《算法设计与分析》

第六章 回溯法

马丙鹏

2024年11月25日

中国科学院大学
University of Chinese Academy of Sciences

第六章 回溯法

- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题
- 6.6 哈密顿环
- 6.7 和最小
- 6.8 跳马问题

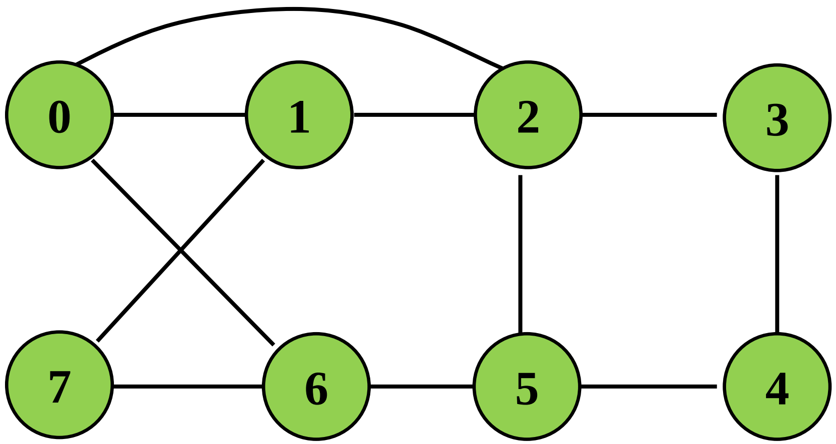
6.6 哈密顿环

- 问题描述
  - 哈密顿环（Hamiltonlan cycle）：连通图G=(V, E)中的一个回路，经过图中每个顶点，且只经过一次。
  - 一个哈密顿环就是从某个结点v_0开始，沿着图G的n条边环行的一条路径$(v_0, v_1, \ldots, v_{n-1}, v_n)$。
    - 除$v_0=v_n$外，路径上其余结点各不相同。
    - $(v_i, v_{i+1}) \in E$ $(0 \le i < n)$。
    - 它访问图中每个结点且仅访问一次，最后返回开始结点。

6.6 哈密顿环

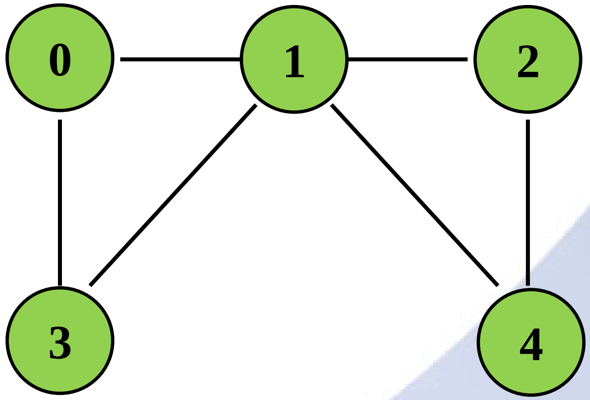
■ 问题描述

□并不是每个连通图都存在哈密顿环!



图G_1包含哈密顿环

(0, 1, 7, 6, 5, 4, 3, 2, 0)



图G_2不包含哈密顿环

要确定一个连通图是否存在哈密顿环没有容易的办法。

6.6 哈密顿环

- 算法说明
  - 采用n-元组($x_1, x_2, \ldots, x_n$)表示哈密顿环问题的解。
    - 显示约束：$x_i \in \{1, 2, \ldots, n-1, n\}, 0<i\leq n$, 代表路径上一个结点的编号 。
    - 隐式约束：$x_i \neq x_j (0<i, j\leq n, i\neq j)$，且$(x_i, x_{i+1}) \in E$ $(i=1, 2, ..., n-2, n-1)$，又$(x_n, x_1) \in E$。
    - 解空间大小为n^n 。

6.6 哈密顿环

- 算法说明
  - 如果已选定$x_1, x_2, \ldots, x_{k-1}$，那么下一步要做的工作是如何找出可能$x_k$的结点集合。
  - 若k=1，则X(1)可以是这n个结点中的任一结点，但为了避免将同一环重复打印n次，可事先指定X(1)=1。
  - 若1<k<n，则X(k)可以是不同于X(1), X(2), …, X(k-1)且和X(k-1)有边相连的任一结点v。
  - X(n)只能是唯一剩下的且必须与X(n-1)和X(1)皆有边相连的结点。
  - 过程NEXTVALUE给出了在求哈密顿环的过程中如何找下一个结点的算法。

算法6.9 生成下一个结点

```
procedure NEXTVALUE(k)
 //X(1),…,X(k-1)是一条有k-1个不同结点的路径。若X(k) =0 ， 则表示再无结点可分配给X(k)。若还有与X(1),…,X(k-1)不同且与X(k-1)有边相连接的结点，则将其中标数最高的结点置于X(k)。若k=n，则还需要与X(1)相连接//
  global integer n, X(1:n); boolean GRAPH(1:n, 1:n); integer k, j
  loop
     X(k) ←( X(k)+1 ) mod ( n+1 )  //下一个结点//
     if X(k)=0 then return endif
     if GRAPH(X(k-1), X(k))   //有边相连吗//
     then for j ← 1 to k-1 do  //检查与前k-1个结点是否相同//
                if X(j)=X(k)   then exit  endif   //有相同结点，退出此循环//
          repeat
          if j=k    //若为真，则是一个不同结点//
              then if k<n or (k=n and GRAPH(X(n), 1)) then return
          endif
     endif
  repeat
end NEXTVALUE
```

