算法设计与分析

第3讲 问题形式化

汪小林

北京大学 信息科学技术学院

主要内容

- 几个著名问题的形式化描述
  - 稳定匹配问题
  - 区间调度问题
  - 带权重的区间调度问题
  - 二分图匹配问题
- 通过例题讲解如何对具体问题进行形式化描述
  - **1034:The dog task**
  - **1195:Mobile phones**
  - **1193**:内存分配
- 作业和小结

稳定匹配问题的背景

- 2012年的诺贝尔经济学奖：夏普利（L. S. Shapley）和罗斯（A. E. Roth），表彰他们在稳定配对和市场设计方面的理论和实践并重的贡献。
- 在医学领域，学生通常在后几年学习生涯中需要去医院实习。1940年代，美国的医院系统开始大规模发展，但医学院学生的数量很少，医院之间的竞争导致对医学院学生需求的急剧增加，于是很多医院就让学生提前实习，甚至在这些学生还没有选定专业领域的情况下就参加实习。但如果学生拒绝一个医院，往往导致医院再去找第二个学生就太迟了，因为第二个人可能已经被另一个医院抢走了。市场在这种情况下是极为不稳定的，因为医院往往会设定一个最后申请期限，迫使学生在不晓得是否还有其他机会之前就做出选择。由于医院未能及时给所有学生提供机会，而学生也未能向所有医院提出及时申请，双方都未能极大化自己的利益。

形式化：稳定匹配问题

- 男生集合 $M=\{m_1, m_2, \ldots, m_n\}$ 和女生集合 $W=\{w_1, w_2, \ldots, w_n\}$，以及男女生的所有有序对 (m, w) 构成的集合 $M*W$，其中 $m \in M$，$w \in W$ 。称由 $M*W$ 中的一组有序对构成的集合 S 为一个匹配，如果 M 和 W 中的每个元素最多只出现在 S 的一个有序对中。如果 M 和 W 中的每个元素恰好都只出现在 S 的某一个有序对中，则称 S 为一个完美匹配。

形式化：稳定匹配问题（续1）

- 男女生间存在者偏好关系。每个男生 $m \in M$ 都对所有的女生打分，如果 m 给 w 的打分高于 w'，则称 m 相对于 w' 而言更喜欢 w。称基于男生 m 的打分对女生排序得到的是 m 的偏好列表。偏好列表中的得分不能相同。同样的，每个女生也给所有的男生打分。
- 在匹配 S 中，如果存在两个有序对 (m, w)和(m', w')，其中 m 相对于 w 更喜欢 w'，而 w' 相对于 m' 更喜欢 m。则称有序对 (m, w') 为相对于 S 的一个不稳定因素。很显然，(m, w') 不属于 S。

形式化：稳定匹配问题（续2）

- 问题的目标是找到男女生间不存在不稳定因素的婚姻关系。
- 称匹配 S 是稳定的，如果 S 是完美的，并且不存在相对于 S 的不稳定因素。
- 两个问题：
  - 对于所有可能的偏好列表，是否都存在稳定匹配？
  - 对于给定的一组偏好列表，如果稳定匹配存在，是否可以把某个稳定匹配高效地构造出来？

区间调度问题

- 存在某种资源，例如教室、计算机集群、电子显微镜等等。许多人都希望某段时间来使用这个资源，但资源只能在每段时间内分配给唯一的使用者。当很多人都来申请时，如何能够满足最多的人利用这个资源？

区间调度问题的形式化

- 有 n 个请求，分别编号为 $1, 2, \ldots, n$ 。每个请求 i 都有一个开始时间 s_i 和一个结束时间 f_i 。并且对于所有的 i ，自然的要求 $s_i < f_i$ 成立。如果两个请求 i 和 j 的时间段不重叠，则称 i 与 j 是相容的，即，请求 i 的时间段或者早于请求 j （ $f_i \leq s_j$ ），或者晚于请求 j （ $f_j \leq s_i$ ）。
- 对于请求的子集 A ，如果任意的 $i, j \in A$，$i \neq j$ 都是相容的，则称 A 是相容请求子集。
- 目标：求最大相容请求子集。

