ISSN 1009-3044

E-mail：xsjl@dnzs.net.cn

Computer Knowledge and Technology 电脑知识与技术

Vol.17, No.30 October 2021

http：//www.dnzs.net.cn

Tel：+86-551-65690963 65690964

实例解析Bellman-ford和Spfa算法

周鑫①，张晶②

（1.山东省泰安第一中学，山东 泰安 271000；2.聊城大学东昌学院，山东 聊城 252000）

摘要：Bellman-ford和Spfa是解决最短路问题的基本算法，是信息学奥赛教学的基本内容。由于算法抽象性和逻辑性强，

教学过程中学生对其基本原理、实现过程理解困难，导致无法灵活运用解决问题。该文旨在用具体实例结合图表对算法

执行过程进行详细解析，深刻剖析了算法的优化原理，有效解决了学生理解和应用困难的问题。

关键词：Bellman-ford；Spfa；算法解析

中图分类号：TP312 文献标识码：A

开放科学（资源服务）标识码（OSID）：

文章编号：1009-3044（2021)30-0079-03

1 前言

3

4

5

6

7

1

5

2

2

4

3

3

3

5

5

5

4

-2

6

Bellman-Ford 算法由查理德·贝尔曼和莱斯特·福特创立

的，其基本思想是利用松弛原理反复对边集中的每条边进行松

弛迭代操作，同时更新顶点集合中每个顶点的最短路径值并记

录其最短路径上的前驱结点，达到收敛时停止迭代操作[1]。由

于反复对边集中的每条边进行松弛，因此产生了很多冗余的松

弛操作，造成时间复杂度较高。Spfa算法针对这一问题进行了

优化，其核心思想是用FIFO队列保存已经被松弛过的顶点，只

处理入队的顶点，并且不断迭代以求得最短路径。因此，深刻

理解 Bellman-Ford 算法有助于充分理解和应用 Spfa 算法解决

2

2 Bellman-Ford算法题解

用

d数组来存储1号顶点到其余各点的路径值。

初始化如下表：

顶点a

d[a]

1

0

2

∞

3

∞

4

∞

5

∞

[2]

实际问题 。下面我们用具体实例来展开探讨这两个算法的实

现过程。

根据边给出的顺序，先处理第一条边“2-4-3”即判断一下

3

d[4]

d[2]+3

d[4] d[2]

是否大于

，由于此时

和

都是无穷大，因此这条

2

3

4

1-2-（- 1）”，我们发现 d

-1

1

边松弛失败。接下来处理第二条边“

[2] > d[1] + (-1) ，通过这条边可以使d[2]的值从∞变为 -1 ，所 以

这个点松弛成功。我们可以用同样的方法来处理第三条边到

第七条边，对所有的边进行一遍松弛操作后的结果如下：

第一轮对所有边进行松弛以后，结果如下表所示：

6

-2

2

5

5

4

顶点a

d[a]

1

0

2

-1

3

-3

4

∞

5

5

图1 最短路径实例

第二轮对所有边进行松弛以后，结果如下表所示：

例题：如图1所示，求1号顶点到其余各顶点的最短距离。

我们用d 数组记录起点到其余各点的最短路径值，用s、e、t

三个数组来存储边的信息。例如第 i 条边存储在 s[i]、e[i]、t[i]

中，表示从顶点s[i]到e[i]这条边的权值为t[i]。

顶点a

d[a]

1

0

2

-1

3

-3

4

2

5

4

在这轮松弛中，通过“

2 4 3”(2→4)这条边，更新了1号顶点

给出边的顺序如下表：

到

4号顶点的距离(d[4]) 。这条边在第一轮松弛失败，却在第二

轮松弛成功。原因是在第一 轮松弛过后，1号顶点到 2号顶点

的距离(d[2]) 已经发生了变化，这一轮再通过“2 4 3”(2-→4)这

条边进行松弛的时候，可以使 1 号顶点到 4 号顶点的距离(d

[4]) 的值变小。也就是说，第一轮遍历图中所有边进行松弛操

边号

起点

终点

权值

3

-1

1

2

2

1

4

2

收稿日期：2021-07-15

基金项目: 基于计算机编程的PBL教学模式对高中生自主学习能力的培养研究（项目编号：TJK2019ZD051）

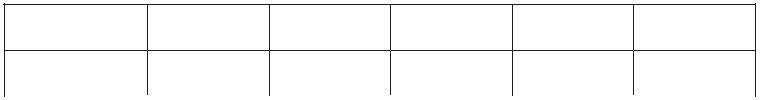
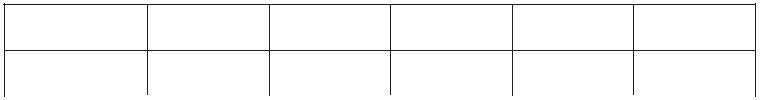
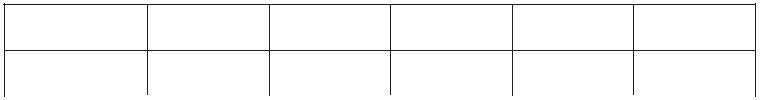
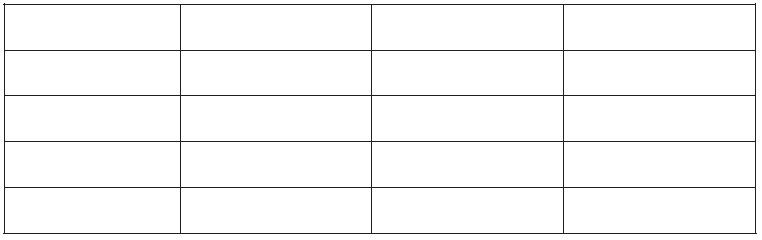
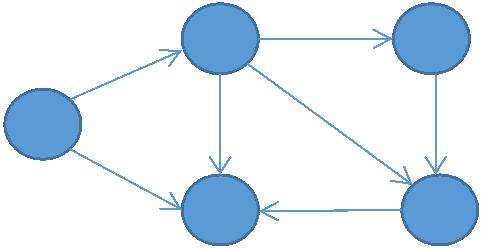
作者简介：周鑫（1977—），女，山东泰安人，中学高级教师，研究方向为计算机教育；张晶（1975—），女，山东聊城人，教授，研究方向

