数与图的完美结合

——————

浅析差分约束系统

华中师大一附中　冯威

引言

- 在面对多种多样的问题时，我们经常会碰到这样的情况：往往我们能够根据题目题面意思来建立一些简单的模型，但却面对这些模型无从下手。这时我们应该意识到，也许能够将这种模型与其他的模型之间搭起一座桥梁，使我们能够用更简单直接的方式解决它。这里我们介绍一种方法，它很好地将某些特殊的不等式组与图相联结，让复杂的问题简单化，将难处理的问题用我们所熟知的方法去解决，它便是**差分约束系统**。这里我们着重介绍**差分约束系统**的原理和其需要掌握的 **bellman-ford** 算法。然后通过 zju1508 和 zju1420 两道题目解析差分约束系统在信息学题目中的应用，并逐渐归纳解决这类问题的思考方向。

Bellman ford 算法

- 算法简单介绍

这个算法能在更一般的情况下解决最短路的问题。

一般在：

1. 该算法下边的权值可以为负

2. 可以运用该算法求有向图的单元最长路径或者最短路径 .

3. …

Bellman ford 算法

- 松弛技术：

对每一个结点 v， 逐步减小从起点 s 到终点 v 最短路的估计量 dist[v] 直到其达到真正的最短路径值 mindist[v] 。

以单元最短路径为例这个操作就是保证

dist[v]<=dist[u]+w[u,v] ，

即　　if dist[v]>dist[u]+w[u,v]

then dist[v]=dist[u]+w[u,v]

如果是最长路径则是保证

dist[v]>=dist[u]+w[u,v]

Bellman ford 算法

- 伪代码如下：

For i=1 to |V|G||-1

Do for 每条边 (u， v)

