

Floyd算法与动态规划

2050825 林鑫宇

Cu_OH_2

Floyd算法简介

- 求解多源最短路问题
- 有/无向图，正/负边权
- 时间复杂度 $O(n^3)$ ，空间复杂度 $O(n^2)$

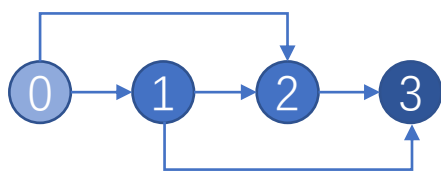
```
for (int k = 1; k <= n; ++k) {  
    for (int i = 1; i <= n; ++i) {  
        for (int j = 1; j <= n; ++j) {  
            dis[i][j] = std::min(dis[i][j], dis[i][k] + dis[k][j]);  
        }  
    }  
}
```

动态规划简介

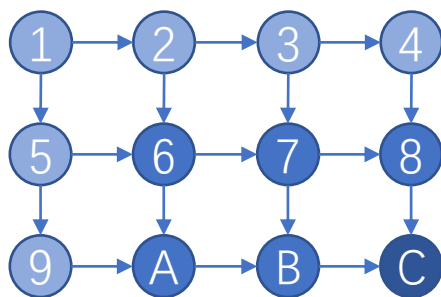
1. 划分子问题，定义**状态**
2. 寻找状态间**转移关系**
3. 按**顺序**求解所有子问题

DAG
(有向无环图)

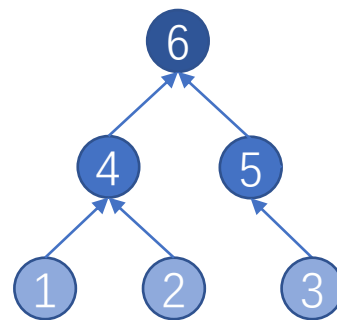
线性
二维
树形
...



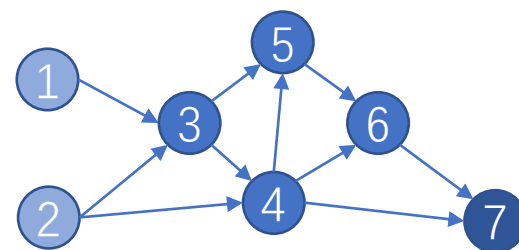
线性



二维



树形



一般DAG

Floyd算法

1. 划分子问题，定义状态

- 总问题：对任意两个结点 i, j ，求从 i 出发经过图中任意点到达 j 的最短路径长度
- 子问题 k ：对任意两个结点 i, j ，求从 i 出发只经过前 k 个结点 ($1 \leq k \leq n$) 中的点到达 j 的最短路径长度
- 状态 (k, i, j) ：从 i 出发只经过前 k 个结点 ($0 \leq k \leq n$) 中的点到达 j 且路径最短
- 定义状态 (k, i, j) 的最短路径长度为 $\text{min_dist}(k, i, j)$

Floyd算法

以 $n = 3$ 为例，状态分布如下

(0,1,1)	(0,1,2)	(0,1,3)
(0,2,1)	(0,2,2)	(0,2,3)
(0,3,1)	(0,3,2)	(0,3,3)

初始状态集

(1,1,1)	(1,1,2)	(1,1,3)
(1,2,1)	(1,2,2)	(1,2,3)
(1,3,1)	(1,3,2)	(1,3,3)

子问题1状态集

(2,1,1)	(2,1,2)	(2,1,3)
(2,2,1)	(2,2,2)	(2,2,3)
(2,3,1)	(2,3,2)	(2,3,3)

子问题2状态集

(3,1,1)	(3,1,2)	(3,1,3)
(3,2,1)	(3,2,2)	(3,2,3)
(3,3,1)	(3,3,2)	(3,3,3)

