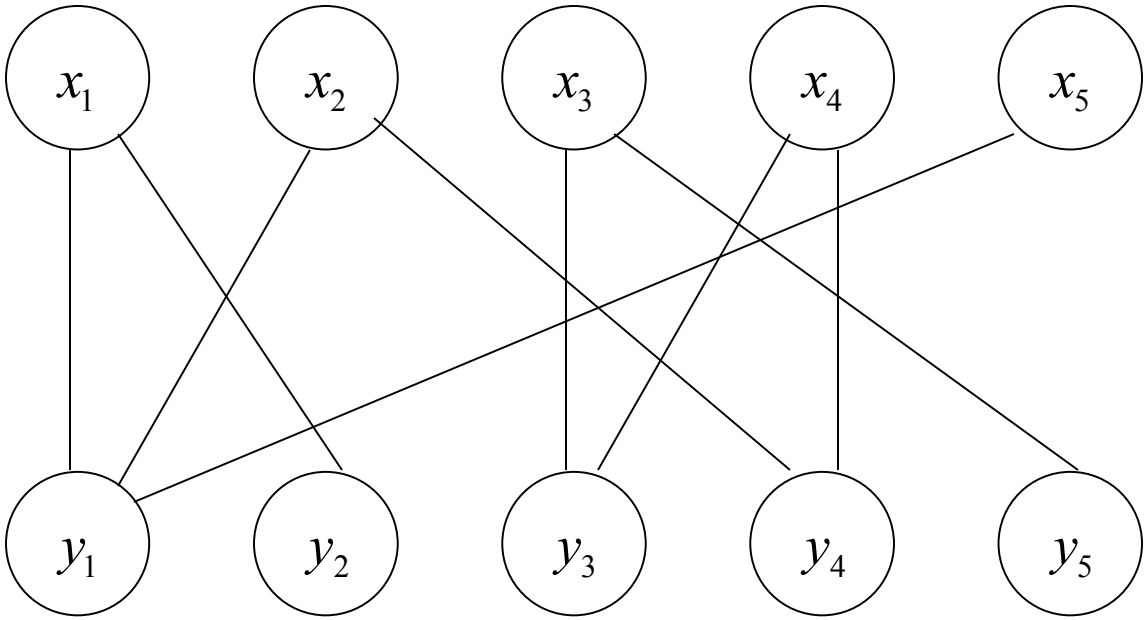
# 增广路和匈牙利算法

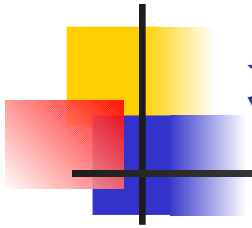
- 增广路（交错轨）的概念
- 匈牙利算法：找增广路
- 如何找？算法选择？

# 增广路和匈牙利算法 实例 1



# 增广路和匈牙利算法 实例 2





# 找增广路的两种办法

- 广度优先搜索
- 深度优先搜索

# 广度优先算法程序

- var
- data: array[1..100,1..100] of integer;
- match1,match2: array[1..100,1..100] of integer;
- n： integer;

# 广度优先算法程序

```
function bmatch:integer;
var
    i,r:integer;
begin
    for i:=1 to n do
        r := r + find(i);
    bmatch := r;
end;
```

# 广度优先算法程序

```
function find(s:integer):integer;
var
    i,j,d,t,qh,ql:integer;
    father2,queue1:array[1..100] of integer;
begin
    fillchar(father2,sizeof(father2),0);
    queue1[1] := s;
    qh := 1;
    ql := 1;
```

# 广度优先算法程序

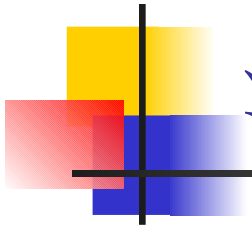
```
while (qh <= q1) do
    begin
        for i:=0 to n do
        begin
            if (father2[i]>0) or
  (data[queue1[qh]][i]=0) then continue;
            if (match2[i]>0) then
            begin
                inc(q1);
                queue1[q1]:=match2[i];
                father2[i]:=queue1[qh];
```