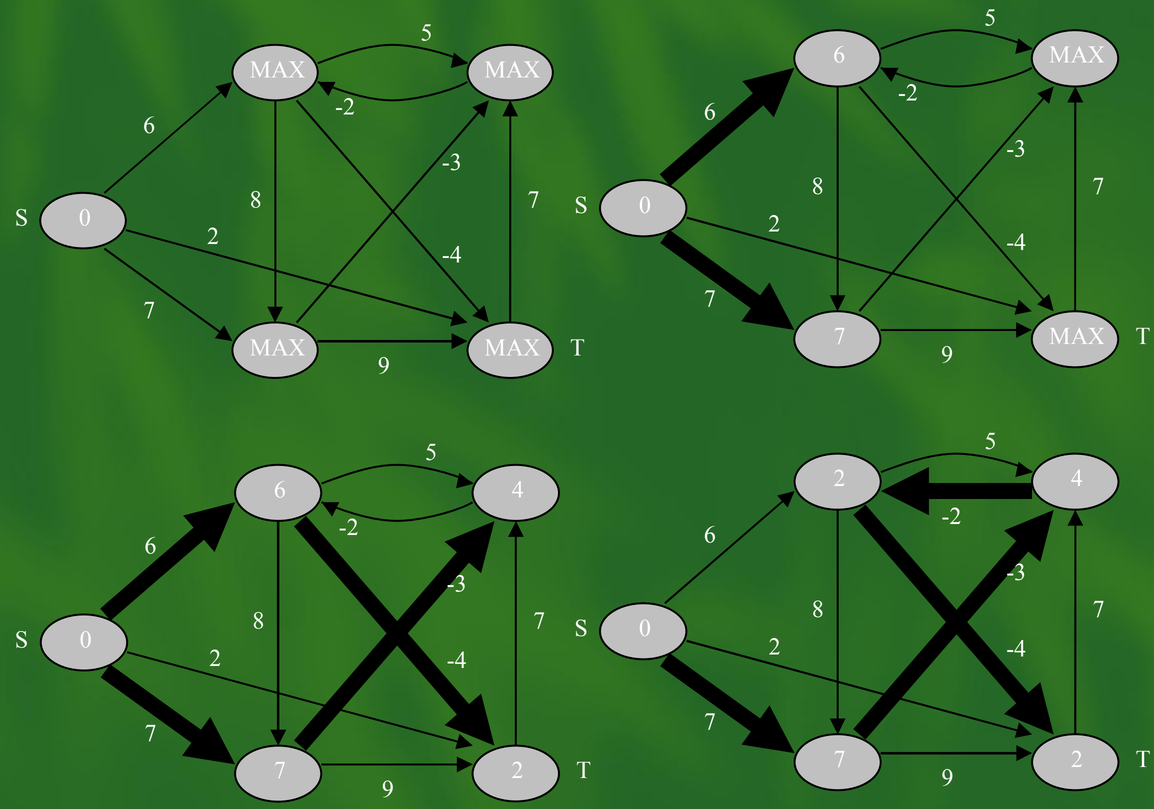
Do 更新操作（u， v， w （u， v））

For 每条边（u， v）

Do if 仍然有可更新内容 then return False

Return True

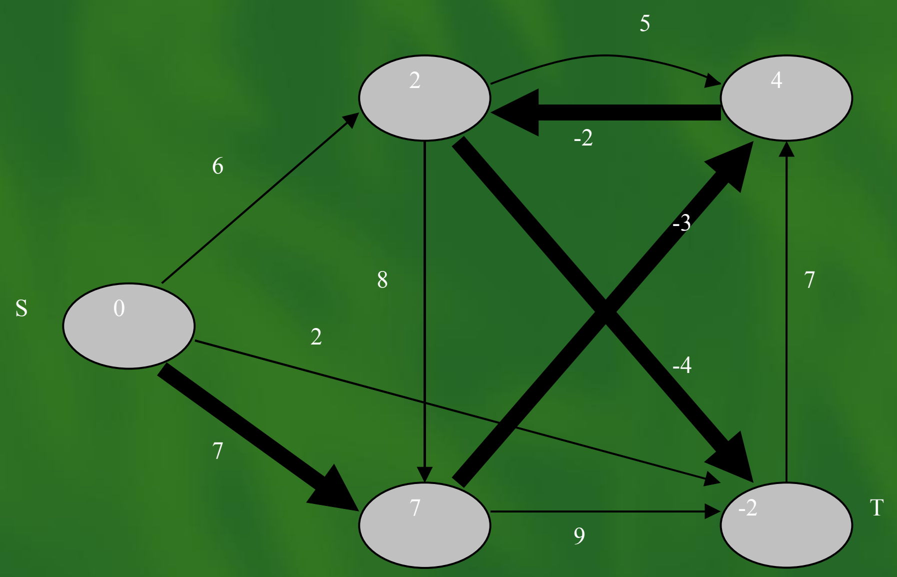
Bellman ford 算法



图例中， S 为源节点，粗线段覆盖的边表示最近一次执行更新操作的边。算法执行 |V|-1 次操作，每次操作都对所有可以进行松弛操作的边进行扩展。

Bellman ford 算法

- 最终状态如下图。



Bellman ford 算法

- 证明算法的正确性

  1．设 G=（V， E）为有向加权图，源节点为 S，加权函数为 w： E-〉 R。如果有负权回路则 Bellman-ford 算法一定会返回布尔值 false, 否则返回 TRUE 。

  2．设 G=（V， E）为有向加权图，源节点为 S 加权函数为 w： E-〉 R，并且 G 不含从 s 可达的负权回路，则算法 Bellman-ford 终止时，对所有从 s 可达的结点 v 有 d[v]=mindist （s，v）。

差分约束系统

- 对于解决差分约束系统问题的操作过程和使用原理，我们通过下面一道简单的题目进行了解。

- 引例： Zju1508 Interval

  题目大意：有一个序列，题目用 n 个整数组合 [a_i， b_i， c_i] 来描述它， [a_i， b_i， c_i] 表示在该序列中处于 [a_i， b_i] 这个区间的整数至少有 c_i 个。如果存在这样的序列，请求出满足题目要求的最短的序列长度是多少。如果不存在则输出 -1 。

