数据结构与算法图论算法

陈宇琪

2020 年 8 月 1 日

摘要

主要内容：最短路、最小生成树、图上动态规划。

DDL：2020-06-21

目录

1 Bellmanford 算法

1.1 边松弛

假设一条边满足 $d[e.v] > d[e.u] + e.w$，则更新 $d[e.v]$。

1.2 队列优化算法描述

- 初始将起点放入队列，并进入循环。
- 每次选取队首顶点 u。
- 对所有从 u 出发的边 $u \to v$ 进行变松弛。
- 如果节点 v 被更新，且 v 入队次数小于 n，则将 v 放入队尾。
- 如果一个点加入队列 n 次，则说明图中存在环。

* 详细算法可以参考 CSDN。

2 填表题

表 1: 四种最短路算法对比

	算法实质（贪心/动规/其他）	适用范围（单点源/多点源）	边权范围（正边权/皆可）	能否判负环（是/否）	算法复杂度
Dijkstra					
SPFA					
Floyd					
*Bellmanford					

* 请自学 Bellmanford 算法。

3 简答题

1、给出一个不等式组：

$$\begin{cases} x_1 - x_2 \leq 0 \\ x_3 - x_1 \leq 5 \\ x_3 - x_2 \leq 6 \\ x_2 - x_4 \geq 0 \end{cases}$$

（1）构造方程组对应的差分约束系统。
（2）根据图的性质判断方程组是否有解？（简要说明理由）
*（3）构造原方程组的一组解。（修改差分约束系统的图结构，加入一个虚拟节点）
2、给出一个不等式组：

$$\begin{cases} x_3 - x_2 \leq 0 \\ x_2 - x_1 \leq 5 \\ x_3 - x_1 \geq 6 \end{cases}$$

（1）构造方程组对应的差分约束系统。

（2）根据图的性质判断方程组是否有解？（简要说明理由）

3、仔细阅读 1.2 中算法描述，并回答问题：

（1）什么是简单路径？

（2）为什么一个点只需要加入队列 n 次？（提示：最短路是简单路径）

（3）简述 Bellmanford 判负环的原理。

*（4）根据 1.2 描述的 Bellmanford 算法是否能够找出负环上的所有点？如果不能，请修改 Bellmanford 算法。（如果有多个负环，请找出所有最短路径长度为负无穷的点）

4、描述 Floyd 算法的状态设计和状态转移方程的具体含义。

4 编程题

1、（完整代码 + 算法描述）使用 Dijkstra 算法计算从 s 到 t 的最短路和次短路。

提示：请使用堆优化的 Dijkstra 算法，算法描述可以参考 PPT。

2、（完整代码 + 算法描述）假设给定一个有向无环图，求从 s 出发到 t 的简单路径条数。

路径计数：假设对于两条从 s 到 t 的简单路径，只要经过的边有一条不一样，则两条路径认为是不同的。

提示 1：你可以认为图中不存在重边，如果你能够处理重边当然更好。

提示 2：对于有向无环图而言，常见做法是先求拓扑排序。

提示 3：一个简单版本，你可以认为所有边 (u,v) 满足 $u<v$。

5 参考答案

5.1 填表题

表 2: 四种最短路算法对比

	算法实质 （贪心/动规/其他）	适用范围 （单点源/多点源）	边权范围 （正边权/皆可）	能否判负环 （是/否）	算法复杂度
Dijkstra	贪心	单点源	正边权	否	$O(V\times log(E))$
SPFA	其他	单点源	皆可	是	$O(V\times E)$
Floyd	动规	多点源	皆可	否	$O(V^3)$
*Bellmanford	其他	单点源	皆可	是	$O(V\times E)$

* 备注：Floyd 也可以判负环的。

5.2 简答题

1、给出一个不等式组：

$$\begin{cases} x_1-x_2\le 0 \\ x_3-x_1\le 5 \\ x_3-x_2\le 6 \\ x_2-x_4\ge 0 \end{cases}$$

（1）构造方程组对应的差分约束系统。

$$\begin{cases} x_1 - x_2 \le 0 \\ x_3 - x_1 \le 5 \\ x_3 - x_2 \le 6 \\ x_4 - x_2 \le 0 \end{cases}$$

