在讲解 Dijkstra 最短路径算法中展示数学之美*

乔海燕 周晓聪 李绿周

中山大学计算机学院，广州，510006

摘 要 从一个命题出发推导出 Dijkstra 最短路径算法，进一步用不变式和归纳法证明算法的正确性，从而说明数学命题对于算法设计以及算法正确性的重要性，展示数学之美，培养学生的数学审美情趣，激发学生探求数学知识的内驱力。最后，用 C++标准库表示数学概念集合和映射的数据结构实现算法，进一步展示数学概念在编写程序中的作用，鼓励学生用数学思维设计程序。

关键字 算法，命题，Dijkstra 最短路径算法，离散数学，数学审美

Demonstrate Beauty of Math while Teaching Dijkstra's Shortest Path Algorithm

Qiao Haiyan Zhou Xiaocong Li Lvzhou

School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, China
qiaohy@mail.sysu.edu.cn isszxc@mail.sysu.eud.cn lilvzh@mail.sysu.edu.cn

Abstract—Starting from a proposition, Dijkstra's shortest path algorithm is derived, its correctness is proved by an invariant and induction. These steps demonstrate the beauty of math, encourage students' interests to study math. We furthermore implement the algorithm by using set and map in C++ library to show how math concepts can be used in programming.

Keywords—algorithm, proposition, Dijkstra's shortest path algorithm, discrete mathematics, math beauty

1 引 言

离散数学是计算机类专业的核心基础理论课程，内容包括数理逻辑、集合论、图论、组合数学、数论、抽象代数多个主题，是计算机类专业许多核心课程，如数据结构、编译原理、数据库原理、人工智能等的先导课程。课程的核心目标是培养学生的离散建模能力[9]。

学生常常提出学习离散数学有什么用的问题。事实上，离散数学教材往往提供许多离散数学应用的例子。许多例子也来源于实际问题，例如使用命题逻辑等值运算化简算法（程序）条件语句中的条件，通过图论模型解决最短路径和工作匹配等问题。另一方面，学习的目的不应该仅仅停留在是否实用的层次。培养学生积极探索真理，追求美好的事物是教育工作者的共识。在学习数学课程时，我们应该让学生学会欣赏数学之美，感受数学之美，从而激发学生学习数学的内驱力。我们选择了著名的 Dijkstra 最短路径算法展示数学之美，部分原因是算法设计和算法正确性证明既是计算机相关专业的难点也是重点，因此更便于学生接受和理解。

多数离散数学教材[2,3,4]讲解Dijkstra最短路径算法时，通常先给出算法，然后给出算法正确性证明，但是，最重要的算法设计和证明中的思想没有充分展现。这种讲法不便学生掌握设计算法的思想，容易造成学生死记硬背，也不便于激发学生学习数学的兴趣。极少离散数学教材讲到这两个算法时说明算法背后的数学。例如，黄亚群等著《离散数学》[5]在讲到 Dijkstra 最短路径算法时提到了算法的基本思想来源于简单的数学命题，但没有讲解从命题到算法的清晰路线。

根据命题等同于类型的 Curry-Howard 同构[1]，一个命题的证明等同于相应类型的函数程序，这样的函数程序既包含了算法，又包含了算法满足命题所表达性质的证明，因此，由此抽取的算法和程序的正确性得以保证。

命题等同于类型原理揭示了数学证明和正确算法的联系。认识这种联系对于激发学生数学的积极性很有意义。虽然讲解 Curry-Howard 同构涉及更高深的知识，而且抽取出的程序是函数式的，不同于学生正

*基金资助：2021 中山大学本科教学质量工程项目。

在学习的命令式程序，但是，许多算法的设计及其正确性仍然可以从一个简单的命题出发，经过简单的计算和推导获得。经典的 Dijkstra 最短路径算法就是一个典型的例子。

2 Dijkstra 最短路径算法的设计

图论是数学建模的最直观最有效语言，它在离散数学课程中的内容多，并需要综合应用逻辑语言、集合语言和算法语言描述问题、模型和解。特别是，图论中的典型问题和典型算法既是离散数学在解决实际问题的典型应用，也是展示数学之美的极好例子。本节以著名的 Dijkstra 单源点最短路径算法为例展示数学之魅力。

假设简单带权图$G=(V,E,w)$，$w:E\to R^+$,其中R^+表示正实数集，即所有边上的权均大于 0。单源点最短路径问题如下：给定一个结点$s\in V$（源点），求源点s到其他各个结点的最短路径。Dijkstra 最短路径算法可以从一个基本事实出发，经过比较简单的推理得到解决问题的方法。

