小马加编信息学教案(二十一)

宽度优先搜索

一. 课程内容

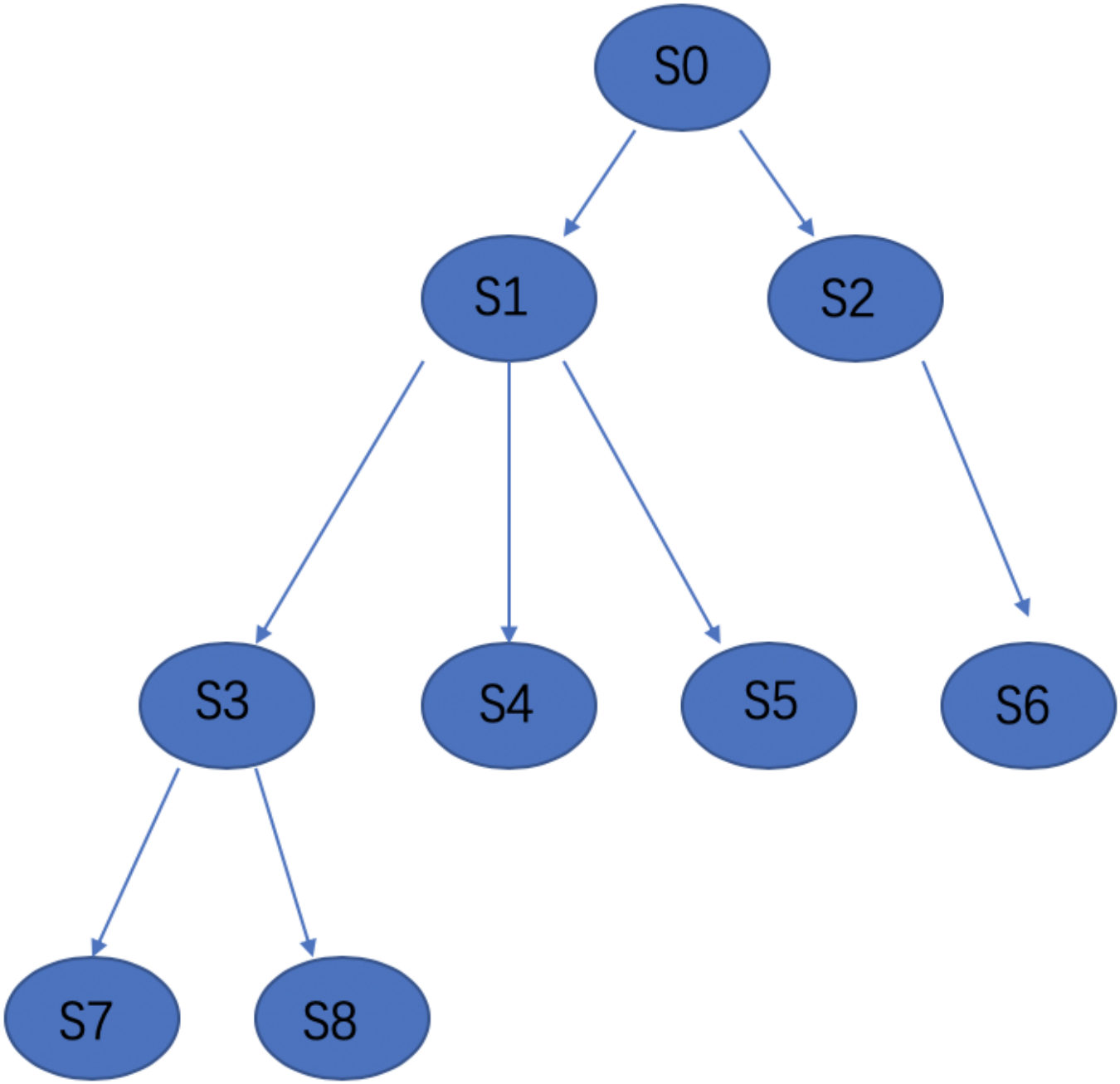
1. 宽度优先搜索简介
2. 宽度优先搜索实现

二. 知识讲解

1. 宽度优先搜索概念

> 宽度优先搜索称为bfs（breadth first search）
> bfs的适用情况与dfs类似。学习完后，不难发现所有的dfs问题都可以使用bfs实现。只不过由于搜索方式的不同效率会有所不同。

> bfs的方式是使用一个队列存储当前搜索到的所有状态，然后取出队头状态s，枚举s所有可行的决策，依次作出枚举到的决策，生成新状态s',s'',s'''.....然后将新状态加入队尾。继续这个过程，直到队列为空。



> 我们记S0,S1,...为状态，箭头为做了决策后的状态转移，S0为初始状态。那么深度优先搜索的搜索过程是
> S0->(S1，S2），S1->(S3，S4，S5），S2->(S6)，S3->(S7，S8）

2. 宽度优先搜索实现

> 伪代码
> 记p为存储状态的队列

```
void bfs()
{
    s0加入队列p
    while(p非空)
    {
        取出p队首状态s
        for(每一种转移方式)
        {
          生成新状态s'
          if(s'是没有生成过的状态)
            将s'加入p队尾
        }
    }
}
```