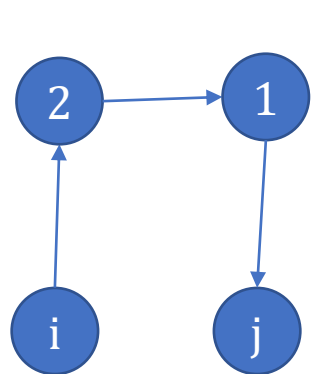
子问题3状态集
(总问题)

Floyd算法

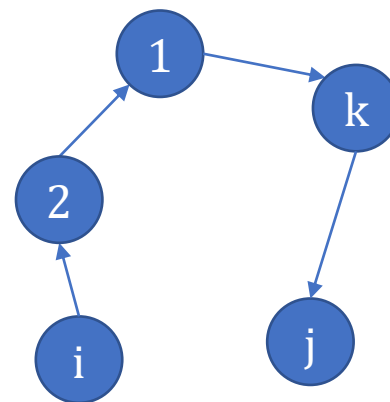
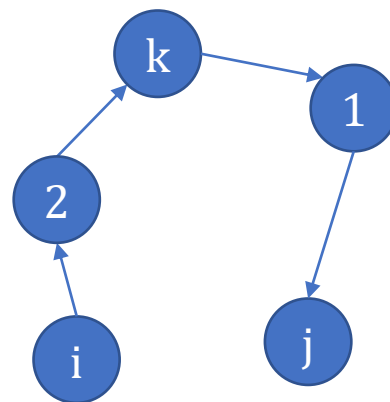
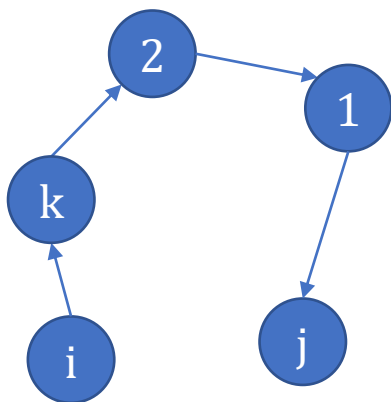
2. 寻找状态间转移关系

i.e. 寻找每个子问题与其子问题间的关系

- 子问题 k 如何由子问题 $k - 1$ 构成?
- 考虑将结点 k 插入子问题 $k - 1$ 所有最优路径中的任意两点之间, 并与插入前路径取最优者

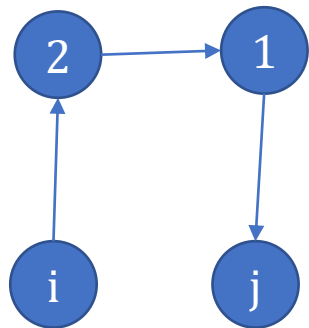


插入前 (可能有多种)

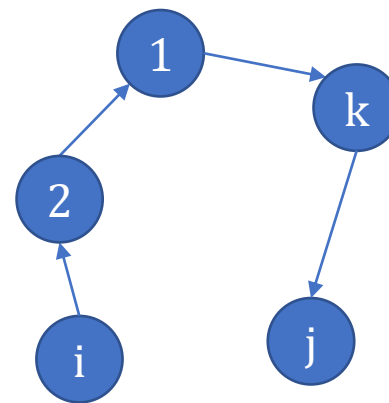
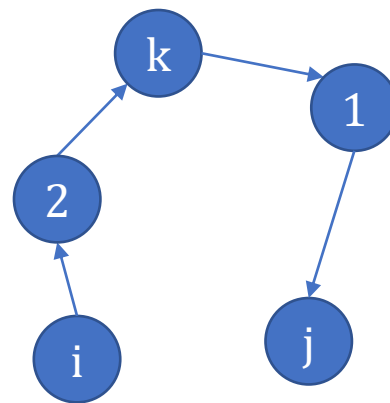
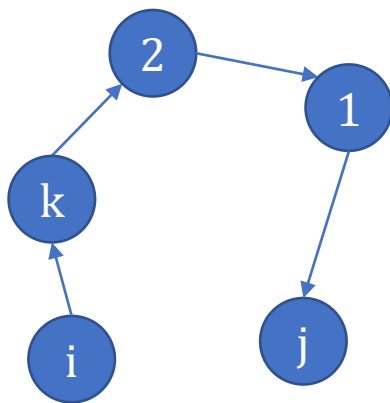