带权重的区间调度问题

- 把区间权重问题更一般化
- 为每个请求区间 i 关联一个数值（或权重） $v_i > 0$，可以表示为我们从满足并安排请求 i 中获得的收益。
- 目标：求总收益最大的相容请求子集。

二分图匹配问题

- 在稳定匹配问题中引入了匹配和完美匹配的概念。
- 匹配是男女生的有序对构成的集合，满足每个男生和每个女生都最多只属于其中的一个有序对。
- 完美匹配是包括了每个男生和每个女生的匹配。
- 如何用图论中的概念定义更通用的匹配的概念。

二分图匹配问题的形式化

- 二分图定义：设 $G = (V, E)$ 是一个无向图，如果顶点 V 可分割为两个互不相交的子集 A 和 B，并且图中的每条边 (i, j) 所关联的两个顶点 i 和 j 分别属于这两个不同的顶点集（$i \in A, j \in B$），则称图 G 为一个二分图。
- 在二分图 $G = (V, E)$ 中，如果边的子集 $M \subseteq E$ 满足每个顶点都最多只出现在 M 的一条边中，则称 M 是一个匹配。
- 如果 G 每个顶点都恰好只出现在匹配 M 的一条边中，则称 M 为完美匹配。
- 二分图匹配问题：给定任意的二分图 G，求最大匹配（包含边最多的匹配）。如果 $|A| = |B| = n$，则当且仅当最大匹配包含 n 条边时，它是一个完美匹配。

例题：1034:The dog task

- http://qwsfsx.openjudge.cn/level4/1034/
- Hunter Bob often walks with his dog Ralph. Bob walks with a constant speed and his route is a polygonal line (possibly self-intersecting) whose vertices are specified by N pairs of integers (Xi, Yi) ? their Cartesian coordinates.
- Ralph walks on his own way but always meets his master at the specified N points. The dog starts his journey simultaneously with Bob at the point (X1, Y1) and finishes it also simultaneously with Bob at the point (XN, YN).

例题：1034:The dog task (cont.)

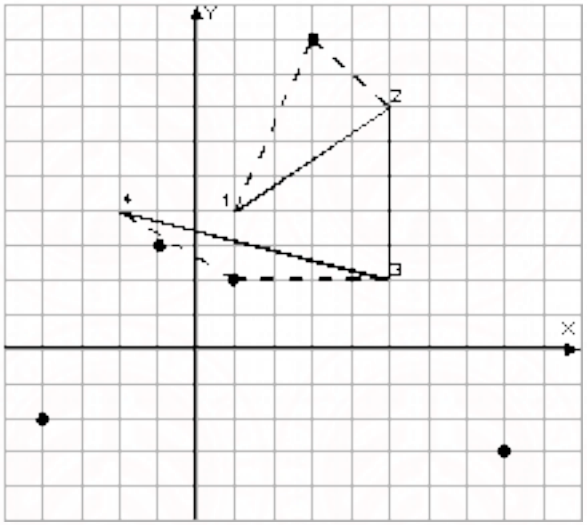
- Ralph can travel at a speed that is up to two times greater than his master's speed. While Bob travels in a straight line from one point to another the cheerful dog seeks trees, bushes, hummocks and all other kinds of interesting places of the local landscape which are specified by M pairs of integers (Xj',Yj'). However, after leaving his master at the point (Xi, Yi) (where 1 <= i < N) the dog visits at most one interesting place before meeting his master again at the point (Xi+1, Yi+1).

例题：1034:The dog task (cont.)

- Your task is to find the dog's route, which meets the above requirements and allows him to visit the maximal possible number of interesting places. The answer should be presented as a polygonal line that represents Ralph's route. The vertices of this route should be all points (Xi, Yi) and the maximal number of interesting places (Xj',Yj'). The latter should be visited (i.e. listed in the route description) at most once.

例题：1034:The dog task (cont.)