差分约束系统

- 输入：第一行包括一个整数 n ，表示区间个数，以下 n 行每行描述这些区间，第 i+1 行三个整数 ai ， bi ， ci ，由空格隔开，其中 0<=ai<=bi<=50000 而且 1<=ci<=bi-ai+1 。
- 输出：一行，输出满足要求的序列的长度的最小值。

差分约束系统

- **将问题数字化：**
- 定义数组 T ，若数字 i 出现在序列中，则 Ti=1 ，否则 Ti=0 ，那么本题约束条件即为

$$\sum_{j=a_i}^{b_i} t_j \ge c_i \quad (i=1,2,\ldots,n)$$

- **建立不等式模型：**
- 这样的描述使一个约束条件所牵涉的变量太多，不妨设 $S_i=\sum_{j=1}^{i} t_j$ (i=1,2,…,n)

差分约束系统

- 约束条件即可用以下不等式表示

$$S_{b_i} - S_{a_i - 1} \geq c_i \ (i = 1, 2, \ldots, n)$$

- 值得注意的是，这样定义的 S 若仅仅满足约束条件的要求是不能完整体现它的意义的， S 中的各个组成之间并不是相对独立的，他们存在着联系。
- 由于 T 数组要么为 1 要么为 0 ，则 S_i 一定比 S_{i-1} 大，且至多大 1 于是有

$$S_i - S_{i-1} \geq 0 \ (i = 1, 2, \ldots, n)$$

$$S_{i-1} - S_i \geq -1 \ (i = 1, 2, \ldots, n)$$

差分约束系统

❖那么如何找到满足要求的这样一组 S，且使其个数最少呢?

联想

我们需要寻找一个满足以下要求的 S 数组

$$S_{b_i} - S_{a_i - 1} \geq c_i \ (i = 1, 2, \ldots, n)$$

$$S_i - S_{i-1} \geq 0 \ (i = 1, 2, \ldots, n)$$

$$S_{i-1} - S_i \geq -1 \ (i = 1, 2, \ldots, n)$$

$$d[u] - d[v] >= -w[u, v]$$

而 Bellman-Ford 每次的更新操作为

$$\text{If } d[u] + w[u, v] \leq d[v] \ \text{ then } d[v] \leftarrow d[u] + w[u, v]$$

$$d[v] <= d[u] + w[u, v]$$

差分约束系统

例

- INPUT

  3

  1 4 2

  3 6 3

  2 3 1

S_4-S_0>=2

S_6-S_2>=3

S_3-S_1>=1

- 于是做出如下的转化：

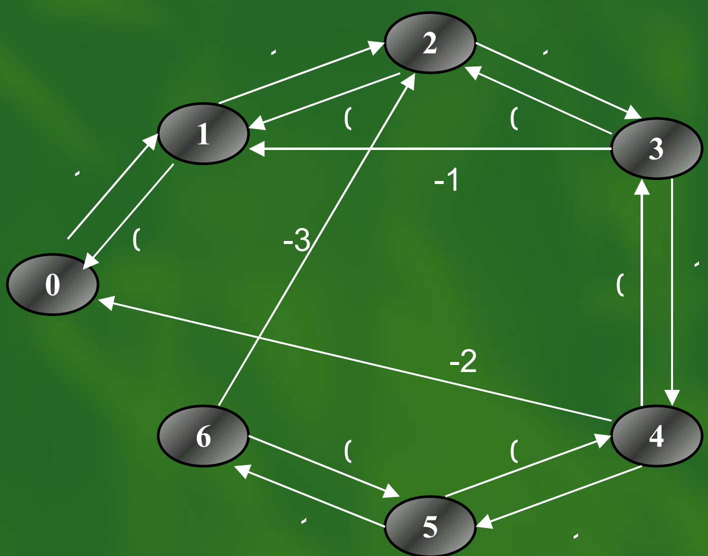
  1. 我们将 S_0， S_1。。。 S_n 看作 n+1 个点 V_0， V_1，。。， V_n