（2）根据图的性质判断方程组是否有解？（简要说明理由）
方程组的差分约束网络：

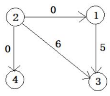

图 1: Ans 1.1

由于上图不存在负环。所以方程组有解。
*（3）构造原方程组的一组解。（修改差分约束系统的图结构，加入一个虚拟节点）
修改后的差分约束网络：

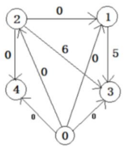

图 2: Ans 1.3

从超级源点 0 到每个点的最短路都是 0，所以 (0,0,0,0) 是原方程组的一组解。
2、给出一个不等式组：

$$\begin{cases} x_3 - x_2 \le 0 \\ x_2 - x_1 \le 5 \\ x_3 - x_1 \ge 6 \end{cases}$$

（1）构造方程组对应的差分约束系统。

$$\begin{cases} x_3 - x_2 \le 0 \\ x_2 - x_1 \le 5 \\ x_1 - x_3 \le -6 \end{cases}$$

（2）根据图的性质判断方程组是否有解？（简要说明理由）
方程组的差分约束网络：

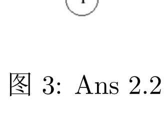

图 3: Ans 2.2

图中存在负环，方程组无解。
3、仔细阅读 1.2 中算法描述，并回答问题：
（1）什么是简单路径？

不经过重复顶点的路径。

（2）为什么一个点只需要加入队列 n 次？（提示：最短路是简单路径）

最短路径一定是简单路径，简单路径的长度最多是 $n-1$。

（3）简述 Bellmanford 判负环的原理。

因为如果存在最短路径，则松弛 $n-1$ 次必然能够找到最短路，如果松弛 $n-1$ 次之后还可以继续松弛，则肯定存在负环。

*（4）根据 1.2 描述的 Bellmanford 算法是否能够找出负环上的所有点？如果不能，请修改 Bellmanford 算法。（如果有多个负环，请找出所有最短路径长度为负无穷的点）

简单修改一些 Bellmanford 算法即可。假设松弛 $n-1$ 次之后的“最短路”长度保存在 d_1 数组中，再松弛 $n-1$ 次之后将“最短路”长度保存在 d_2，如果点 v_i 不在任何一个负环上，则必有 $d_1[v_i]=d_2[v_i]$。反之，如果 $d_1[v_i]\neq d_2[v_i]$，则 v_i 在某个负环上。

4、描述 Floyd 算法的状态设计和状态转移方程的具体含义。

$f[i][j][k]$ 表示从 i 到 j，中间之间经过 $\{v_1,...,v_k\}$ 的最短路。

对于 $f[i][j][k]$ 而言，可以选择经过 v_k 或者不经过 v_k，所以转移为：

$f[i][j][k]=min(f[i][k][k-1]+f[k][j][k-1],f[i][j][k-1])$。

使用类似滚动数组的思想，可以省略数组的第三个维度。

5.3 编程题

1、（完整代码 + 算法描述）使用 Dijkstra 算法计算从 s 到 t 的最短路和次短路。

对于 k 短路问题而言，每一个节点维护一个大小为 k 的容器，类似 Dijkstra 的思想，每次从优先级队列中 pop 出来的元素就是这个节点的一个最短路的解，当一个节点被 pop 出来 k 次，则这个节点不必继续进行松弛。时间复杂度 $O(k\times v\times log(e))$。

2、（完整代码 + 算法描述）假设给定一个有向无环图，求从 s 出发到 t 的简单路径条数。

首先求这个图的拓扑排序并按照拓扑排序对顶点重新编号，使得 $1,2,...,n$ 是这个新图的一个拓扑排序。

假设这时候计算从 s 到 t 的路径条数，则如果 $s>t$，则答案为 0，否则可以用 dp 求解。

$$dp[t]=\sum_{(u,t)\in E\& u<t}dp[u]$$

初始化：$dp[s]=1$，最终的答案为 $dp[t]$，复杂度为 $O(E)$。