- An example of Bob's route (solid line), a set of interesting places (dots) and one of the best Ralph's routes (dotted line) are presented in the following picture:



直观的形式化描述

已知平面上的一个点序列 $\boldsymbol{P}$ 和一个点集 $\boldsymbol{Q}$，其中 $\boldsymbol{P}=\{\boldsymbol{p_1},\boldsymbol{p_2},\dots,\boldsymbol{p_n}\},\boldsymbol{Q}=\{\boldsymbol{q_1},\boldsymbol{q_2},\dots,\boldsymbol{q_m}\},\boldsymbol{m},\boldsymbol{n}\in\boldsymbol{N},\boldsymbol{m}\geq\boldsymbol{1},\boldsymbol{n}\geq\boldsymbol{2}$。求长度最大的点序列$\boldsymbol{V}=\{\boldsymbol{v_1},\boldsymbol{v_2},\dots,\boldsymbol{v_k}\}$，其中$\boldsymbol{n}\leq\boldsymbol{k}\leq\boldsymbol{n}+\boldsymbol{m}$，并满足下列条件：

1）序列 $\boldsymbol{V}$ 的起点与终点与序列 $\boldsymbol{P}$ 相同：即$\boldsymbol{v_1}=\boldsymbol{p_1}$，$\boldsymbol{v_k}=\boldsymbol{p_n}$；

2）序列 $\boldsymbol{V}$ 是利用 $\boldsymbol{Q}$ 中的点对序列 $\boldsymbol{P}$ 的一个扩充，且 $\boldsymbol{Q}$ 中的每个点在 $\boldsymbol{V}$ 中只能出现一次：即对任意的$\boldsymbol{1}\leq\boldsymbol{i}<\boldsymbol{j}\leq\boldsymbol{k}$，

$$\{\boldsymbol{v_i},\boldsymbol{v_j}\}=\{\boldsymbol{p_s},\boldsymbol{p_t}\}\subseteq\boldsymbol{P}\Rightarrow\boldsymbol{s}<\boldsymbol{t},$$

$$\{\boldsymbol{v_i},\boldsymbol{v_j}\}=\{\boldsymbol{q_s},\boldsymbol{q_t}\}\subseteq\boldsymbol{Q}\Rightarrow\boldsymbol{s}\neq\boldsymbol{t},$$

$$\boldsymbol{V}-\boldsymbol{Q}=\boldsymbol{P};$$

直观的形式化描述（续）

3）在 $\boldsymbol{P}$ 中相邻的两点（$\boldsymbol{p_i, p_{i+1} \in P}$）之间在 $\boldsymbol{V}$ 中最多只能插入一个来自 $\boldsymbol{Q}$ 的点（$\boldsymbol{q \in Q}$），且插入的点应满足$\boldsymbol{q}$到$\boldsymbol{p_i}$和$\boldsymbol{p_{i+1}}$的距离之和不超过$\boldsymbol{p_i}$和$\boldsymbol{p_{i+1}}$间距离的两倍。即对任意的 $\boldsymbol{1 < j < k}$，$\boldsymbol{v_j \in V}$，

$$\boldsymbol{v_j \in Q \Rightarrow \begin{cases} \{v_{j-1}, v_{j+1}\} \subseteq P \\ d(v_{j-1}, v_j) + d(v_j, v_{j+1}) \leq 2d(v_{j-1}, v_{j+1}) \end{cases}}$$

其中函数 $\boldsymbol{d}$ 为平面上两点间的欧几里得距离，即如果两点 $\boldsymbol{\{v_i, v_j\} = \{(x_i, y_i), (x_j, y_j)\}}$，则定义

$$\boldsymbol{d(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$$

转化为二分图匹配问题

已知平面上的一个点序列 P 和一个点集 Q，其中 $P=\{p_1,p_2,\dots,p_n\}, Q=\{q_1,q_2,\dots,q_m\}, m,n\in N, m\geq 1, n\geq 2$。定义二分图 $G=\langle L\cup Q, E\rangle$，其中 $L=\{l_i=(p_i,p_{i+1})|0\leq i<n, \{p_i,p_{i+1}\}\subseteq P\}$，$E=\{(l,q)\mid d(l,q)\leq 2d(l),\ l\in L, q\in Q\}$，求二分图 G 的最大匹配。