为计算机应用。

79

本栏目责任编辑：谢媛媛

软件技术



Computer Knowledge and Technology电脑知识与技术

第17卷第30期 (2021年10月)

作之后，得到的是起点“经过一条边”到达其余各点的最短路径

值。第二轮遍历图中所有边进行松弛操作之后，得到的是从起

点“至多经过两条边”到达其余各点的最短路径值。如果进行

n-1 轮的话，得到的就是起点“至多经过 n-1 条边”到达其余各

顶点的最短路径值。在一个含有n个顶点的图中，由于任意两

点之间的最短路径最多经过n-1条边，因此最多松弛n-1轮。

第三轮对所有边进行松弛以后，结果如下表所示：

必要的冗余计算，我们用一个队列来维护。初始时将起始点加

入队列，每次从队列中取出队首元素，并对所有与他相邻的点

进行松弛，并将松弛成功的顶点入队，直到队列为空时算法

结束。

针对本例题的具体实现过程如下：

首先建立起始点1到其余各点的最短路径表格，d[i]表示起

点1到i点的最短路径值。

顶点a

d[a]

1

0

2

-1

3

-3

4

2

5

4

1

0

2

3

∞

4

∞

5

∞

d[i]

∞

第四轮对所有边松弛以后，结果如下表所示：

首先源点1入队，当队列非空时：

队首元素1出队，对以1为起点的所有边的终点（2、3）进 行

顶点a

d[a]

1

0

2

-1

3

-3

4

2

5

4

松弛操作，此时路径表格状态为：

1

0

2

-1

3

5

4

∞

5

∞

从第三轮开始，对所有边进行松弛操作，发现没有顶点需

d[i]

要更新，此时便可以提前结束遍历，优化效率。最后表中数据

就是1号顶点到其余各点的最短路径值。

松弛以后，2和3两个顶点的最短路径估值变小，而这两个

根据以上分析本题完整代码如下：

点队列中都没有，因此入队。

#include<bits/stdc++.h>

队首元素2出队，对以2为起点的所有边的终点（3，4，5）依

次进行松弛操作，此时路径表格状态为：

const int maxd = 1e9;

using namespace std;

1

0

2

-1

3

-3

4

2

5

5

int main(){

d[i]

int s[100] , e[100] , t[100] , d[100] , n , m ;

此时，3，4，5三个顶点的最短路径估值变小，其中3这个点

已经在队列中，不用入队，4，5两个点入队。

队首元素3出队，但是以3为起点没有出边，因此此时无松

弛操作。

cin>>n>>m;

for(int i = 1 ; i <= m ; i ++)

cin>>s[i] >>e[i] >>t[i];

for(int i = 1 ; i <= n ; i ++)d[i] = maxd;

队首元素4出队，对以4为起点的所有边的终点（5）进行松

d[1] = 0;

弛，此时路径表格状态为：

while(1){

bool upd=false;

1

0

2

-1

3

-3

4

2

5

4

for(int j = 1 ; j <= m ; j ++){

d[i]

if(d[e[j]] > d[s[j]] + t[j]){

队首元素5出队，但是以5为起点没有出边，此时无松弛操

d[e[j]] = d[s[j]] + t[j];

作，并且队列已空，算法结束，表中数据便是顶点1到其余各点

upd=true;

的最短路径值。

完整代码如下：

#include<bits/stdc++.h>

using namespace std;

const int N=1e5+5;

int cnt,head[N],d[N],n,m,vis[N];

struct edge{

}

}

if(upd==false) break;

}

for(int i = 1 ; i <= n ; i ++) cout<<d[i]<<"";

return 0 ;

}

int v,nxt,w;

/\*

}a[N\*2];

5 7

void addedge(int u,int v,int w){

a[++cnt].v=v;

2 4 3

1 2 -1

1 3 5

5 3 4

2 3 -2

2 5 6

4 5 2

\*/

a[cnt].w=w;

a[cnt].nxt=head[u];

head[u]=cnt;

}

void spfa(int s){

queue<int>q;

memset(d,0x3f,sizeof(d));

d[s]=0;

3 Spfa算法题解

q.push(s);

Spfa 是 Bellman-Ford 算法的一种队列实现，为了减少了不

vis[s]=1;

80

软件技术

本栏目责任编辑：谢媛媛