算法6.10 找所有的哈密顿环

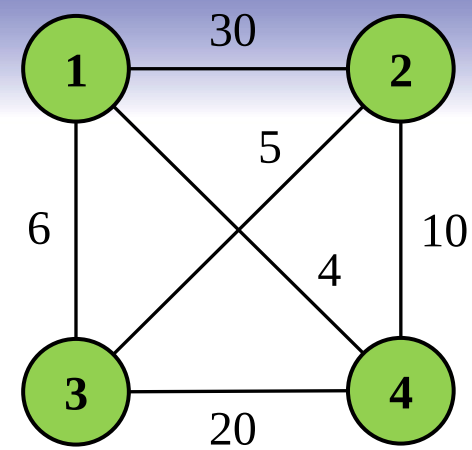
```
procedure HAMILTONIAN(k)
  //这是找出图G中所有哈密顿环的递归算法。图G用它的布尔邻接矩阵GRAPH(1:n,1:n)表示。每个环都从结点1开始//
    global integer X(1:n)
    local integer k, n
    loop  //生成X(k)的值//
        call NEXTVALUE(k)   //下一个合法结点分配给X(k)//
        if X(k)=0 then return endif
        if k=n
        then print(X, '1')   //打印一个环//
        else call HAMILTONIAN(k+1)
        endif
    repeat
end HAMILTONIAN
```

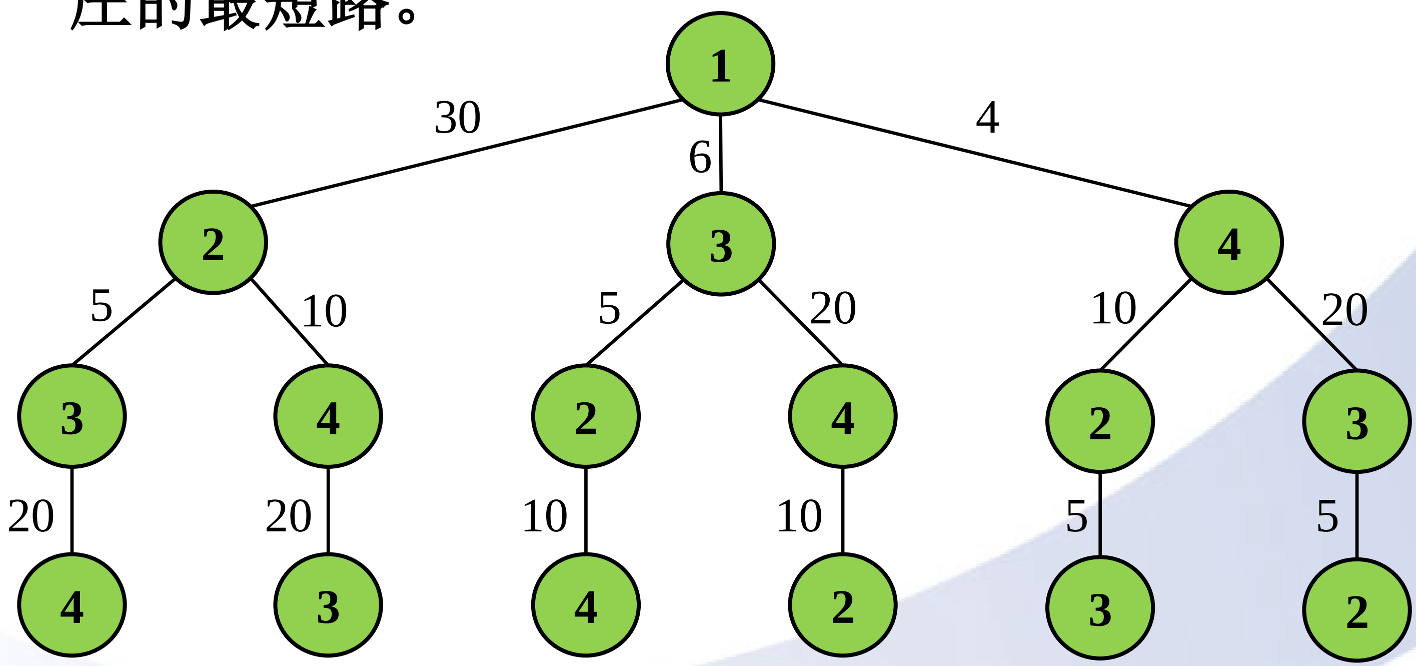
这个过程首先初始化邻接矩阵GRAPH(1:n,1:n)，然后置X(2:n) ← 0，X(1) ← 1，在执行call HAMILTONIAN(2)