其中将欧几里得距离函数 d 扩展定义如下：设 $l=l_i=(p_i,p_{i+1})\in L$，$q=q_j\in Q$，其中 $0\leq i<n, 0\leq j\leq m$，则

$$d(l,q)=d(p_i,q_j)+d(q_j,p_{i+1})$$

$$d(l)=d(p_i,p_{i+1})$$

例题：1195:Mobile phones

- http://qwsfsx.openjudge.cn/level4/1195/
- Suppose that the fourth generation mobile phone base stations in the Tampere area operate as follows. The area is divided into squares. The squares form an S * S matrix with the rows and columns numbered from 0 to S-1. Each square contains a base station. The number of active mobile phones inside a square can change because a phone is moved from a square to another or a phone is switched on or off. At times, each base station reports the change in the number of active phones to the main base station along with the row and the column of the matrix.

例题：1195:Mobile phones (cont.)

- Write a program, which receives these reports and answers queries about the current total number of active mobile phones in any rectangle-shaped area.
- The input is read from standard input as integers and the answers to the queries are written to standard output as integers. The input is encoded as follows. Each input comes on a separate line, and consists of one instruction integer and a number of parameter integers according to the following table.

例题：1195:Mobile phones (cont.)

Instruction	Parameters	Meaning
0	S	Initialize the matrix size to S * S containing all zeros. This instruction is given only once and it will be the first instruction.
1	X Y A	Add A to the number of active phones in table square (X, Y). A may be positive or negative.
2	L B R T	Query the current sum of numbers of active mobile phones in squares (X, Y), where L <= X <= R, B <= Y <= T
3		Terminate program. This instruction is given only once and it will be the last instruction.

例题：1195:Mobile phones (cont.)

- The values will always be in range, so there is no need to check them. In particular, if A is negative, it can be assumed that it will not reduce the square value below zero. The indexing starts at 0, e.g. for a table of size 4 * 4, we have 0 <= X <= 3 and 0 <= Y <= 3.
- Table size: 1 * 1 <= S * S <= 1024 * 1024
- Cell value V at any time: 0 <= V <= 32767
- Update amount: -32768 <= A <= 32767
- No of instructions in input: 3 <= U <= 60002
- Maximum number of phones in the whole table: M= 2^30

问题形式化：动态子矩阵求和问题

已知方阵$A_s = \{a_{ij}: 0 \le i,j < s < 1024\}$，其中每个元素$a_{ij}$的值在整个过程中满足$0 \le a_{ij} \le 2^{16}-1$，且方阵$A_s$中所有元素的和$\sum_{0 \le i,j \le n} a_{ij} \le 2^{30}$。

方阵A_s的子矩阵定义为：$A_s(l,b,r,t) = \{a_{ij}: 0 \le l \le i \le r \le s; 0 \le b \le j \le t \le s\}$。

在每两个时刻(t_{k-1}, t_k)之间，可能有多个元素$\{a_{xy}\}$的值会发生增减，但增减后的值不会超出元素的取值范围。

现给定时刻序列$\{t_k: k = 1,2,\dots\}$，求在每个t_k时刻，指定的子矩阵$A_s(l_k,b_k,r_k,t_k)$中所有元素的和。

例题：1193:内存分配

内存是计算机重要的资源之一，程序运行的过程中必须对内存进行分配。

经典的内存分配过程是这样进行的：

1. 内存以内存单元为基本单位，每个内存单元用一个固定的整数作为标识，称为地址。地址从0开始连续排列，地址相邻的内存单元被认为是逻辑上连续的。我们把从地址 i 开始的 s 个连续的内存单元称为首地址为 i 长度为 s 的地址片。