定理 1. 如果$P=(u_0,\cdots,u,v)$是u_0到v的最短路径，那么P的前缀$P'=(u_0,\cdots,u)$ 是u_0到u的最短路径，而且$d(u_0,v)=d(u_0,u)+w(u,v)$，其中$d(u,v)$是结点u到v的距离，即最短路径长度。

证明：假设$P=(u_0,\cdots,u,v)$ 是u_0到v最短路径，但P的前缀$P'=(u_0,\cdots,u)$不是u_0到u的最短路径，因此存在u_0到u比P'更短的路径$P''=(u_0,\cdots,u)$，因此，

$P''=(u_0,\cdots,u)$和边(u,v)构成u_0到v的路径，而且其长度小于P的长度，这与假设矛盾。证毕。

令$S\subset V$是已求得最短路径的结点集，$\overline{S}=V-S$，用$d(u_0,\overline{S})$表示u_0到集合$\overline{S}$的距离，则

$$d(u_0,\overline{S})=min_{\{u\in S\ ,v\in\overline{S}\}}\{d(u_0,u)+w(u,v)\}$$
$$=min_{\{v\in\overline{S}\}}min_{\{u\in S\}}\{d(u_0,u)+w(u,v)\}$$
$$=min_{\{v\in\overline{S}\}}D(v)$$

其中$D(v)=min_{\{u\in S\}}\{d(u_0,u)+w(u,v)\}$

这表明，如果对每个$v\in\overline{S}$标识$D(v)$，则可求得具有最小值的结点$u\in\overline{S}$，u成为下一个属于S的结点。

由此可以设计求单源点最短路径的 Dijkstra 最短路径算法如下：

(1) 令$S=\{u_0\}$，$\overline{S}=V-S$
(2) 对每个$v\in V$，令$D(v)=d(u_0,u_0)+w(u_0,v)=w(u_0,v)$
(3) 重复下列步骤直至$S=V$:
  (a) 设结点$u\in\overline{S}$是满足$D(u)=min_{\{v\in\overline{S}\}}D(v)$的结点
  (b) 令$S=S\cup\{u\}$，$\overline{S}=V-S$
  (c) 对于每个$v\in\overline{S}$ 如果$(u,v)\in E$，则令$D(v)=min\{D(v),d(u_0,u)+w(u,v)\}$

严格地说，以上的描述并不能称为算法。算法应该用更准确更规范的伪代码描述。我们将以上的自然语言描述称为方法。

至此，可以用伪代码表示 Dijkstra 最短路径算法（见图 1 算法 1）。对于具有一定程序设计基础的学生，容易将算法 1 转换为程序设计语言表示的代码。

需要强调的是，熟练使用伪代码描述算法是训练学生算法思维的重要内容。学生往往可以用一种方法解决一个问题实例，例如，给定一个具体的带权图，一个源点 s,按照算法求源点 s 到其他各个结点的最短路径。但是，将一种方法用计算机程序设计语言实现则往往无从下手。实际上，编写伪代码对学习离散数学甚至学习后续课程数据结构和算法的学生都具有极大的挑战性。我们的经验和建议是：将从解决问题的方法到编写程序代码的过程进一步细化为三步，即三步法[7]：

第一步：用自然语言比较条理地描述解决问题的方法。例如，以上对 Dijkstra 最短路径算法的自然语言描述(1)-(3)。

第二步：用规范的伪代码描述算法。例如，图 1 的 Dijkstra 最短路径算法伪代码。

第三步：用程序设计语言实现伪代码算法。例如，代码 1 是伪代码算法 1 的 C++实现。

事实证明，使用三步法可以有效降低从方法到代码的复杂性，减少代码中的错误，提高程序设计效率和程序质量[8]。

3 Dijkstra 最短路径算法的正确性

一个算法的正确性指：算法对于合法的输入总是能给出正确的输出。这种正确性命题可以用一阶逻辑语言给出形式化描述，从而为进一步进行算法的形式化证明和验证，以至对于后续程序代码的测试提供基础。限于篇幅，本节只给出 Dijkstra 算法正确性的数学证明。

算法的正确性通常用一个不变式说明。一个不变式是一个命题，该命题在算法的所有循环中总是保持为真，而且当算法结束时，该命题意味着算法的正确性。

定理 2. 在算法 1 中，每次循环 6-13 结束时：

(1) 对于任意$v \in S$，$D(v)$是从u_0到v的最短路径长度。

(2) 对于任意$v \in \overline{S}$， $D(v)$记录了从u_0到v中间通过S结点的最短路径长度。

因此，在语句 7 处$D(u)$是新求得的u_0到$u \in \overline{S}$的最短路径长度，特别是， 算法结束时，$D(v)$记录了u_0到各结点v的最短路径长度。