6.6 哈密顿环

■ 实例分析

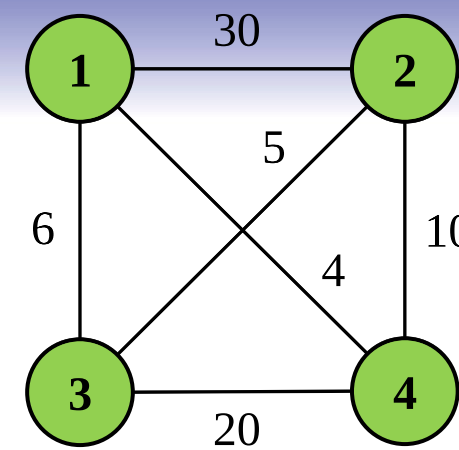
- □有n个村庄，每个村庄必须经过一次，也只能经过一次，求一条走遍全部村庄的最短路。

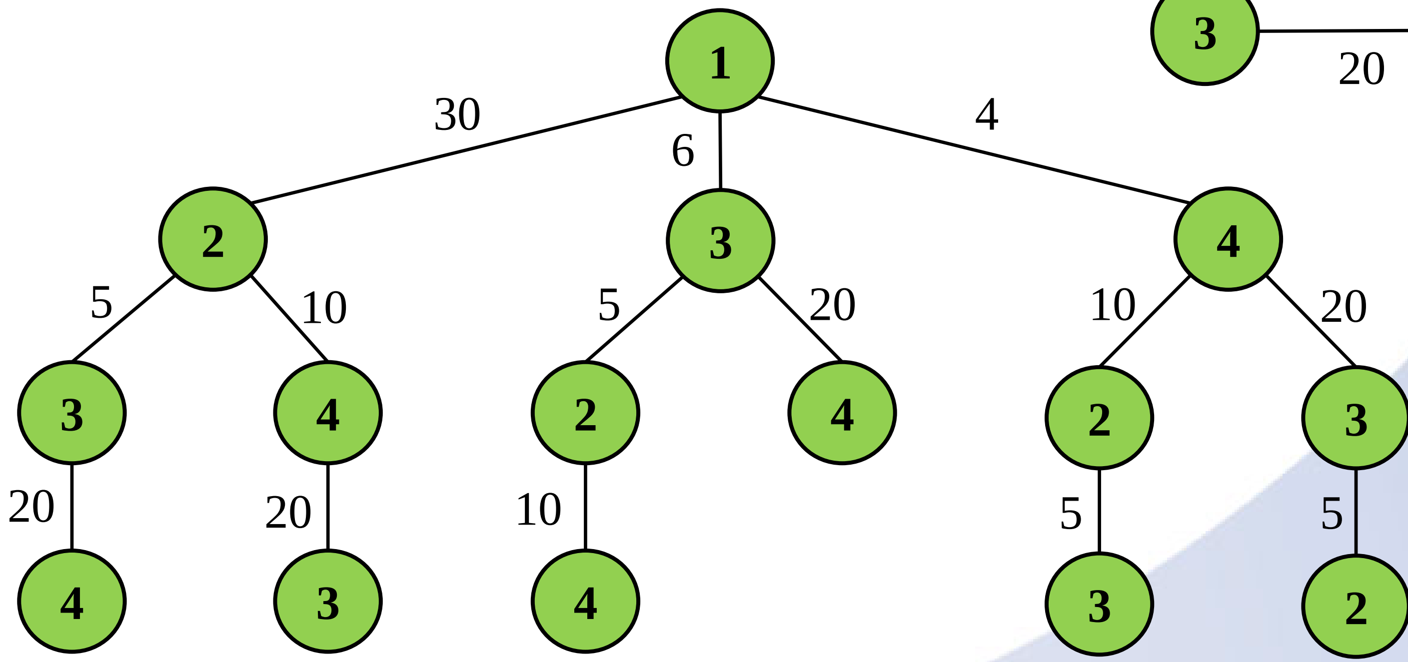




6.6 哈密顿环

■ 实例分析





30+5+20 +4=59

30+10+20 +6=66

6+5+10 +4=25

第六章 回溯法

- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题
- 6.6 哈密顿环
- 6.7 和最小
- 6.8 跳马问题

6.7 和最小

- 问题描述
  - 设有一个长度为N的数字串，要求使用K个加号将它分成K+1个部分，找出一种分法，使得这K+1个部分的和能够为最小。
  - 例：有一个数字串：312，当N=3，K=1时会有以下两种分法：