插入后

Floyd算法



插入前（可能有多种）为
 $\min_dis(k-1, i, j)$



插入后最短为
 $\min_dis(k-1, i, k) + \min_dis(k-1, k, j)$

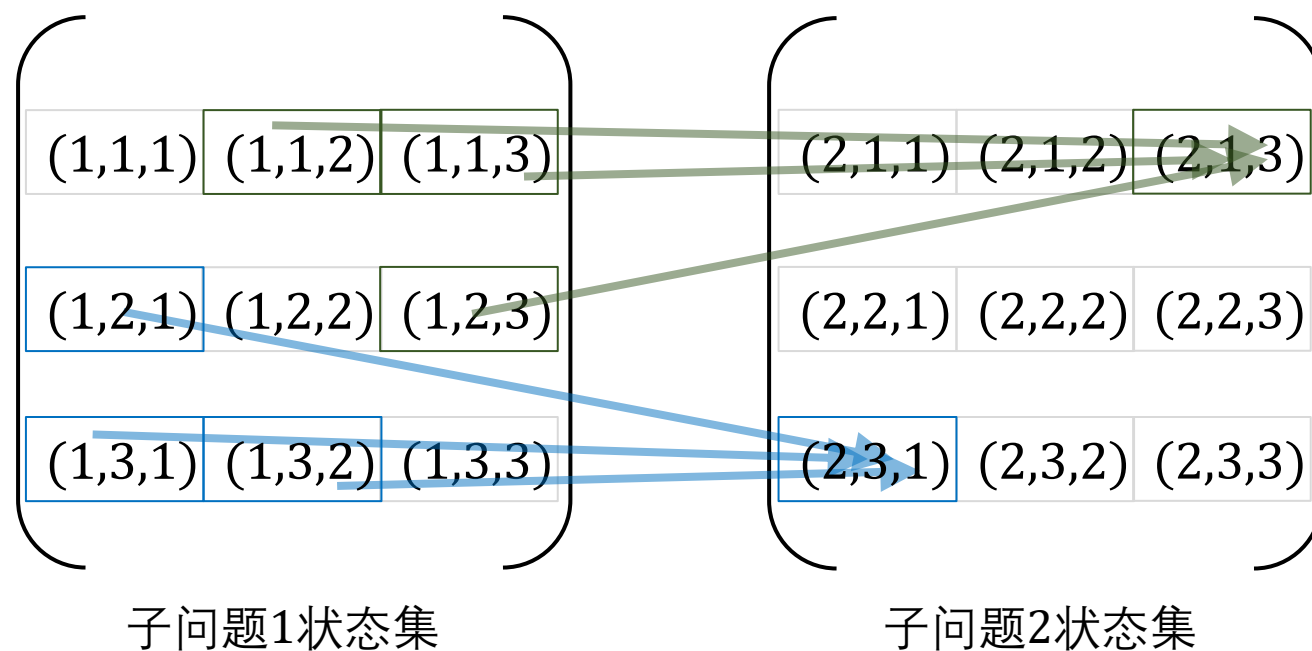
- 插入结点 k 后， i 与 k 之间、 k 与 j 之间依然只经过前 $k-1$ 个结点中的点
- 实际上不用考虑所有插入后的情况，因为插入后的所有情况中一定包含 i, k 间距、 k, j 间距都取到最小值的情况，所以插入后 i, j 最短距离一定为 $\min_dis(k-1, i, k) + \min_dis(k-1, k, j)$
- 将插入前后的结果取最小值即可构成新状态结果，状态转移方程为：

$$\min_dis(k, i, j) = \min(\min_dis(k-1, i, j), \min_dis(k-1, i, k) + \min_dis(k-1, k, j))$$

