Introduction to Algorithm
ID: 519021911248
Name: ZhuoHao Li

Homework 5

2021-11-11

Question 1.

1. 假设你设计了一个矩阵运算加速器的专用集成电路，可以快速对矩阵求逆。但电路设计出了点小问题，导致求解的逆矩阵的一些行或列中会出现某个数错误的情况。为了校验电路设计是否正确，需要对其输出结果进行检验。请参考随机素数测试算法，设计一个时间复杂度不超过 $O(n^2)$ 的随机算法对电路输出进行检验。简单起见，假设电路处理的矩阵是方阵，维度为 n 且 n 远大于 1。请给出算法伪代码，分析算法复杂度，并对解正确的概率进行分析计算。

My Answer 1. 这样考虑问题：

> 问题可以形式化为：我们现有一个确定的待求逆的矩阵 A，矩阵运算加速器接受该矩阵作为输入参数，然后输出一个待测试的逆矩阵 B，如果 B 是正确的，即 B 满足 *AB=E(*)*.

可以根据 *(*)* 式求解，随机选取 A 的某一行 $m_i = (a_{i1}, a_{i2}, ..., a_{in}), 1 \le i \le n$ 和 B 的某一列 $n_j = (b_{1j}, b_{2j}, ..., b_{nj}), 1 \le j \le n$，做行向量和列向量的乘法（对应相乘）,$P_{ij} = \sum_{k=1}^{k=n} a_{ik} b_{kj}$，显然，我们有下面的关系：

$$P_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \begin{cases} 0, i \ne j \\ \\ 1, i = j \end{cases}$$

定义以下操作为一个【选择】：随机地选取 A 的某行，然后随机选取 B 的某列。因为 n 足够大（远大于 *1*），相邻两次选择相等的概率 *P(equal adjacent choice)*=$\frac{1}{n^2}$, 认为这是非常小的, 认为这不可能。因此我们随机 m 次选择，其中某一次 *i=j* 的概率为 $\frac{1}{n}$，这也是非常小的，我们对每一次选择都求 P，检查 P 的值。如果 P 的值不为 *1* 或 *0*，那么立刻停止并返回一个 *false boolean*。如果 P 的值为 *1*，检查 *(i==j?)*，*if false*，程序也立刻停止并且返回一个 *false*。如果 P 的值为 *0*，检查 *(i!=j?)*，*if false*，程序也立刻停止并返回一个 *false*。检查完上面三种中断情况后程序仍未终止，则继续计算下一个 P。

概率计算 每一次选择，*i=j* 的概率为 $\frac{1}{n}$，因此期望为 $1 \times \frac{1}{n} + 0 \times \frac{n-1}{n}$，经过算法后，正确的概率为 $1 - \frac{1}{n^m}$，这在 n 非常大的时候十分趋向于 *1*；如果判断为错误，结果百分百正确。

时间复杂度 该算法的时间复杂度为 *O(mn)*，主要出现随机寻找 m 个选择和对于每一个选择计算对应的 P 当中，每次计算 P 会带来 n 的计算量，所以总的来说时间复杂度为 *O(mn)*<*O(*n^2*)*

pseudo code

Algorithm 1 MATRIX-MULTIPLICATION(A,B)

Require: $n \geq 0, n$ *is much larger than* 0
Require: A,B is two matrixes, A is the input, B is the output

```
Initialize i[0...m-1]= random choose m rows in A    ▷ 随机选取 A 中 m 行，i[] 存放的是选取 row 的 number
Initialize j[0...m-1]= random choose m columns in B    ▷ 随机选取 B 中 m 列,j[] 存放的是选取 column 的 number
Initialize Pij=0
for i' and j' in range m do    ▷ i[i'] 是所取的 A 的行数，j[j'] 是所取的 B 的列数
    for k in range n do
        Pij+ = a_{i[i']k} b_{kj[j']}
    end for
    if Pij!=0 or Pij!= 1 then
        break
        return FALSE
    end if
    if  Pij == 0 and i=j then
        break
        return FALSE
    end if
    if  Pij == 1 and i!=j then
        break
        return FALSE
    end if
end for
   if TRUE
return TRUE
```

My Answer 2. 借鉴 *MIT6.046* 上讲到的 *Frievald's Algorithm,* 我给出第二种有意思的算法.