    ① 3+12=15

    ② 31+2=33

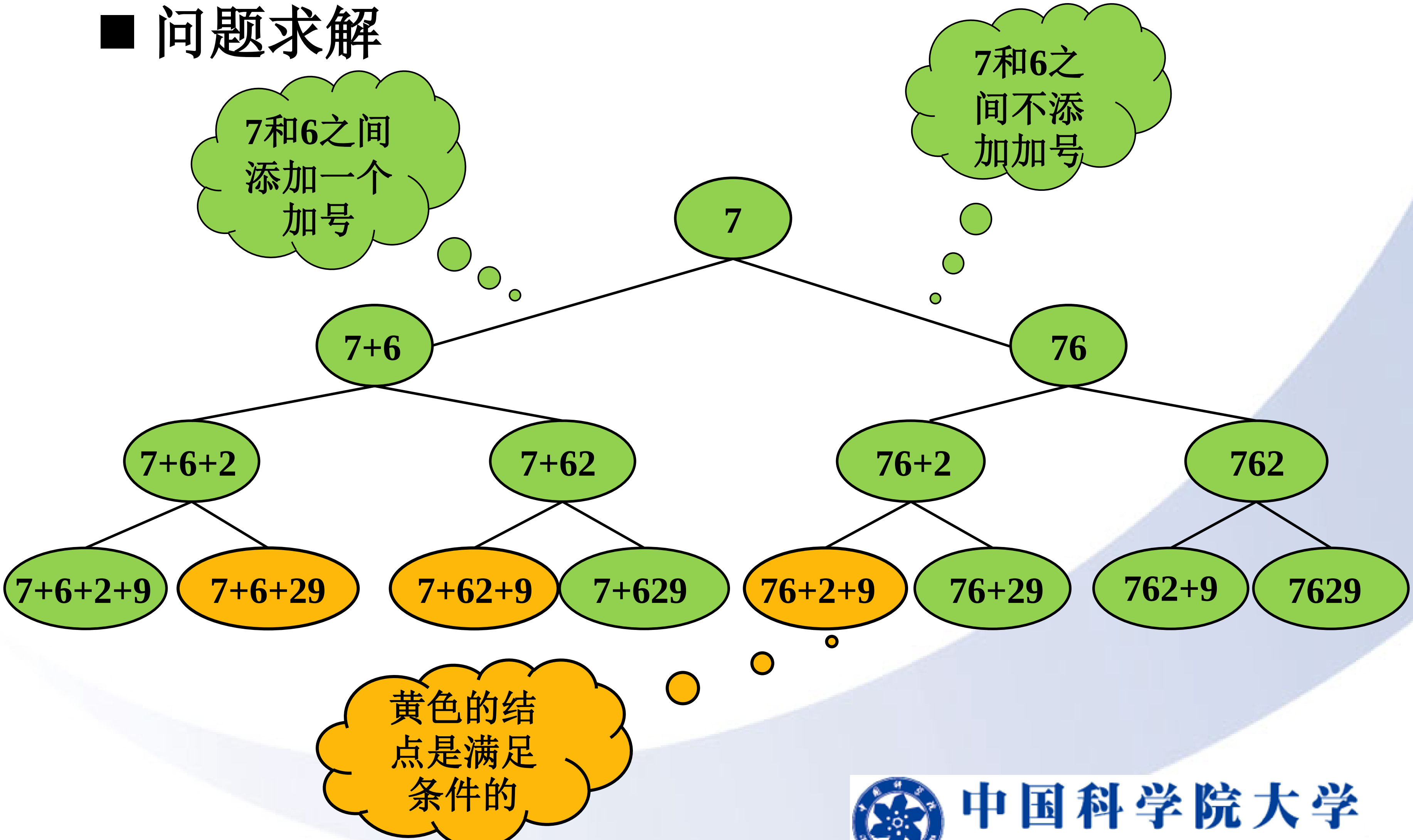
    这时，符合题目要求的结果是：3+12=15

6.7 和最小

- 问题分析
  - 题目要求的就是在每个数字之间：或者填加号，或者什么都不填。
  - 根据这个要求，我们可以从头开始扫描整个数字串，逐个考察是否要填加号，然后检查下一个数字间的位置，直到最后一个数字。
  - 例：
    - 数字7629需要插入2个加号的完整的搜索树。