算法 1 Dijkstra(G,u_0)

输入： $G=(V,E,w)$ 是一个带权图,u_0是源点

输出： $D(v)$ 记录源点u_0到其他各结点v的最短路径长度

```
1: S ← {u0}
2: S̄ ← V − S
3: for 每个结点 v ∈ S do
4:     D(v) ← w(u0, v)
5: end for
6: while S ≠ V do
7:     设 D(u) = min_{v∈S̄} D(v)
8:     S ← S ∪ {u}
9:     S̄ ← V − S
10:    for 每个边 (u, v) ∈ E do
11:        D(v) ← min{D(v), D(u) + w(u, v)}
12:    end for
13: end while
14: 输出D
```

图 1 Dijkstra最短路径算法

证明： 对循环次数i用归纳法证明。

当$i=0$时，即第一次 while 循环(语句 6-13)之前，根据语句 1-5，$S=\{u_0\}$，对于任意$v \in V$，$D(v)=w(u_0,v)$，因此，$D(u_0)=0$是u_0到自身的最短路径长度，$D(v)(v \in \overline{S})$是u_0到v 但不经过其他结点的最短路径长度。

假设$i=k$时命题成立，在进入第$k+1$次循环时，如果$u \in \overline{S}$根据语句 7 求得的最小标记结点，那么$D(u)$是u_0到u的最短路径长度，否则，存在更短的u_0到u的路径$P=(u_0,\cdots,u_i,u)$。如果$u_i \in S$，则P是u_0经过S到达u的更短路径，这与归纳假设$D(u)$为u_0经过S到达u的最短路径长度矛盾，因此$u_i \in \overline{S}$。 因为$w(u_i,u)>0$，因此，P的前缀$P'=(u_0,\cdots,u_i)$的长度小于$D(u)$，$D(u_i)<D(u)$，这与假设$D(u)(u \in \overline{S})$为最小矛盾。因此，(1)成立，即$D(u)$ 是u_0到u的最短路径长度。

再证(2)成立。对于任意$v \in \overline{S}$，如果$(u,v) \in E$，则根据语句 10-12，$D(v)$是从u_0到v中间通过S结点的最短路径长度。如果$(u,v) \notin E$，那么根据归纳假设，$D(v)$是从u_0到v 中间通过S结点的最短路径长度。证毕。

归纳法是数学证明中最常见的证明，也是某种意义上最简单最漂亮的证明。掌握归纳法是学习数学证明最基本要求。算法的正确性得以用数学证明确认，这为进一步设计正确的程序代码奠定了基础。事实上，许多数学证明中特别是构造性证明中包含了解决问题的算法，而且可以从这个证明中自动抽取出算法，从而为设计正确程序提供了一种途径。[1]

注意，以上证明中使用了边权大于零的假设。在学习数学定理和证明中，学生应该养成质疑其中的条件是否必要的习惯。如果条件是必要的，那么应该给出反例。事实上，众所周知，Dijkstra 最短路径算法不适用于边权小于零的带权图。例如，$G=(V,E)$，$V=\{u,v,w\}$，$E=\{(u,v,2),(u,w,3),(w,v,-2)\}$，并设$u$为源点，如图 2 所示。按照 Dijkstra 最短路径算法，初始标记结点$D(u)=0,D(v)=2,D(w)=3$。 按照 Dijkstra 算法首次求得最小标记结点为v,根据定理 1,源点u到v的最短路径长度为 2，但是，(u,w,v)显然是更短的路径，其长度为 1。

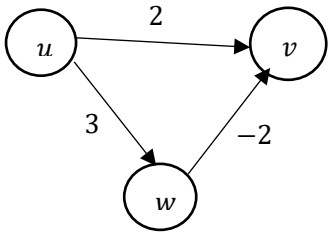

图 2 Dijkstra 算法不适用于负权图的反例

4 使用数学概念实现算法

对于具有初步程序设计基础的同学，完全可以将伪代码算法 1 转换为程序设计语言如 C++代码。如前所述，编写伪代码是编写 C++代码的基础。只有基于精确描述解决问题方法的伪代码，并且正确性得到证明，才有可能编写正确的程序设计语言代码。

在讲解该算法时，学生已经理解用邻接矩阵表示一个图。矩阵和矩阵的 C++表示如二维数组是学生已经比较熟悉的内容。

为了实现算法 1，只需要选择适当的数据结构表示算法中的集合 S、$\overline{S}$ 和D。学生往往选择数组表达这些数据。为了结合数学与代码之间的紧密联系，可以在程序中仍然把 S 和 $\overline{S}$视为集合，并选择用高阶数据结构 set 表示S和 $\overline{S}$（用变量 Sbar 表示）。另外，把D视为一个函数，并用 C++标准库的映射 map 表示D。注意到，这里能将D视为函数表现了学生对应函数概念的进一步认识。

