重点页 内容总结 重点页

- 宽度优先：往往采用图搜索实现，需要存储所有闭节点，需要的内存随着层数加深而指数增长，但一层层搜索保证找到的是最优解。开节点弹出用“先进先出”队列实现。
- 深度优先：往往采用树搜索实现，只存开节点，需要内存少，某些问题下有可能进入死循环，不能保证找到的是最优解。如果用图搜索实现深度优先，那么它的空间复杂度的优势就没了。开节点弹出用“先进后出”栈实现。
- 深度受限：可以防止搜索太深或者进入死循环，但难设置L，可用迭代加深解决
- 清楚 b d m f(n) g(n)的含义，清楚先进先出的队列（FIFO queue）、先进后出的栈（FILO stack）、优先队列（priority queue）和哈希表（hash）的功能

重点页 内容总结 重点页

- 一致代价：考虑每条边的权重的宽度优先（前面的都假设每条边权重为1）

  f(n) = g(n) 看过去真实发生的花费，开节点弹出用优先队列（priority queue），有最优解

- 贪婪最佳优先：只看未来估计花费h(n)的一致代价搜索，f(n) = h(n)，不一定找到最优解

- A* 搜索：同时看过去真实发生g(n)的和未来估计h(n)花费的一致代价搜索，f(n) = g(n) + h(n)

- 关于启发式函数 h(n)

  一致代价、贪婪、A* 的代码框架是一样的

  - 只要 h(n) <= 未来真实的花费，就能找到最优解！ (h(n)是未来真实花费的下限)
  - 如地图问题中，h(n)是直线飞行距离，肯定<=未来真实的花费，那个A*例子有最优解
  - 如 h(n) = 0的时候，h(n)肯定 <= 未来真实的花费，为 f(n)=g(n)+0 的一致性搜索
  - h(n)设计方面，值越大则搜索的范围越小、搜索越快
    - 松弛法（太松的话搜索范围就大了）、复合式启发函数