6.7 和最小

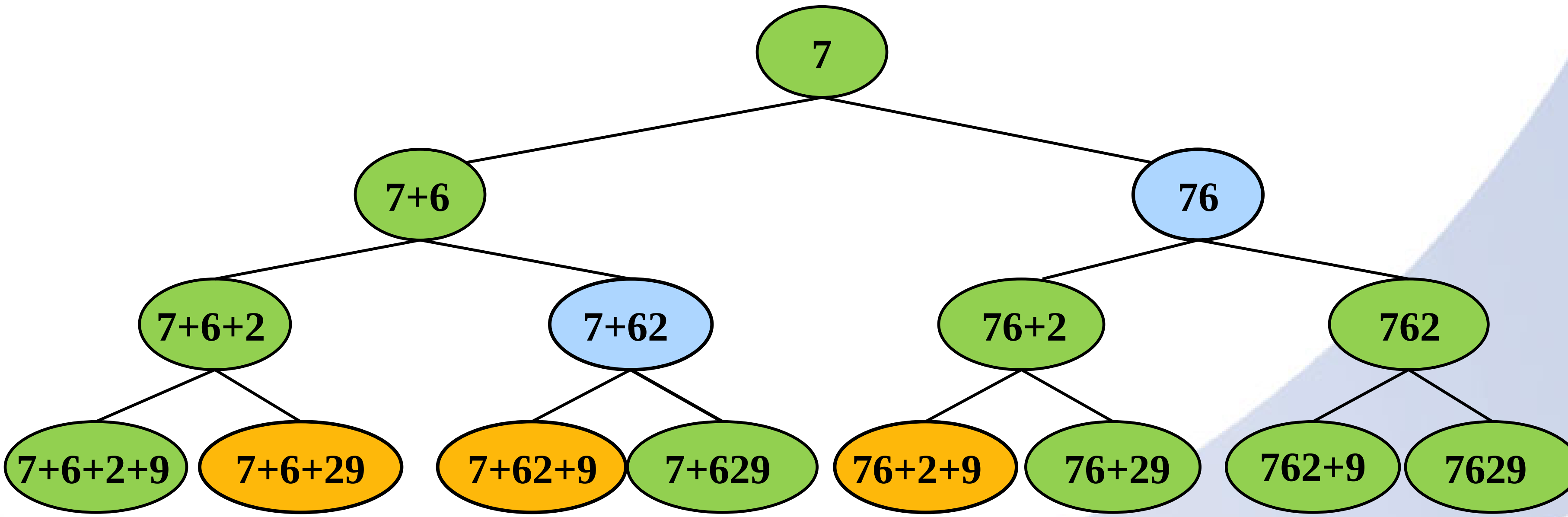
■ 问题求解



7和6之间
添加一个
加号
7和6之
间不添
加加号
7
7+6
76
7+6+2
7+62
76+2
762
7+6+2+9
7+6+29
7+62+9
7+629
76+2+9
76+29
762+9
7629
黄色的结
点是满足
条件的

6.7 和最小

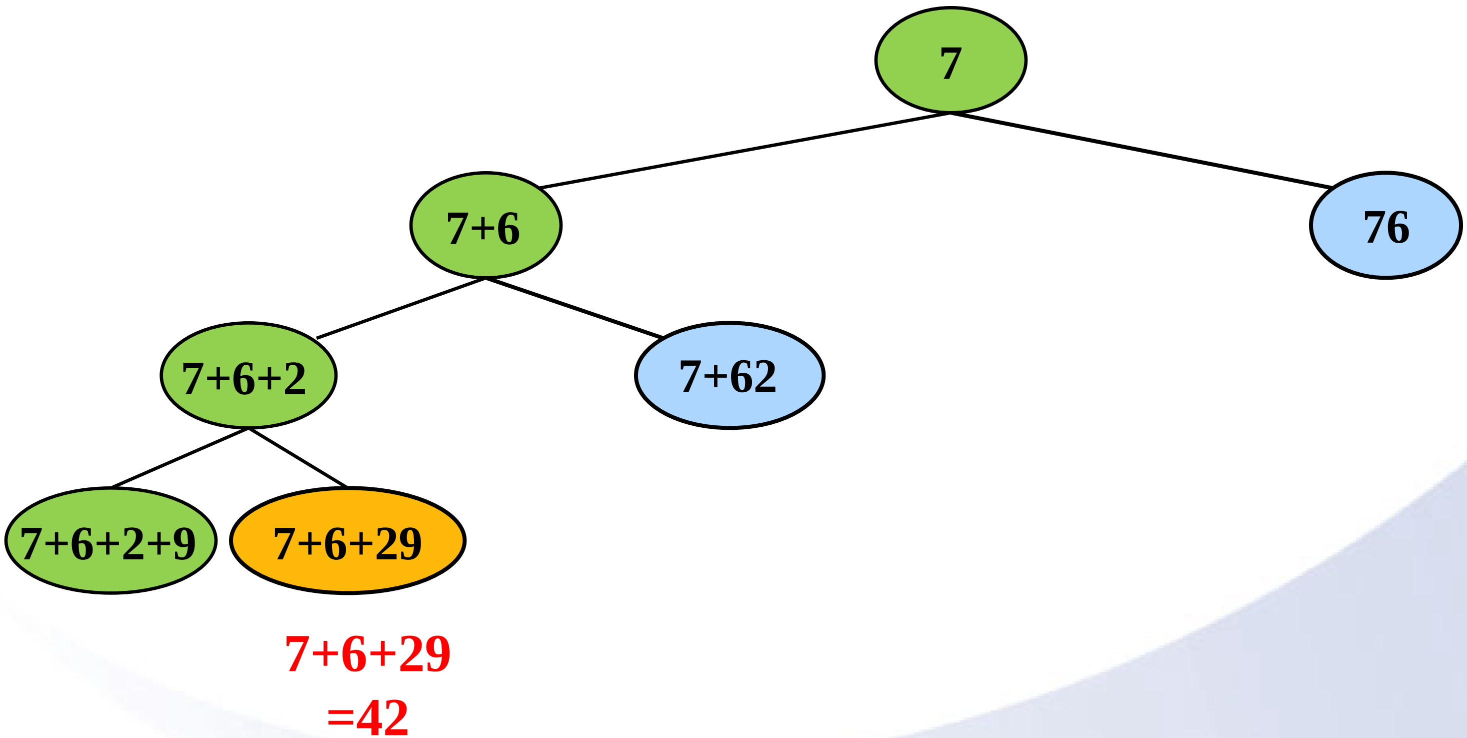
■ 问题求解



蓝色结点的子节点不可能有最优解!