Floyd算法

以 $n = 3, k = 2$ 为例

考虑状态 $(2, 1, 3)$ 和 $(2, 3, 1)$ 时的转移关系如下



所有状态及状态间关系将形成一个DAG，可以按照拓扑序递推

Floyd算法

3. 按顺序求解所有子问题

- 初始化：若 i 到 j 有边，则赋 $dis[0][i][j]$ 为其中的最小边权；若 i 到 j 没有边，则赋 $dis[0][i][j]$ 为一个极大的数（例如 $0x3f3f3f3f$ ）代表正无穷。这个极大数既要足够大，达到正常距离之和无法达到的数量级；又要保证相加不会越界。

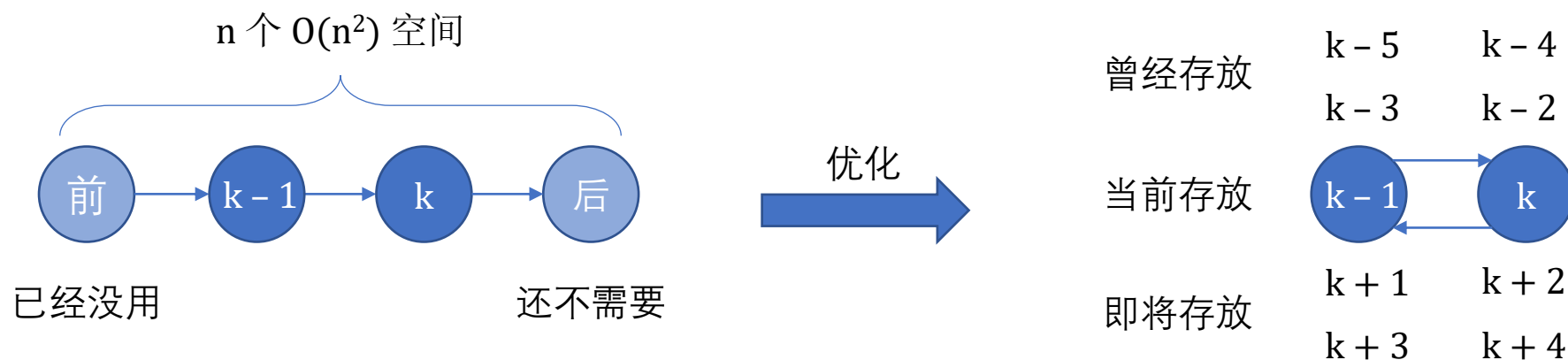
- 代码（C++）：

```
for (int k = 1; k <= n; ++k) {  
    for (int i = 1; i <= n; ++i) {  
        for (int j = 1; j <= n; ++j) {  
            dis[k][i][j] = std::min(dis[k - 1][i][j], dis[k - 1][i][k] + dis[k - 1][k][j]);  
        }  
    }  
}
```

空间优化

动态规划的经典空间优化技巧：滚动数组

- 在状态转移时，子问题 k 的状态只由子问题 $k - 1$ 的状态转移而来，因此递推过程中同一时刻内存中只需要存储两个子问题的状态，新状态信息可以直接覆盖旧状态信息。
- 此时已将空间复杂度优化至 $O(2 \times n \times n) = O(n^2)$ 。