  2. 对诸如 A-B>=C 的形式，我们从 A 向 B 连一条有向边，权值为 -C 。

     1)S_{bi}-S_{ai-1}>=C_i

     2)S_i-S_{i-1}>=0

     3)S_{i-1}-S_i>=-1



这样图就能完整地描述本题了！

用 Bellman-ford 求解！！

差分约束系统

- 这样如果我们从 V_0 出发，求出结点 V_0 到 V_n 的最短路径，则 S_n-S_0 为满足要求情况下的最小值。相反如果我们发现在 Bellman-ford 算法执行的过程中存在有负权回路，则说明不存在满足要求的式子。
- 于是通过合理的建立数学模型，将不等式图形化，用 Bellmanford 作为武器，最终此题得到了圆满的解决!

差分约束系统

- **线性程序设计：**

　我们为线性程序设计问题制定一个严格的数学描述：

　给定一个 m*n 矩阵 A， 维向量 b 和维向量 c，我们希望找出由 n 个元素组成的向量 x，在由 Ax<=b 所给出的 m 个约束条件下，使目标函数 $\sum_{i=1}^{n} c_i x_i$ 达到最大值。

差分约束系统

- 其实很多问题都可以通过这样一个线性程序设计框架来进行描述。在实际的问题中，也经常要对其进行分析和解决。上述例题使我们对这一类线性程序设计问题提供了一个多项式时间的算法。它将一类特殊的线性不等式与图论紧密联系在了一起。这类特殊的线性不等式，我们称它为**差分约束系统**，它是可以用单元最短路径来求解的。

差分约束系统

- 差分约束系统：

　　差分约束系统是一个线性程序设计中特殊的一种，线性程序设计中矩阵 A 的每一行包含一个 1 和一个 -1 ， A 的所有其他元素均为 0 。由 Ax<=b 给出的约束条件形成 m 个差分约束的集合，其中包含 n 个未知单元。每个约束条件均可够成简单的不等式如下： $x_j-x_i<=b_k$ （ 1<=i,j<=n,1<=k<=m ）

差分约束系统

- 简单举例：

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \leq \begin{bmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{bmatrix}$$

差分约束系统

- 找出未知量 x_1,x_2,x_3,x_4,x_5, 并且满足以下差分约束条件：

$x_1-x_5<=-1$

$x_2-x_5<=1$

$x_3-x_1<=5$

$x_4-x_1<=-1$

$x_4-x_3<=-1$

$x_5-x_3<=-3$

$x_5-x_4<=-3$

需要注意的是，用Bellmanford 求出的具体答案只是众多答案中的一种，答案并不唯一，但答案之间却也有着联系。这里我们并不对其进行专门的探讨。

差分约束系统

- [例题三] zju1420 Cashier Employment 出纳员问题

- Tehran 的一家每天 24 小时营业的超市，需要一批出纳员来满足它的需要。超市经理雇佣你来帮他解决他的问题——超市在每天的不同时段需要不同数目的出纳员（例如：午夜时只需一小批，而下午则需要很多）来为顾客提供优质服务。他希望雇佣最少数目的出纳员。

  经理已经提供你一天的每一小时需要出纳员的最少数量—— R(0), R(1), ..., R(23) 。 R （ 0 ）表示从午夜到上午 1 ： 00 需要出纳员的最少数目， R （ 1 ）表示上午 1 ： 00 到 2 ： 00 之间需要的，等等。每一天，这些数据都是相同的。有 N 人申请这项工作，每个申请者 I 在没 24 小时中，从一个特定的时刻开始连续工作恰好 8 小时，定义 tI （ 0 <= tI <= 23 ）为上面提到的开始时刻。也就是说，如果第 I 个申请者被录取，他（她）将从 tI 时刻开始连续工作 8 小时。