# 广度优先算法程序

```
end else
            begin
                j := queue[qh];
                while (true) do
                begin
                    t:=match1[j];
                    match1[j]:=i;
                    match2[i]:=j;
                    if (t = 0) then break;
                    i:=t;
                    j:=father2[t];
```

# 广度优先算法程序

```
 end;
                    find := 1;
                end;
            end;
        end;
        find := 0;
end;
```



# 深度优先算法程序

```
var
data: array[1..100,1..100] of integer;
match1,match2: array[1..100,1..100] of integer;
int n;

used2: array[1..100] of integer;
```

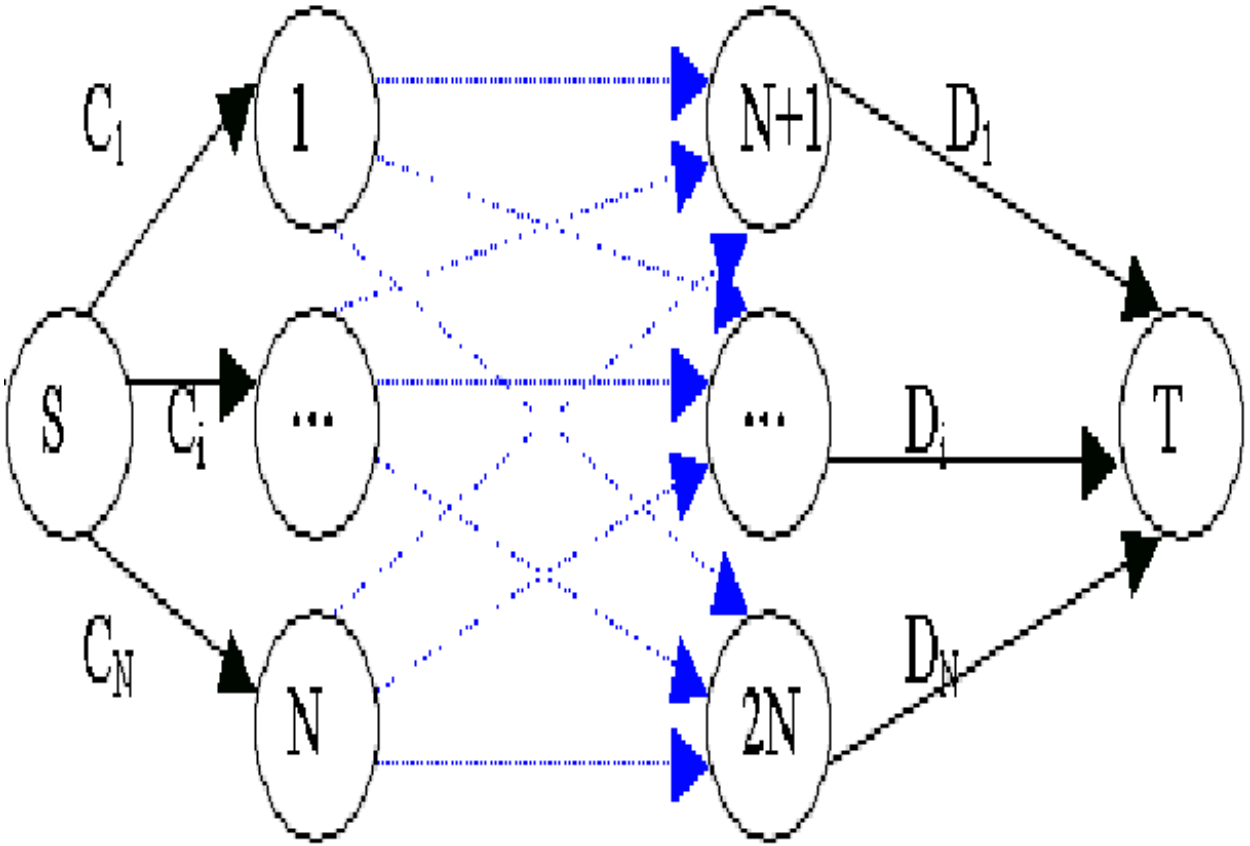
# 深度优先算法程序

```
function bmatch:integer;
var
    i,r:integer;
begin
    for i:=1 to n do
    begin
        fillchar(used2,sizeof(used2),0);
        r := r + find(i);
    end;
    bmatch := r;
end;
```