> 结合例题讲解

> 注意bfs的搜索特点。观察上方状态的转移图，状态是“一层一层”被搜索出来的，即需要做决策数少的状态先被搜索出来。而dfs的特点是一条路走到底。搜索方式的不同导致同一问题使用dfs和bfs效率不同。

> 一般来讲，由于能够用当前解剪枝，dfs在最优解搜索效率高于bfs
> 而bfs由于“一层一层”搜索的特点适合用于一些类似最短路问题

> 在做题时注意比较具体题目中dfs和bfs的优劣

三. 经典例题

1. 分油

> 有3个油瓶，容量分别为10，7，3。开始时，容量10的瓶子里装满了油，其它瓶子是空的。现在一次操作可以选择一个瓶子a，另一个瓶子b，然后将a的油倒入b，直到a为空或b为满。现在你要编程找出一个操作最少的方案，使油被分成5和5两份在容量10和7的瓶子里。

> 如可以通过9次操作完成目标。
> 一种操作方法每次操作完每个瓶子的油量分别为：
> 10 0 0
> 3 7 0
> 3 4 3
> 6 4 0
> 6 1 3
> 9 1 0
> 9 0 1
> 2 7 1
> 2 5 3
> 5 5 0

2. 最短路

> 给出n个点之间是否有边，问从点x到点y至少要经过多少条边?

> 输入格式：
> 第一行三个整数n，x，y
> 下面是一个n*n的01矩阵
> 矩阵第i行第j列为1表示i和j之间有边，否则没有
> 保证（i，j）位置元素和（j，i）位置元素相同

> 输出格式：
> 一个整数表示最少边数。无法到达输出impossible。

3. 飞越原野

> 原野是一个n*m的矩阵，有平地（P）和湖泊（L）两种地形。
> 鸟人现在要从（1，1）到（n，m），途中鸟人只能停留在平地上。
> 每个时间单位里，鸟人可以向上下左右中的一个方向走一格或者飞行。
> 飞行无论飞多远，都只需一个单位时间。
> 飞行的目的地与当前点需要在同一行或一列。
> 鸟人总共能飞行的距离不超过D。
> 定义（x，y）与（x'，y'）的距离为|x-x'|+|y-y'|。
> 求最短时间。
> $n, m, D \leq 100$

> 输入格式：
> 第一行三个整数n，m，D
> 下面一个n*m的矩阵表示地形。保证（1，1）为P。

> 输出格式：
> 一个整数表示答案。
> 若无法到达输出impossible。

样例输入	样例输出
4 4 2 PLLP PPLP PPPP PLLP	5

四. 提高巩固

1.八数码问题

> 一个九宫格，其中8个格子填上了1~8，还有一个空格。
> 每次操作可以将空格和其上下左右中的一个格子交换位置。
> 问从初始状态到目标状态至少要做几次操作。

> 输入格式:
> 两个3*3矩阵，表示初始状态和目标状态
> 0表示空格

> 输出格式:
> 一个整数表示最少操作次数。不可行时输出impossible。

样例输入	样例输出
2 8 3 1 0 4 7 6 5 1 2 3 8 0 4 7 6 5	4

2.棋盘（noip2017普及组）

> 有一个m×m的棋盘，棋盘上每一个格子可能是红色、黄色或没有任何颜色的。你现在要从棋盘的最左上角走到棋盘的最右下角。
>
> 任何一个时刻，你所站在的位置必须是有颜色的（不能是无色的），你只能向上、下、左、右四个方向前进。当你从一个格子走向另一个格子时，如果两个格子的颜色相同，那你不需要花费金币；如果不同，则你需要花费 1 个金币。
>
> 另外，你可以花费 2 个金币施展魔法让下一个无色格子暂时变为你指定的颜色。但这个魔法不能连续使用，而且这个魔法的持续时间很短，也就是说，如果你使用了这个魔法，走到了这个暂时有颜色的格子上，你就不能继续使用魔法；只有当你离开这个位置，走到一个本来就有颜色的格子上的时候，你才能继续使用这个魔法，而当你离开了这个位置（施展魔法使得变为有颜色的格子）时，这个格子恢复为无色。
>
> 现在你要从棋盘的最左上角，走到棋盘的最右下角，求花费的最少金币是多少?

> 输入格式：
> 第一行包含两个正整数m,n，以一个空格分开，分别代表棋盘的大小，棋盘上有颜色的格子的数量。
> 接下来的 n 行，每行三个正整数 x, y, c 分别表示坐标为(x,y)的格子有颜色c。
> 其中 c=1 代表黄色， c=0 代表红色。相邻两个数之间用一个空格隔开。棋盘左上角的坐标为(1,1)，右下角的坐标为(m,m)。

> 棋盘上其余的格子都是无色。保证棋盘(1,1)一定是有颜色的。
> $m \leq 100, n \leq 1000$

> 输出格式：
> 一个整数，表示花费的金币的最小值，如果无法到达，输出-1。

样例输入	样例输出
5 7 1 1 0 1 2 0 2 2 1 3 3 1 3 4 0 4 4 1 5 5 0	8