DOI:10.14132/j.cnki.1673-5439.2012.06.017

第32卷 第6期 南京邮电大学学报(自然科学版) Vol. 32 No. 6
2012年12月 Journal of Nanjing University of Posts and Telecommunications(Natural Science) Dec. 2012

求解子集和问题的快速算法

王 蔚,邱伟星
(南京邮电大学 计算机学院,江苏 南京 210023)

摘 要: 针对子集和问题,文中提出了一种快速算法。该算法设计运用了整数带余除法和生日问题的原理。理论分析表明该算法时间复杂度为 $O(n^2)$,其正确率为 $1-\left(\frac{T-2}{T-1}\right)^{n^2m}$。随机试验显示,该算法在时间效率上明显优于传统指数时间复杂度算法,且对大集合问题具有很高的正确率。

关键词: 子集和问题;背包问题;整数除法;生日问题
中图分类号: TP312.8 **文献标识码:** A **文章编号:** 1673-5439(2012)06-0092-04

A Quick Algorithm for the Subset Sum Problem

WANG Wei, QIU Wei-xing
(College of Computer Science & Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: In this paper we present a quick algorithm for the subset sum problem by using principles of integer division with a remainder and birthday problem. We theoretically prove that the algorithm has a time complexity of $O(n^2)$, and its success rate is $1-\left(\frac{T-2}{T-1}\right)^{n^2m}$. Experimental results show that our approach has much lower time consumption than the traditional exponential-time algorithm and has a very high success rate for the problem samples of large sets.

Key words: subset sum problem; knapsack problem; integer division; birthday problem

0 引 言

子集和问题(the subset sum problem)可描述为:给定一正整数集合 $A=\{a_i \mid a_i \in Z_+, 1\leqslant i\leqslant n\}$ 和正整数 $T\in Z_+$,问是否存在非空子集合 $A_1\subseteq A$,使得 A_1 元素之和 $\sum_{a_j\in A_1} a_j = T$? 子集和问题是背包问题的一个特例,是NP完全问题[1],直接枚举求解时,最坏情况下时间复杂性为 $O(2^n)$,这使该问题成为计算复杂度理论和密码学中的一个重要问题[2-3]。如何有效地求解该问题,降低其计算复杂度,具有重要的理论和实际意义[4]。

针对子集和问题的解法有多种,其中,分枝限界算法对于某些实例表现了较好的时间性能,但其最坏情形的时间复杂性仍为 $O(2^n)$。近似算法(approximate algorithm)虽具有多项式时间复杂度[1],但牺牲了精确解从而使原始问题变成了另一个问题。较著名的二表算法(two-list algorithm)[5]将时间复杂度降为 $O(n2^{n/2})$,其后又出现了动态二表算法的改进版本[6]将时间复杂度进一步降为 $O(2^{n/2})$。

在数论整除理论中,整数的带余除法(integer division with a remainder)[7]是一个基本定理,其可表述为:若 a,b 是两个整数,其中 $b>0$,则存在一对整数 q 和 r,使得 $a=bq+r$,$0\leqslant r<b$ 成立,且上述 q 和 r 是唯一的。我们运用整数带余除法的原理,借鉴生日攻击的思想,提出了求解子集和问题的一种快速算法[8-10]。本文证明了该算法时间复杂度为 $O(n^2)$,且随着集合元素个数 n 增大,算法的正确率

收稿日期: 2011-12-20; **修回日期:** 2012-01-20
通讯作者: 王 蔚 **电话:** 13675131089 E-mail: weiwangnjupt2010@gmail.com

$1-\left(\frac{T-2}{T-1}\right)^{n^2m}$ 快速逼近 100%。

本文首先介绍了传统的指数时间复杂度算法，描述了整数带余除法快速算法，然后分析该算法的时间复杂度和正确率，并设计随机试验比较上述两种算法，最后总结了算法的特点及其实用价值。

1　指数时间复杂度算法

算法 1 描述了传统的指数时间复杂度算法[11]。初始集合 $L_0=\{0\}$。历经共 n 次迭代，每次迭代分别得到集合 $L_1,L_2,\cdots,L_n$。其中第 i 次迭代是将集合 A 元素 a_i 与上一次迭代所得集合 L_{i-1} 的每个元素分别求和得到集合 $L_{i-1}+a_i$，其与集合 L_{i-1} 合并得 L_i。在第 i 次迭代，内层循环体执行次数取决于 L_i 的长度 $|L_i|$，而 $|L_i|\leqslant 2|L_{i-1}|\leqslant 2^2|L_{i-2}|\cdots\leqslant 2^i|L_0|=2^i$。因此，算法的时间复杂度为 $O(2^n)$。

算法 1　EXACT-SUBSET-SUM (A,T)