  你将编写一个程序，输入 R （ I ）（ I = 0..23 ）和 tI （ I = 1..N ），它们都是非负整数，计算为满足上述限制需要雇佣的最少出纳员数目。在每一时刻可以有比对应的 R （ I ）更多的出纳员在工作。

差分约束系统

- 输入

  输入文件的第一行为测试点个数（ <= 20 ）。每组测试数据的第一行为 24 个整数表示 R （ 0 ）， R （ 1 ）， ... ， R （ 23 ）（ R （ I ） <= 1000 ）。接下来一行是 N ，表示申请者数目（ 0 <= N <= 1000 ），接下来每行包含一个整数 tI （ 0 <= tI <= 23 ）。两组测试数据之间没有空行。

  输出

  对于每个测试点，输出只有一行，包含一个整数，表示需要出纳员的最少数目。如果无解，你应当输出 “ No Solution” 。

差分约束系统

- 我们按刚才所讲到的方法对此题进行处理。这题很容易想到如下的不等式模型：

  设 num[i] 为 i 时刻能够开始工作的人数，x[i] 为实际雇佣的人数，那么 x[I] <=num[I] 。
  设 r[i] 为 i 时刻至少需要工作的人数，于是有如下关系：

差分约束系统

$x[I-7]+x[I-6]+\ldots+x[I]>=r[I]$

设 $s[I]=x[1]+x[2]\ldots+x[I]$，

得到

$0<=s[I]-s[I-1]<=num[I]$，　$0<=I<=23$

$s[I]-s[I-8]>=r[I]$，　$8<=I<=23$

$s[23]+s[I]-s[I+16]>=r[I]$，　$0<=I<=7$

差分约束系统

- 对于以上的几组不等式，我们采用一种非常笨拙的办法处理这一系列的不等式(其实也是让零乱的式子变得更加整齐、易于处理)。首先我们要明白差分约束的应用对象（它通常针对多个二项相减的不等式的）于是我们将上面的所有式子都化成两项未知项在左边，另外的常数项在右边，切中间用 >= 连接的式子

差分约束系统

- s[I]-s[I-1]>=0　　　　　(0<=I<=23)
  s[I-1]-s[I]>=-num[I]　　　(0<=I<=23)
  s[I]-s[I-8]>=r[I]　　　　(8<=I<=23)
  s[I]-s[I+16]>=r[I]-s[23]　(0<=I<= 7)

这里出现了小的困难，我们发现以上式子并不是标准的差分约束系统，因为在最后一个式子中出现了三个未知单位。但是注意到其中跟随I 变化的只有两个，于是 s[23] 就变得特殊起来，看来是需要我们单独处理，于是我们把 s[23] 当作已知量放在右边

差分约束系统

- 经过这样的整理，整个图就很容易创建了，将所有形如 A-B>=C 的式子 我们从节点 B 引出一条有向边指向 A 边的权值为 C （这里注意由于左右确定，式子又是统一的 >= 的不等式，所以 A 和 B 是相对确定的，边是一定是指向 A 的），图就建成了 。

  最后枚举所有 s[23] 的可能值，对于每一个 s[23]，我们都进行一次常规差分约束系统问题的求解，判断这种情况是否可行，如果可行求出需要的最优值，记录到 Ans 中，最后的 Ans 的值即为所求。

总结

- 对于许多复杂的问题，我们通常选择将不够清晰、难以处理的模型转化为容易理解、易于处理的模型。就像用已知的知识作为工具去探索未知领域一样，联想、发散、转化将成为相当有用的武器。本文选择了差分约束系统这样一个平台，通过介绍差分约束系统的相关知识和其在信息学问题中的应用以小见大，为读者提供一个解题的思路和技巧。

❖谢谢