先将问题形式化：

> 问题模型*:* 对于任意连两个 $n \times n$ 的矩阵 A, B*,* 我们需要检测出 $A \times B$ 是否与 E 相等*.*

稍加思考可以发现，这种问题的表示方法和我们期望解决的问题是相通的！一般的，问题的 E 可以换成任意矩阵 E。问题本身十分简单，通常的做法是将 A 与 B 进行矩阵的乘法运算，并将结果与 C 进行比较。然而矩阵的乘法运算本身是比较费时的，所需要的时间复杂度为 $O(n^3)$, 即使使用复杂的施特拉森算法 *(Strassen Algorithm)* 也需要 $O(n^{log_2 7})$ 的时间复杂度。对于该问题，我们并不需要计算出 AB 的具体数值，因此通过随机算法能够很容易的将求解的时间复杂度降为 $O(n^2)$ ，再配合多次求解，从而使得算法的正确性也得以提高。

算法分析:

1. 从 *0* 和 *1* 中随机取 n 次，组成一个长度为 n 的列向量 γ
2. 判断 $A(B\gamma)$ 与 $C\gamma$ 是否相等，即判断 $A(B\gamma) - C\gamma = 0$?

3. 若相等，则认为 $AB = C$, 否则我们可认为 $AB \neq C$

算法的时间复杂度主要来自第二步的矩阵运算 (包含三次 3×3 矩阵与 3×1 矩阵的乘法运算，一次矩阵减法): $O(3n^3) + O(n) = O(n^2)$

下面来探讨一下算法的正确性：对于算法的第二步我们可以对其进行分解：

$$A(B\gamma) - C\gamma = (AB - C)\gamma = 0$$

因为矩阵不具备消去率，因此上面的关系仅仅是 $AB = C$ 的必要不充分条件。根据问题的不同，我们可以将算法的执行分成下面两种情况：

1. 如果 $AB = C$，那么执行的结果一定是正确，P(true)=1

2. 如果 $AB \neq C$ ，那么执行的结果有可能是错误的，$P(true) \leq \frac{1}{2}$ ，即出错的可能性 $\leq \frac{1}{2}$

对于第一种情况，由矩阵乘法的性质，很显然是正确的。而第二种情况我们将基于随机算法的特点对其进行分析，这将是整个算法的重点与难点所在, 接下来我们将证明第二种情况算法执行正确的概率 $\geq \frac{1}{2}$

证明:

令 $D = AB - C$，等价于证明 $D \neq 0$，$P(D\gamma \neq 0) \geq \frac{1}{2}$，假设 R 为所有能使 $D\gamma = 0$ 的 γ 的集合，我们的目标是要找出符合这样的 γ 的数量。
$D = AB - C \neq 0 \rightarrow \exists i,j$ 使得 $d_{ij} \neq 0$, 令向量 v 为 $v_j = 1$，其余都为 0 的列向量，则：

$$D \times v = \begin{pmatrix} d_{1j} \\ d_{2j} \\ . \\ . \\ . \\ d_{ij} \\ . \\ . \\ . \\ d_{nj} \end{pmatrix}$$

$(Dv)i = dij \neq 0 \rightarrow Dv \neq 0$，对于 $\forall r_k \in R$, 均有 $Dr_k = 0$ 我们根据 r_k 第 i 行的数值是 0 还是 1，令：$r' = r_k + v$ 或者 $r' = r_k - v$
$Dr' = D(r_k + v) = Dr_k + Dv = Dv \neq 0$，减法同理。其中 r_k 是随机生成的能够使我们算法出错的向量，而 r' 是使得算法正确的向量，并且 r_k, r' 的差别仅仅在第 i 行！这就意味着，当 $AB \neq C$，每当有一个随机产生的能使我们算法出错的向量 r_k，我们均能找到一个与之对应的能使算法正确的向量 r'，所以,

$$|r'| \geq |r_k|$$

因此执行正确的概率 $\geq \frac{1}{2}$

这样的话，算法执行 m 次后正确的概率 $P(true) \geq 1 - \frac{1}{2^m}$！当 m 足够大的时候，算法的正确性我们是可以接受的！

pseudo code:

Algorithm 2 Strassen(A,B)

Require: $n \geq 0, n$ *is much larger than* 0

Require: A,B is two matrixes, A is the input, B is the output

```
γ = randomly select 0/1 for n times, then combine them as a column vector
if A(Bγ)=Cγ then
    return TRUE
else
    return FALSE
```