输入: 非空集合 $A=\{a_i\mid a_i\in Z_+,1\leqslant i\leqslant n\}$，正整数 $T\in Z_+$

输出: 返回最大值 z^*，满足条件:

非空子集 $A_1\subseteq A$ 且 $\sum_{a_j\in A_1}a_j=z^*\leqslant T$

1. $n\leftarrow|A|$
2. $L_0\leftarrow\langle 0\rangle$
3. for $i\leftarrow 1$ to n
4. do $L_i\leftarrow$MERGE-LISTS$(L_{i-1},L_{i-1}+a_i)$
5. 移除 L_i 中大于 T 的元素
6. return L_n 之最大元素

2　快速算法

整数带余除法快速算法(quick algorithm using integer division with a remainder，QAIDR)的求解思路为: 设法取得集合 A 的某个子集 A_1，满足 $\sum_{a_j\in A_1}a_j=T+r$，其中 $0\leqslant r<T$。假如无法找到这样的 A_1，则原问题无解; 如找到 A_1 使得 $r=0$，则 A_1 便是原问题的解; 若 $r\neq 0$，则原问题就变成: 如何在 A_1 中去找若干个元素和等于 r? 故而原问题的规模随之缩小。

2.1　算法描述

记 $P_K=\sum_{i=1}^{K}a_i$，$1\leqslant K\leqslant n$。整个算法由两个阶段组成。首先执行第 1 阶段，如能求解问题则算法结束，否则需执行第 2 阶段以得出最终结果。第 1 阶段首先执行预处理，然后执行后续步骤。各步骤分别描述如下:

预处理步骤　首先将 A 中元素按顺序存入集合 B 中。接着将 A 中元素按照升序排列，使得 $a_1<a_2<\cdots<a_n$。

步骤 1　若 A 为空，本阶段结束。

步骤 2　依次计算 $P_1,P_2,\cdots,P_n$，直到出现某个 $S(1\leqslant S\leqslant n)$ 满足 $T\leqslant P_S<2T$。根据计算结果，分成以下 3 种情形分别加以处理:

(1) 未找到满足条件的 P_S，则判定无解，算法结束;

(2) 找到 $P_S=T$，则判定解为 $\{a_1,a_2,\cdots,a_S\}$，算法结束;

(3) 找到 $T<P_S<2T$，令 $r=P_S-T$，判断 $r\in\{a_1,a_2,\cdots,a_{S-1}\}$ 成立与否:

1) 如成立，设 $r=a_i$，则解为 $\{a_1,a_2,\cdots,a_{i-1},a_{i+1},\cdots,a_S\}$，算法结束;

2) 如不成立，则用 r 依次与 $P_1,P_2,\cdots,P_{S-1}$ 比较，一定会有 $P_1<P_2<\cdots<P_{t-1}<r\leqslant P_t<P_{t+1}<\cdots<P_{S-1}$，接着判断:

① 如果 $P_t=r$，则判定解为 $\{a_{t+1},a_{t+2},\cdots,a_S\}$，算法结束。

② 如果 $P_t>r$，则从原集合 A 中去掉 $P_t=a_1+a_2+\cdots+a_t$ 中最大的那个元素 a_t，转步骤 1。

如果第 1 阶段未得出判定(有解或无解)，则算法转至第 2 阶段继续执行。第 2 阶段不执行排序预处理，而是先用集合 B(原始集合)填充集合 A，然后从步骤 1 开始求解，故而在求解流程上也稍有差异。

2.2　算法时间复杂度

算法中执行频度最高的是步骤 2 计算 $P_K=\sum_{i=1}^{K}a_i$，$1\leqslant K\leqslant n$ 的过程中的加法运算。

在算法第 1 阶段，假设集合 A 的初始元素个数为 n，则第 1 次执行步骤 2，最多有 n 次加法。

第 2 次执行步骤 2，由于集合 A 已被缩减了 1 个元素，故最多有 $n-1$ 次加法。

第 3 次执行步骤 2 时，由于集合 A 已经被缩减了 2 个元素，故最多有 $n-2$ 次加法。

……

归纳起来，算法第 1 阶段加法总数不超过: $n+(n-1)+\cdots+1=(1+n)n/2$

所以，m 个阶段加法总数不超过: $m(1+n)n/2$

因此，算法的时间复杂度为 $O(n^2)$。

2.3　算法判定结果正确率

文献[10]已证明: 对于客观上无解的问题，算法可正确地得出无解判定; 对于客观上有解的问题，

如果该问题满足规定的条件,则算法可正确地得出的有解判定。如果问题不满足条件,则判定过程带有偶然性。

对于客观上有解的问题,依照生日问题(birthday problem)[12]的原理,在阶段1、2中由 r 寻找子集和的过程,实质上是一个诱导引发碰撞的过程,虽然带有随机性,却大幅降低了计算复杂度。下面将会证明,通过为算法增设阶段3,阶段4,……,阶段 m,每个阶段都使用相同的元素值而元素排列顺序各不相同,可以增加碰撞机率,从而提高判定正确率。