2. 运行过程中有若干进程需要占用内存，对于每个进程有一个申请时刻T，需要内存单元数M及运行时间P。在运行时间P内（即T时刻开始，$T+P$时刻结束），这M个被占用的内存单元不能再被其他进程使用。

例题：1193:内存分配(续)

3.假设在T时刻有一个进程申请M个单元且运行时间为P，则：

- 1. 若T时刻内存中存在长度为M的空闲地址片，则系统将这M个空闲单元分配给该进程。若存在多个长度为M个空闲地址片，则系统将首地址最小的那个空闲地址片分配给该进程。
- 2. 如果T时刻不存在长度为M的空闲地址片，则该进程被放入一个等待队列。对于处于等待队列队头的进程，只要在任一时刻，存在长度为M的空闲地址片，系统马上将该进程取出队列，并为它分配内存单元。注意，在进行内存分配处理过程中，处于等待队列队头的进程的处理优先级最高，队列中的其它进程不能先于队头进程被处理。

例题：1193:内存分配(续)

现在给出一系列描述进程的数据，请编写一程序模拟系统分配内存的过程。

输入

- 第一行是一个数N，表示总内存单元数（即地址范围从0到$N-1$）。从第二行开始每行描述一个进程的三个整数T、M、P（$M <= N$）。最后一行用三个0表示结束。
- 数据已按T从小到大排序。
- 输入文件最多10000行，且所有数据都小于10^9。
- 输入文件中同一行相邻两项之间用一个或多个空格隔开。

输出

- 包括2行。第一行是全部进程都运行完毕的时刻。第二行是被放入过等待队列的进程总数。

内存分配问题的形式化描述

- 系统中的通过 **alloc(*m*)** 和 **free(*a*, *m*)** 函数维护的内存空闲块列表 **freelist**。初始时刻，**freelist** 中有唯一的内存空闲块 **(0, *N*)**。
- 调用 **alloc(*m*)** 时，在所有空闲块大小不小于 ***m*** 的空闲块中，返回起始地址最小的空闲块起始地址。如果该空闲块大小大于 ***m***，则更新该空闲块起始地址和大小，否则删除该空闲块。如果所有空闲块大小均小于 ***m***，则返回**-1**。
- 调用 **free(*a*, *m*)** 时，加入新的空闲块 **(*a*, *m*)**，并尝试将它与相邻的空闲块合并。如果能合并，删除被合并的空闲块元素。

内存分配问题的形式化描述（续）

- 任务队列 **tlist** 中的进程由三元组 (t_i, m_i, p_i) 描述（$1 \le i \le T$），其中对于$i < j$ 有 $t_i < t_j$，$m_i < N$。在时刻 t_i 调度进程 i 时，调用**alloc(m_i)** 尝试为进程 i 分配内存。
- 如果成功，记返回地址为 a_i，将三元组 (t_i+p_i, m_i, a_i) 加入当前进程队列 **clist**；
- 如果失败，将三元组 (t_i, m_i, p_i) 加入等待进程队列 **wlist**。

内存分配问题的形式化描述（续）

- ✧ 在任意时刻 t，如果 clist 中有进程 i 恰好结束，则调用 free(a_i, m_i) 释放内存，并从 clist 中移除进程 i （如果此时 clist , tlist 和 wlist 均为空，则输出当前时间 t 为最终输出）；如果 clist 中有多个进程同时结束，则应同时调用 free 释放内存，并移除进程。
- 有内存释放后，依次尝试调度 wlist 中的进程。如果队首进程的分配内存请求被满足，则将进程三元组 ($t+p_i$, m_i, a_i) 加入 clist ，接着在尝试下一个进程。否则结束尝试。
- ✧ 如果 tlist 存在其开始时刻 t_i 等于 t 的进程，则接着按在队列中的顺序依次调度这些进程。

小结

- 形式化问题描述是求解问题的第一步
- 两种形式化问题方式
  - 直接形式化：严格定义问题相关各个方面
  - 问题抽象：将问题转化为一个通用问题
- 将问题用映射或规约转化为标准的常见问题
- 通过严格的形式化问题描述发现问题中的毛病