图搜索和树搜索

```
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier
```

```
function GRAPH-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    initialize the explored set to be empty
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        add the node to the explored set
        expand the chosen node, adding the resulting nodes to the frontier
            only if not in the frontier or explored set
```

先将父亲节点从frontier中取出

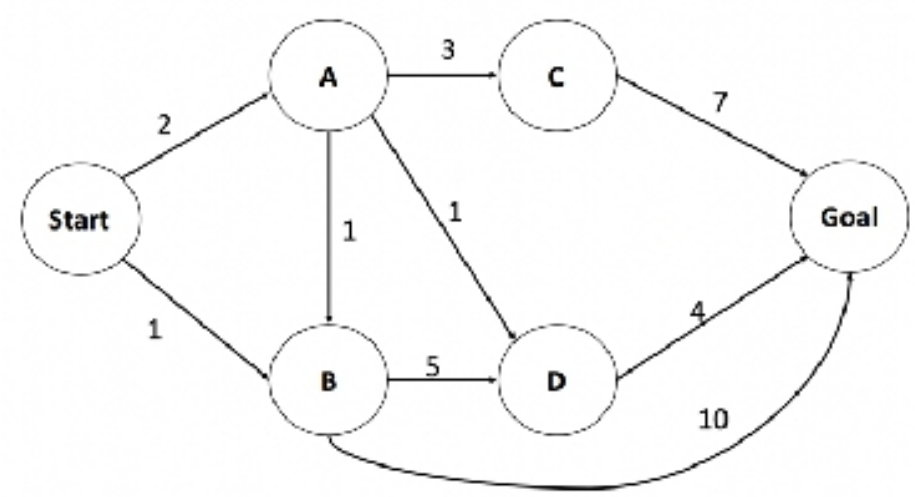
再将孩子节点加入frontier中

队列说明

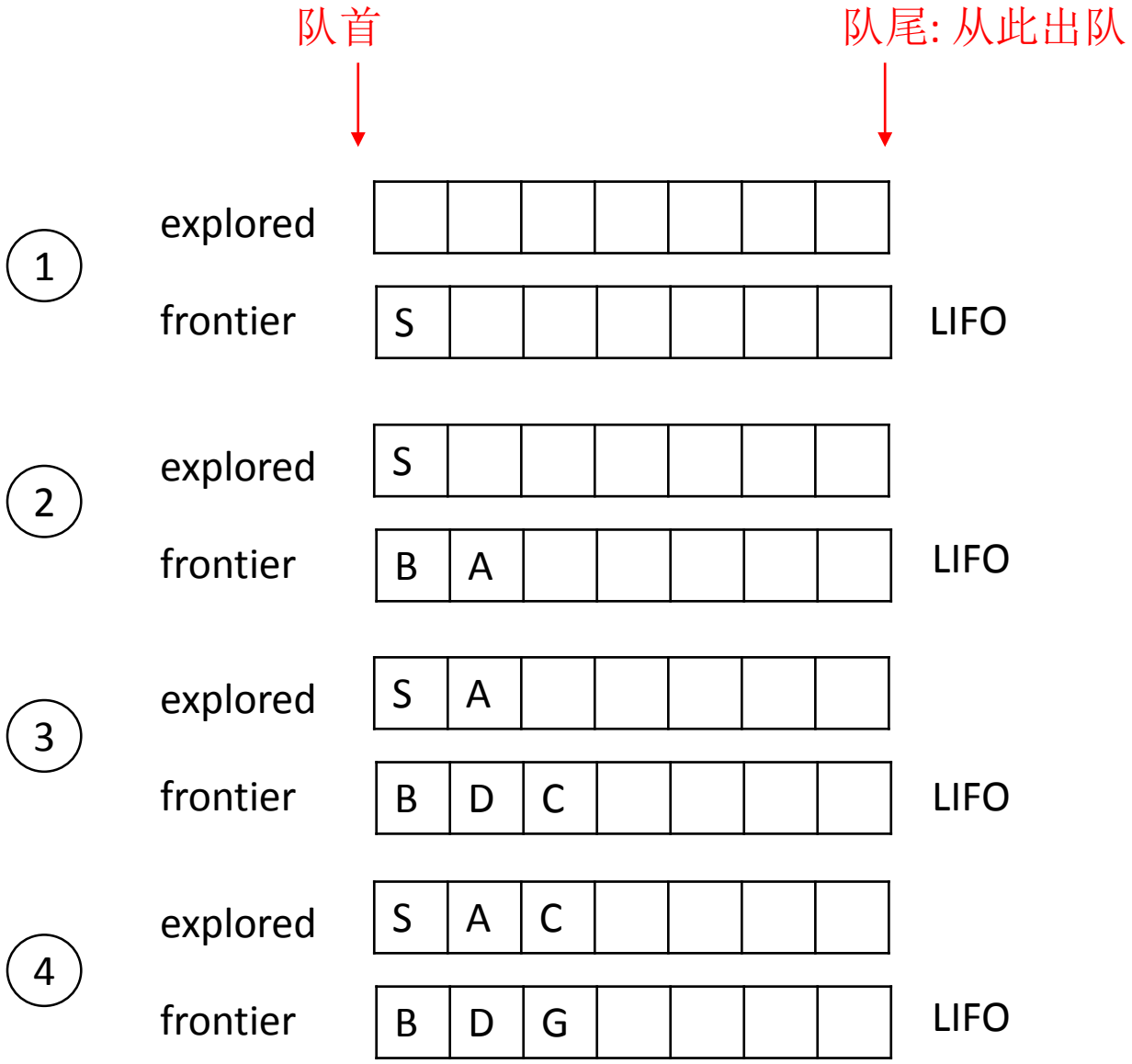
- explored : 始终使用FIFO的队列实现
- frontier : 不同算法采用不同队列：
  - ✓FIFO : 先进先出，每次元素按序进队，从对首开始出队
  - ✓LIFO : 后进先出，每次元素按序进队，从对尾开始出队
  - ✓优先级队列 : 每次元素进队后，按照优先级进行排序。最高优先级的元素出队