# 深度优先算法程序

```
function find(s:integer):integer;
var
    i:integer;
begin
    for i:=1 to n do if (data[s][i] <> 0) and (used2[i]=0) then
    begin
        used2[i] := 1;
        if (match2[i] = 0) or find(match2[i]) then
        begin
            match2[i] := s;
            match1[s] := i;
            find := 1;
            exit;
        end;
    end;
    find := 0;
end;
```

# 二分图与网络流的关系



# 例题 4　拦截导弹

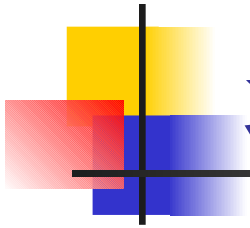
- 某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统 有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高 于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于只有一套系统，因此有可能不能拦截所有的导弹。
- 已知导弹依次飞来的高度，计算 这套系统最多能拦截多少导弹，和如果要拦截所有导弹最少要配备多少套这种导弹拦截 系统。

# 例题 4　拦截导弹　标准解法

- 动态规划
- 贪心法

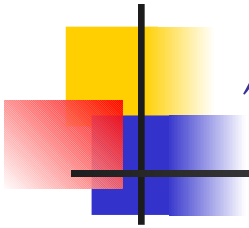
# 例题 4　拦截导弹　匹配解法

- 将特殊情况一般化
- 建图
- 化为最小路径覆盖



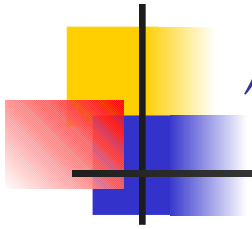
# 最小路径覆盖

- 转换
- 拆分顶点
- 化为二分图匹配



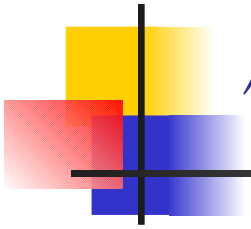
# 例题 5　伞兵降落

- 某个城市的街道图是一个有向无环图，现在伞兵可在图中任意一个节点降落，并负责清扫后面他走过的街道。
- 任意一条街道必须有一个伞兵清扫，并不允许其他伞兵经过
- 求最少需要的伞兵数



# 例题 6　旅馆预订

- 一个旅馆接到 n 张定单，表示要从第 s 天开始入住，从第 e 天结束并离开，
- 求最少需要的房间数目，以满足所有定单的要求。



# 例题 7　火星探测器

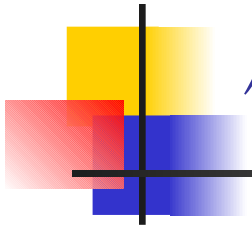
- 火星探测器要在一个 n*n 的网格内采集岩石标本，它从（1,1）出发，到达（n,n）
- 有些网格内是标本，探测器第一次经过时将得到此标本
- 有些网格内是障碍物，不能通过
- 探测器只能走最短路线

# 例题 7　火星探测器 问题

- 问题 1． 只有一个探测器，如何走采集到最多的标本？
- 问题 2． 至少要几个探测器才能收集完这里所有的标本？
- 问题 3． 若只有 k 个探测器，最多能收集到几个标本？

# 例题 8　打猎

- 猎人要在 n*n 的格子里打鸟，他可以在某一行中打一枪，这样此行中的所有鸟都被打掉，也可以在某一列中打，这样此列中的所有鸟都打掉
- 至少打几枪，才能打光所有的鸟？



# 例题 8　打猎　解答

- 建图
- 二分图的最小边覆盖
- 转化
- 二分图的最大匹配