假设某次执行步骤2找到 $P_S = T + r$ 且有 $r > 0$。显然,r 与 $P_1, P_2, \cdots, P_{S-1}$ 均服从 $\{1, 2, \cdots, T-1\}$ 上的均匀分布。

r 与某 $P_i\ (1 \leqslant i < S)$ 发生碰撞的概率为:

$$P(r, P_i) = \frac{1}{T-1} \tag{1}$$

无碰撞的概率为:

$$1 - P(r, P_i) = \frac{T-2}{T-1} \tag{2}$$

r 与 $P_1, P_2, \cdots, P_{S-1}$ 均无碰撞的概率为:

$$[1 - P(r, P_i)]^{S-1} = \left(\frac{T-2}{T-1}\right)^{S-1} \tag{3}$$

假设在算法第 j 阶段($1 \leqslant j \leqslant m$),步骤2共执行了 k 次,其中第 i 次共执行了 S_i 次加法运算($1 \leqslant i \leqslant k$)。将算法第 j 阶段无碰撞的事件空间记为 A_j,则有:

$$P(A_j) = \left(\frac{T-2}{T-1}\right)^{S_1-1}\left(\frac{T-2}{T-1}\right)^{S_2-1}\cdots\left(\frac{T-2}{T-1}\right)^{S_k-1} = \left(\frac{T-2}{T-1}\right)^{\sum_{i=1}^{k} S_i - k} \tag{4}$$

根据2.2节时间复杂度的计算,上式中 $\sum_{i=1}^{k} S_i - k = O(n^2)$。故式(4)可近似为:

$$P(A_j) = \left(\frac{T-2}{T-1}\right)^{n^2} \tag{5}$$

所以,算法第 $1 \sim m$ 阶段无碰撞的概率为:

$$P\left(\bigcap_{j=1}^{m} A_j\right) = \left(\frac{T-2}{T-1}\right)^{n^2 m} \tag{6}$$

而算法第 $1 \sim m$ 阶段均发生碰撞的概率为:

$$1 - P\left(\bigcap_{j=1}^{m} A_j\right) = 1 - \left(\frac{T-2}{T-1}\right)^{n^2 m} \tag{7}$$

随着 n 增大,上述碰撞概率快速逼近1。

3 随机试验

试验比较了仅含第1阶段的QAIDR算法(下称QAIDR-1)、两阶段QAIDR算法(QAIDR)以及指数时间复杂度算法(EXP)的试验数据。其中,按照集合 A 元素数目 n 取值的不同分成4组,以测试输入问题规模对算法的影响。每组生成10 000个随机问题样本(A, T),每个问题由上述3个算法独立求解。因EXP算法的正确性已被认可[1],它可用来判定每个问题样本客观上是否有解及统计每组试验有解问题数 Q_Y、无解问题数 Q_N。对于单个问题,将QAIDR及QAIDR-1判定结果分别与EXP比较,判定结果相同则算成功,否则算失败,3个算法判定正确率及时间开销统。

3.1 试验环境

用于试验的PC配置为Intel Pentium G840@2.8 GHz CPU、4 GB RAM、Windows XP操作系统。由于每组10 000个问题的样本空间 Ω 可任意划分为若干子空间之和,故利用 m 台相同配置的计算机,每台计算机上最多可同时运行 k 个测试进程,每个进程负责生成并测试一个样本子空间 $\Omega_{i,j}$,其中 $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant k$。定期监测每组试验已经产生的样本总数,只要满足 $\sum_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant k} |\Omega_{i,j}| \geqslant |\Omega| = 10\,000$,则本组试验结束,取其中10 000个样本的统计结果作为本组测试结果。

根据问题规模 n 不同,k 的取值以CPU和内存使用率不超过100%为上限,而m取值只受计算机池容量的限制。例如,当 $n = 512$ 时,取 $k = 4$,使用 $m = 30$ 台计算机连续计算约12小时可得出数据。试验数据见表1。

表1 EXP、QAIDR-1、QAIDR性能比较

(a) $n = 32, Q_Y = 4\,970, Q_N = 5\,030$

	EXP	QAIDR-1	QAIDR
有解判定次数、正确率/%	4 970 100	4 745 100	4 855 100
无解判定次数、正确率/%	5 030 100	5 255 95.72	5 145 97.77
总正确率/%	100	97.75	98.85
总耗时/ms	95 153	253	409