DFS

- 基于LIFO的队列实现

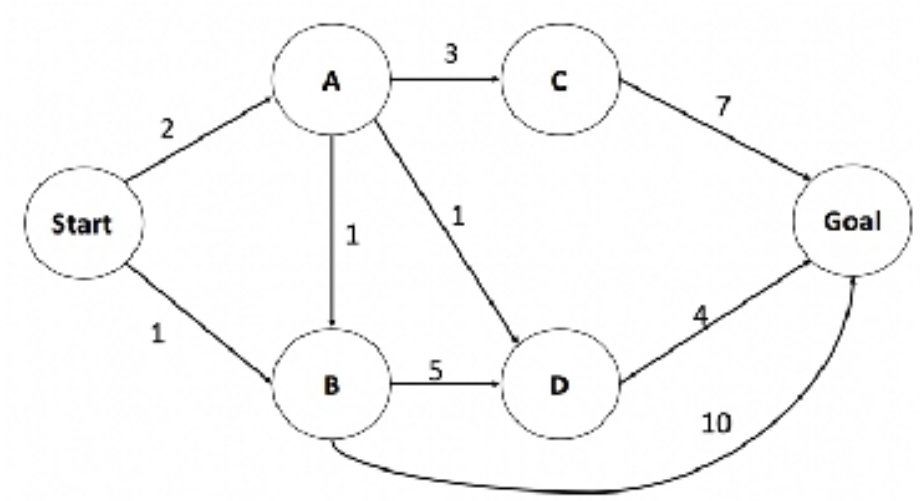


Result path: S->A->C->G
Visited sequence: S->A->C->G



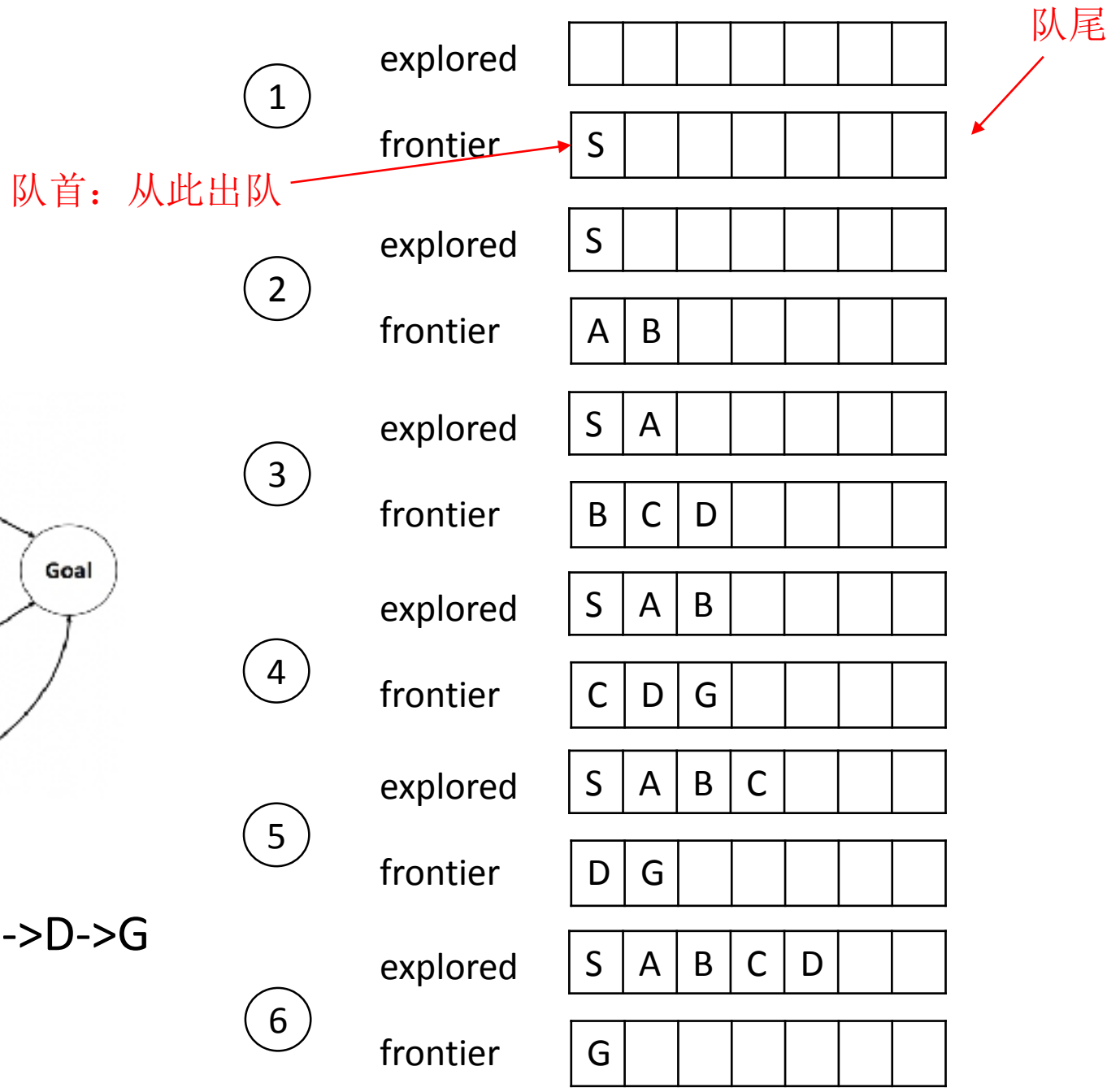
BFS

- 基于FIFO的队列实现



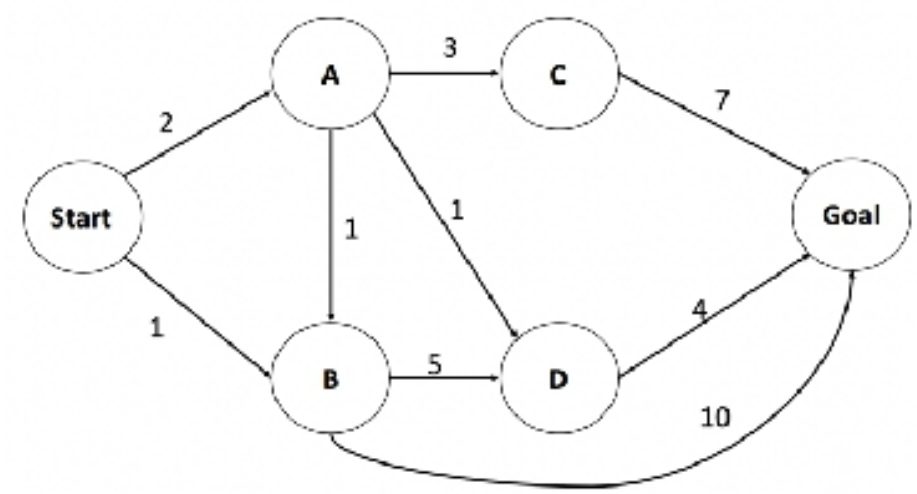
Result path: S->B->G

Visited sequence: S->A->B->C->D->G



UCS

- 优先级队列
- 在找到更小代价时更新优先级信息



Result path: S->A->D->G

Visited sequence: S->B->A->D->C->G

队首：从此出队

队尾

1

explored

frontier S0

2

explored S0

frontier B1 A2

3

explored S0 B1

frontier A2 D5 G10

4

explored S0 B1 A2

frontier D3 C5 G10

5

explored S0 B1 A2 D3

frontier C5 G7

6

explored S0 B1 A2 D3 C5

frontier G7

启发式算法设计

- 真实代价 – g(n)
- 以该节点为起点的最小路径代价
- 该节点到目标节点经过的最少节点数
- 满足可采纳性和一致性情况下的规划求解结果

- 曼哈顿距离
  - 不能作为启发式函数，因为图中的距离与现实距离并非等比例的