6.7 和最小

■ 问题求解



第六章 回溯法

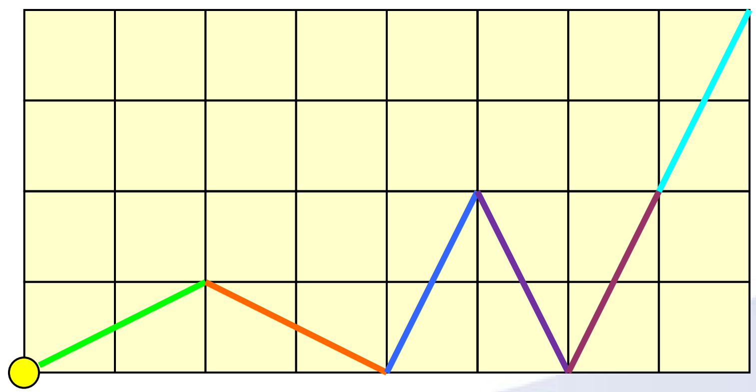
- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题
- 6.6 哈密顿环
- 6.7 和最小
- 6.8 跳马问题

6.8 跳马问题

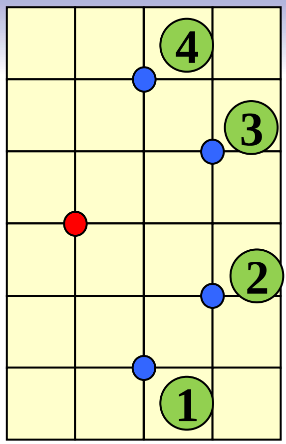
- 问题描述
  - 在n×m棋盘上有一中国象棋中的马：

  ① 马走日字；

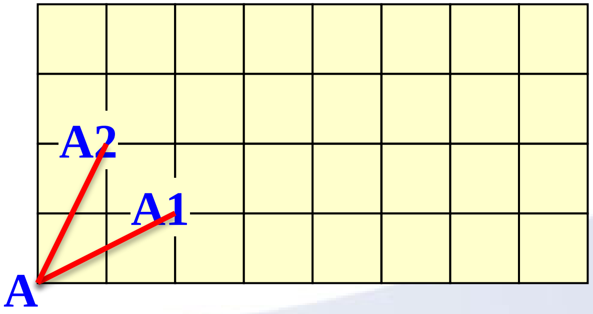
  ② 马只能往右走。
  - 请你找出一条可行路径，使得马可以从棋盘的左下角(1, 1)走到右上角(n, m)。
  - 例：



6.8 跳马问题

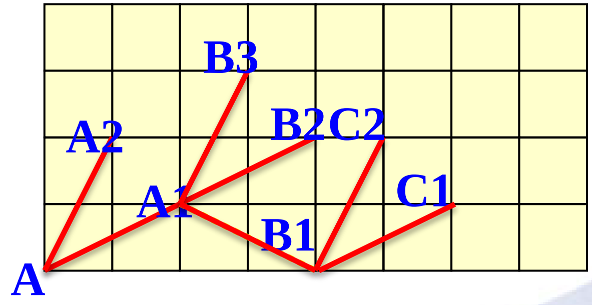


- ■ 问题分析
  - □按题意，马每一步可以有4种走法！
- ■ 搜索过程
  - □当马一开始位于左下角的时候，根据规则，它只有两条线路可以选择（另外两条超出棋盘的范围），我们无法预知该走哪条，故任意选择一条，到达A1。

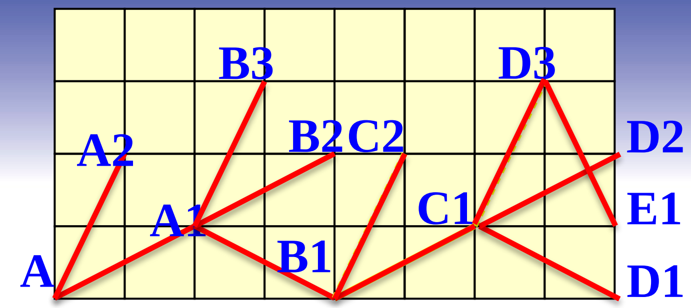


6.8 跳马问题

- ■ 搜索过程
  - □ 当到达A1点后，又有三条线路可以选择，于是再任意选择一条，到达B1。
  - □ 从B1再出发，又有两条线路可以选择，先选一条，到达C1。



6.8 跳马问题



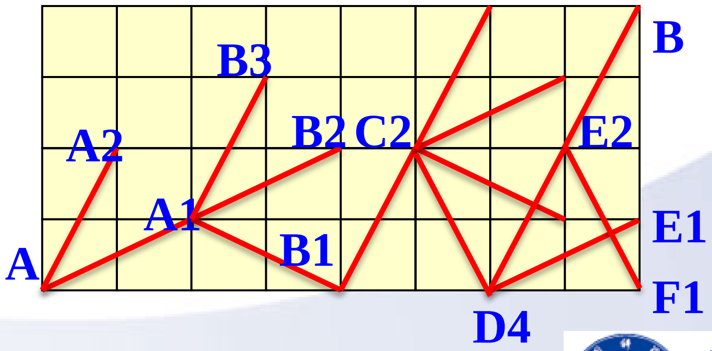
■ 搜索过程

- 从C1出发，可以有三条路径，选择D1。
- 到了D1以后，无路可走且D1也不是最终目标点，因此，选择D1是错误的，退回C1重新选择D2。
- 同样D2也是错误的。
- 再回到C1选择D3。
- D3只可以到E1，但E1也是错误的。
- 返回D3后，没有其他选择，说明D3也是错误的，再回到C1。
- 此时C1不再有其他选择，故C1也是错误的，退回B1，选择C2进行尝试。