(b) $n = 128, Q_Y = 4\,999, Q_N = 5\,001$

	EXP	QAIDR-1	QAIDR
有解判定次数、正确率/%	4 999 100	4 985 100	4 997 100
无解判定次数、正确率/%	5 001 100	5 015 99.72	5 003 99.96
总正确率/%	100	99.86	99.98
总耗时/ms	2.07E+7	1 062	1 699

（续表 1）

（c）$n=256, Q_Y=4\ 923, Q_N=5\ 077$

	EXP	QAIDR-1	QAIDR
有解判定次数、正确率/%	4 923 100	4 921 100	4 923 100
无解判定次数、正确率/%	5 077 100	5 079 99.96	5 077 100
总正确率/%	100	99.98	100
总耗时/ms	2.46E+8	1 995	3 199

（d）$n=512, Q_Y=5\ 101, Q_N=4\ 809$

	EXP	QAIDR-1	QAIDR
有解判定次数、正确率/%	5 101 100	5 101 100	5 101 100
无解判定次数、正确率/%	4 809 100	4 809 100	4 809 100
总正确率/%	100	100	100
总耗时/ms	5.08E+9	4 690	7 520

3.2　试验结果分析

从以上试验数据可以看出，在判定正确率方面，在所有问题样本空间上，有解判定正确率均保持为 100%，这说明算法有解判定结果的确凿性。对于无解判定而言，在问题规模 n 较小时，算法包含了较低的错误率。随着 n 变大，无解判定正确率逐渐逼近 100%，这说明算法对元素个数较多的大集合特别有效。在时间效率方面，可以看出，QAIDR 算法在时间效率上较传统指数时间复杂度算法有显著的优势，且 n 越大优势约明显。

试验结果表明，QAIDR 算法在时间效率上较传统指数时间复杂度算法有显著的改善。在问题样本空间上，有解判定 100% 正确; 无解判定包含了较低的错误率，且随着问题规模 n 变大，判定正确率逼近 100%。

4　结束语

理论和试验数据证明，与传统指数时间复杂度算法相比，QAIDR 算法具有较快的判定速度，且对于元素个数多的大集合具有很高的正确率，因而具备了实用价值。

参考文献:

[1] CORMEN T H, LEISERSON C E, RIVEST R L, et al. Introduction to algorithms [M]. 3rd ed. New York: MIT Press and McGraw-Hill, 2009.

[2] The Free Encyclopedia. The subset sum problem [EB/OL]. (2011-08-01) [2011-08-10]. http://en.wikipedia.org/wiki/Subset_sum_problem.

[3] DING Yanyan, FEI Xiangdong, PAN Yu. Security reconsideration of knapsack public-key cryptosystem [J]. Journal of Computer Applications, 2012, 32(3): 694 - 698.

[4] LI Linying, MA Guifeng, WANG Jincai, et al. The multi-knapsack public key cryptosystem generated by the vectors' production [J]. Network and Computer Security, 2011(3): 43 - 46.

[5] SHARMA S, SHARMA P, DHAKAR R S. RSA algorithm using modified subset sum cryptosystem [C] // 2nd International Conference on Computer and Communication Technology (ICCCT). 2011: 457 - 461

[6] MINE T. An implementation of space - time tradeoff method for subset sum problem [C] // 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT). 2011: 618 - 621.

[7] SCHROEPPEL R, SCHAMIR A. A T-O(2n/2), S-O(2n/4) algorithm for certain NP-complete problems [J]. SIAM Journal on Computing, 1981, 10(3): 456 - 464.

[8] LI Kenli, LI Qinghua, ZHANG Hongjun. An improved algorithm for the subset sum problem [J]. Computer Science, 2003, 30(11): 16 - 17.

[9] MIN Sihe, YAN Shijian. Elementary number theory [M]. 3rd ed. Beijing: Higher Education Press, 2003: 1 - 20.

[10] WANG Wei, QIU Weixing. Application of integer division with remainder in subset sum problem [J]. Computer Engineering, 2011, 37(S1): 191 - 193, 208.

[11] HAR-LELED S. CS 473g Algorithms [EB/OL]. (2006-07-28) [2011-11-12]. http://www.cs.uiuc.edu/class/fa07/cs473g/files/book.pdf.

[12] LU Kaicheng. Computer cryptography. [M]. 3rd ed. Beijing: Qinghua University Press, 2003: 351 - 353.

作者简介:

王　蔚（1970 -），男，江苏南京人。南京邮电大学计算机学院讲师。主要研究方向为组合优化、通信网络优化。

邱伟星（1952 -），男，上海市人。南京邮电大学计算机学院副教授。主要研究方向为数论、组合优化、密码学。

（本文责任编辑: 寇笑笑）