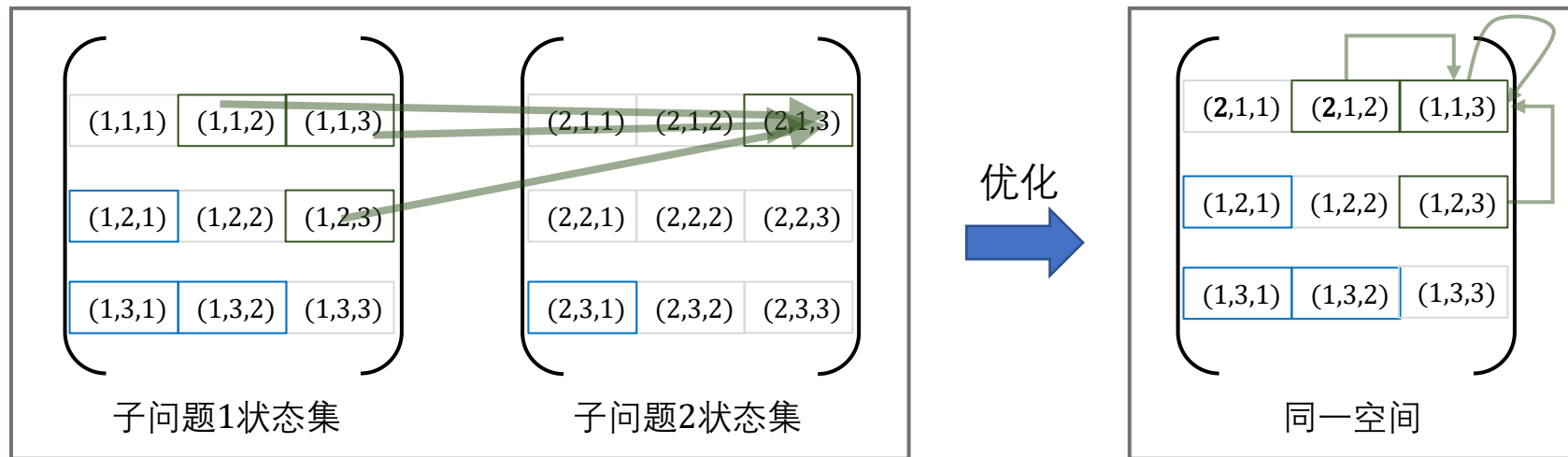


空间优化

更进一步

继续考虑转移过程。

观察转移方程：



$$\text{min_dis}(k, i, j) = \min(\text{min_dis}(k-1, i, j), \text{min_dis}(k-1, i, k) + \text{min_dis}(k-1, k, j))$$

发现其实有：

$$\text{min_dis}(k-1, i, k) = \text{min_dis}(k, i, k)$$

$$\text{min_dis}(k-1, k, j) = \text{min_dis}(k, k, j)$$

也就是说在转移过程中，即使 $\text{min_dis}(k-1, i, k)$ 或 $\text{min_dis}(k-1, k, j)$ 已经被 $\text{min_dis}(k, i, k)$ 或 $\text{min_dis}(k, k, j)$ 覆盖， $\text{min_dis}(k, i, j)$ 依然可以从同一位置取数值进行转移而不影响结果。

这意味着我们可以只用一个 n^2 空间存储所有状态信息，在其内部直接进行转移，削去空间的第一维。

空间优化

```
dis[k][i][j] = std::min(dis[k - 1][i][j], dis[k - 1][i][k] + dis[k - 1][k][j]);
```



```
dis[(k % 2)][i][j] = std::min(dis[(k - 1) % 2][i][j], dis[(k - 1) % 2][i][k] + dis[(k - 1) % 2][k][j]);
```



```
dis[i][j] = std::min(dis[i][j], dis[i][k] + dis[k][j]);
```

可行性分析

- 最优子结构 (Bellman最优性原理)

- 母问题最优解一定包含子问题最优解, i.e. 子问题最优解一定能构成母问题最优解。

- 证: 记 $\text{path}(k, i, j)$ 为从 i 到 j 只经过前 k 个点的最短路。若 $\text{path}(k, i, j)$ 不经过 k , 则等于 $\text{path}(k-1, i, j)$; 若经过 k , 则一定可以拆分成 $\text{path}(k-1, i, k)$ 和 $\text{path}(k-1, k, j)$ 。所以只经过前 k 个点的最短路一定由只经过前 $k-1$ 个点的最短路构成。

- 状态无后效性 (Markov性质)

- 母问题的求解只与子问题的结果有关, 与子问题结果的获得过程无关。

- 经过前 k 个点的最短路长度与只经过前 $k-1/k-2/\dots/1$ 个点的最短路长度有关, 无论前驱状态的最短路径形态如何, 一定都可以用于转移, 构成后继状态的最短路径, 很显然符合无后效性特点。

谢谢观看

2050825 林鑫宇

Cu_OH_2