中国科学院大学
University of Chinese Academy of Sciences

6.8 跳马问题

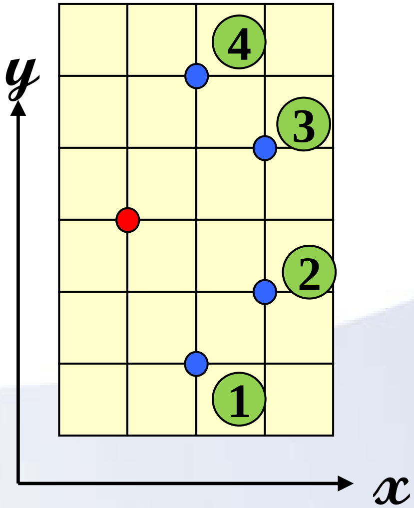
- 搜索过程
  - 从C2出发，有四条路径可以选择，
  - 选择D4，从D4出发又有两条路径，
  - 选择E1错误，返回D4选择E2，
  - 从E2出发有两条路径，先选择F1错误，
  - 返回E2选择B，而B恰好是要到达的目标点，
  - 至此，一条路径查找成功。



6.8 跳马问题

■ 算法实现

- □在无法确定走哪条线路的时候，任选一条线路进行尝试；为方便路径表示，对马可以走到的四个点（方向）都编上号；
- □当从某点出发，所有可能到达的点都不能到达终点时，说明此点是一个死节点，必须回溯到上一个点，并重新选择一条新的线路进行尝试。



6.8 跳马问题

- 算法实现
  - 解空间：为了描述路径，我们最直接的方法就是记录路径上所有点的坐标。
  - 约束条件: 不越界：
    - $(x + dx[i] \le n)$ and $(y + dy[i] > 0)$ and $(y + dy[i] \le m)$

若马所处的位置为(x, y)，则其下一步可以到达的四个位置分别是(x+1, y-2)，(x+2, y-1)，(x+2, y+1)，(x+1, y+2)。

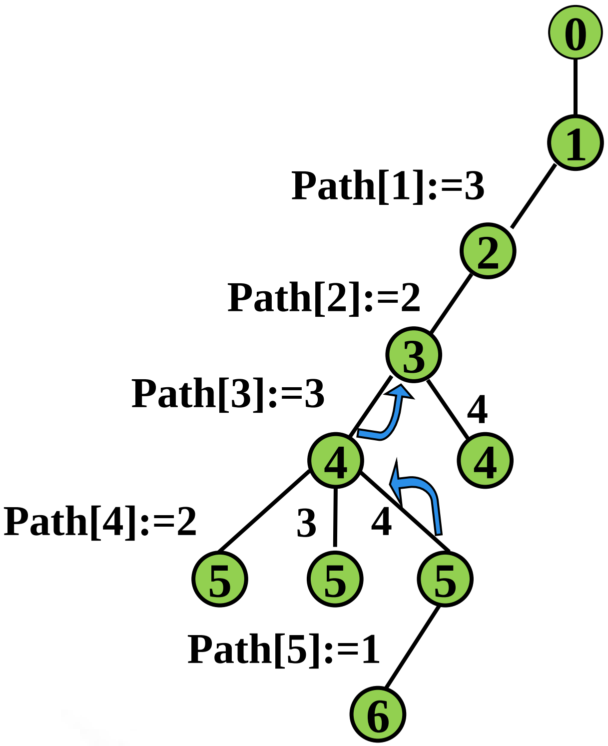
增量数组：
dx = (1, 2, 2, 1)
dy = (-2, -1, 1, 2)

方向t，下一步的位置就是(x+dx[t], y+dy[t])。

path:array[1…m] of integer;
其中，path[i]：表示第i个节点所走的方向

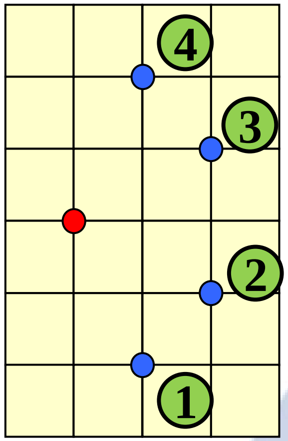
6.8 跳马问题

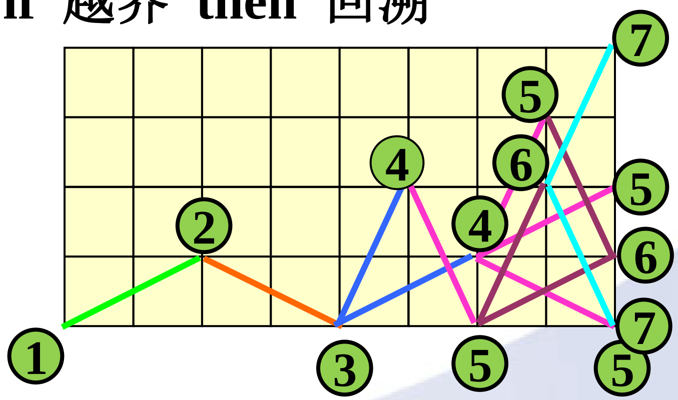
状态树：



算法描述：

1. 产生一种新走法
2. 越界，继续用新走法，直到找到一种走法不越界----不超过4种走法
3. if 不越界 then k<n→k+1
   k=n→一组解
4. if 越界 then 回溯