基于以上数据结构的选项，由此得到 C++源代码（代码 1）。这里假定了用$0,1,\cdots,n-1$表示图的n个结

点，其类型用 Vertex 表示。图的邻接矩阵用 vector<vector<int>>表示，其别名为 Graph。

```
map<Vertex, int> dijkstra(Graph &G, Vertex u0){
//求带权图G中从源点u0到其他各点的最短路径
    set<Vertex> S, Sbar, V;//结点集合
    S.insert(u0);
    size_t n = G.size();
    for (size_t i=0;i<n;i++){
        V.insert(i);
        if (i!=u0)
            Sbar.insert(i);
    }
    map<Vertex,int> D;//用映射表示结点标记
    for (size_t v=0;v<n;v++)
        D[v] = G[u0][v];
    while (S != V){
        Vertex u = findMin(D, Sbar);
        S.insert(u);
        Sbar.erase(u);
        for (size_t v=0;v<n;v++){
            if (Sbar.find(v)!= Sbar.end()
                    && G[u][v])
                D[v] = min(D[v], D[u]+G[u][v]);
        }
    }
    return D;
}
```

代码 1 算法 1 的 C++实现

这里需要强调的是，集合和函数是离散数学最基本概念，能否熟练使用集合和函数表达数据和对象，是检验学生是否真正理解这些概念，并用离散数学语言建模的一个标准。事实上，学过一学期程序设计的学生往往没有意识到数学函数和 C++函数的区别，也不能自觉地在程序设计中使用集合和数学函数的概念。在学习离散数学期间，有意识地让学生熟悉集合和数学函数在如 C++标准库中的对应工具 set 和 map，对于提高学生对抽象数学概念集合和函数的进一步理解，习惯于使用数学语言思维，特别是让学生能够欣赏使用高层次概念和对应 C++高级工具对解决问题带来的简洁和便利，从而进一步激发学生对数学抽象之美的欣赏，激发做程序的兴趣。

5 结束语

离散数学是计算机相关专业的核心课程，让学生学好该课程，不仅应该从“有用”的角度讲解，而且要进一步从审美角度让学生感受数学之美，激发学生的兴趣和探求数学知识的内驱力。德智体美劳全面发展是党的教育方针。美育应该融入每个学科以及每个课程之中。

本文以 Dijkstra 最短路径算法为例，说明如何从一个简单命题推演出单源点最短路径问题的 Dijkstra 最短路径算法，进一步用归纳法证明算法的正确性，从中展示数学的魅力。

从伪代码到 C++代码使用了学生熟知的矩阵和集合以及 C++对应的工具，引导学生将数学概念和方法应用于程序设计中，降低了编写代码的难度。

从命题的分析到方法的细化，从细化的方法到伪代码，最后从伪代码到程序设计语言代码的三步法[7]给初学程序设计的学生展示了编写程序的一条途径，使得学生不畏惧程序设计的难度，并将数学概念应用于程序设计形成习惯。

最后，需要说明的是，从一个简单命题推演出单源点最短路径问题的 Dijkstra 最短路径算法并非本文创新，一些离散数学教材（如[5]）和数据结构教材（如[6]）采用了类似的讲解方法。本文的重点在于以 Dijkstra 最短路径算法为例，展示了从命题到代码的程序设计过程中数学的作用和魅力。

参 考 文 献

[1] Bengt Nordström, Kent Petersson, Jan M.Smith. Programming in Martin-Löf's Type Theory: An Introduction[M].Oxford University Press, 1990.

[2] Kenneth H. Rosen. Discrete Mathematics and Its Applications[M]. 7th ed. 机械工业出版社，2017.

[3] 刘芳. 离散数学及其应用[M]. 科学出版社，2018.

[4] 汤姆·詹金斯，本·斯蒂芬森. 计算机离散数学基础[M]. 机械工业出版社， 2020.

[5] 黄亚群，蒋慕蓉，赵春娜. 离散数学[M]. 科学出版社，2020.

[6] 裘宗燕. 数据结构与算法：Python 语言描述[M]. 机械工业出版社，2016.

[7] 乔海燕，周晓聪. Python 程序设计基础--程序设计三步法[M] 清华大学出版社，2022.

[8] 乔海燕 ,周晓聪,王若梅,万海. 提高在线评测代码提交接受率的三步法[J]. 软件导刊，2020（12）：138-140.

[9] 周晓聪,乔海燕,李绿周. 离散数学课程教学目标的细化与课程内容整合[J]. 教育教学论坛，2020（24）：261-264.