跳马问题（递归）

```
Begin
    read(n, m);
    x:=1; y:=1;
    search(1);
End.
```

```
procedure search(k: integer); // 递归查找
begin
  for i := 1 to 4 do  // 依次尝试四个方向
    if (x+dx[i] ≤ n) and (y+dy[i] > 0) and (y+dy[i] ≤ m) then //在棋盘上
    begin
      path[k] := i;  // 记录下当前方向
      x := x + dx[i];  y := y + dy[i];  // 修改扩展节点坐标
      if (x = n) and (y = m) then // 是否是目标点
      begin
          output(k); halt;  // 是目标点，输出结果并终止程序
      end
      else
         search(k+1);  // 不是目标点，继续尝试下一步
      // 扩展出的点是死点，回溯
      x := x - dx[i];  y := y - dy[i];  // 恢复扩展节点坐标，状态恢复
    end;
end;
```

第六章 回溯法

- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题
- 6.6 哈密顿环
- 6.7 和最小
- 6.8 跳马问题
- 6.9 装载问题

6.9 装载问题

■ 问题描述

- 一批共n个集装箱要装上2艘载重量分别为C_1和C_2的轮船，其中集装箱i的重量为w_i，且$\sum w_i \le C_1+C_2$
- 要求确定是否有一个合理的装载方案可将n个集装箱装上2艘轮船。
- 如果一个给定装载问题有解，采用下面的策略可得到最优装载方案：
  ① 首先将第一艘轮船尽可能装满
  ✓选取子集，重量和最接近C_1。
  ② 将剩余的集装箱装上第二艘轮船。

中国科学院大学 University of Chinese Academy of Sciences

6.9 装载问题

- 问题分析
  - 例如：n=3, $C_1=C_2=50$,
  - w=[10, 40, 40]时，集装箱1、2装第一艘船，3装第二艘船。
  - w=[20, 40, 40]时，无可行解。
  - 当$\sum w_i = C_1+C_2$时，两艘船的装载问题等价于子集之和问题，即有n个数字，要求找到一个子集使它的和为C_1。

6.9 装载问题

- 问题分析
  - 将第一艘轮船尽可能装满等价于选取全体集装箱的一个子集，重量之和最接近轮船载重量。
  - 由此可知，装载问题等价于以下特殊的0-1背包问题。

$$\max \sum_{i=1}^{n} w_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \le c_1$$

$$x_i \in \{0,1\}, 1 \le i \le n$$

用回溯法设计解装载问题的**O(2^n)**计算时间算法，某些情况下优于动态规划算法。

6.9 装载问题

- ■ 问题求解
  - 口解空间：子集树，完全二叉树
  - 口设定解向量：$(x_1, x_2, \ldots, x_n)$
  - 口约束条件
    - ➢显式约束：$x_i = 0, 1 \quad (i=1, 2, \ldots, n)$
    - ➢隐式约束：

$$\sum_{i=1}^{n} w_i x_i \le c_1$$

cw：是当前载重量，

bestw：当前最优载重量

r：剩余集装箱的重量

```
void backtrack (int i)  {// 搜索第i层结点
    if (i > n)  // 到达叶结点
    {   if (cw>bestw) {   //更新最优解bestx, bestw;;
            for(int j=1; j<=n; j++)
                bestx[j]=x[j]; bestw=cw; }
        return;         }
    r -= w[i];
    if (cw + w[i] <= c) {// 搜索左子树
      x[i] = 1;  cw += w[i];
      backtrack(i + 1);
      cw -= w[i];      }
    if (cw + r > bestw)  { // 搜索右子树
      x[i] = 0;
      backtrack(i + 1);      }
    r += w[i];
}
```

约束函数

限界函数

作业-算法实现5

- 问题描述-算24点
  - 几十年前全世界就流行一种数字游戏，至今仍有人乐此不疲．在中国我们把这种游戏称为“算24点”。您作为游戏者将得到4个1~9之间的自然数作为操作数，而您的任务是对这4个操作数进行适当的算术运算，要求运算结果等于24。
  - 您可以使用的运算只有：+，-，*，/，您还可以使用（）来改变运算顺序。注意：所有的中间结果须是整数，所以一些除法运算是不允许的（例如，(2*2)/4是合法的，2*(2/4)是不合法的）。下面我们给出一个游戏的具体例子：
  - 若给出的4个操作数是：1、2、3、7，则一种可能的解答是1+2+3*7=24。

作业-算法实现5

- 问题描述-算24点
  - 输入
    - 只有一行，四个1到9之间的自然数。
  - 输出
    - 如果有解的话，只要输出一个解，输出的是三行数据，分别表示运算的步骤。其中第一行是输入的两个数和一个运算符和运算后的结果，第二行是第一行的结果和一个输入的数据、运算符、运算后的结果；第三行是第二行的结果和输入的一个数、运算符和“=24”。如果两个操作数有大小的话则先输出大的。
    - 如果没有解则输出“No answer!”

中国科学院大学
University of Chinese Academy of Sciences

作业-算法实现5

- 问题描述-算24点
  - 输入样例
    - 1 2 3 7
  - 输出样例
    - 2+1=3
    - 7*3=21
    - 21+3=24
- 要求
  - 作业提交到课程网站上
  - 用C(C++)或者matlab实现
  - 要有算法的求解说明

END