Graduation Project Report

Spring 2024

Hanheng He (400485161)

Course Instructor: Dr. Antoine Deza

April 10, 2024

Contents

1 Introduction

Let matrix $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, with the *spectral norm* defined as $||A||_s = sup_{x\neq 0}|Ax|/|x|$, the *condition number* of A is $c(A) = ||A||_s||A^{-1}||_s$. The *condition number* is a measurement of the sensibility of the equation $Ax = b$ when right hand side is changed [1]. If $c(A)$ is large, then A is called $ill - conditioned$.

With such an importance property of *condition number*, ill-conditioned matrices are important in numerical algebra, and have been studied extensively by various researchers, such as [2], [3] and [4]. In [5], researchers restricted entries into set $\{0, 1\}$ or $\{-1, 1\}$, denoted by $\mathcal{A}_n^1$ or $\mathcal{A}_n^2$, which they call $anti - Hadamard$ matrices, with is of interest in linear algebra, objects in combinatorics and related areas. With such conditions, many quantities are equivalent to the condition number. Let A be a non-singular $(0, 1)$ matrix, $B = A^{-1} = (b_{ij})$, the following quantity as an equivalent condition number is considered in [5]:

$$\chi(A) = max_{i,j}|b_{ij}| \text{ and } \chi(n) = max_A\chi(A),$$

which is bounded controlled by n and some absolute positive constant c. Meanwhile, since matrix A only contains 0 and 1, it also stands for a 0/1-polytope space, which is of great interest in geometry.

In this report, we aim to construct an ill-conditioned $(0, 1)$ matrix C satisfied

$$\chi(C) \geq 2^{\frac{1}{2}n\log n - n(2+o(1))}.$$

That is said, matrix C has a controlled lower bound of it's *condition number* with respect to n.

As is shown in [1], we started from generating matrix $A \in \mathcal{A}_n^2$, and we generate $B \in \mathcal{A}_n^1$ based on A. Using a series of matrix B with different shapes, we can further concatenate a matrix C with it's condition number controlled.

2 Generate Ill-conditioned Matrix C

2.1 Generate Set Ω

To start constructing matrix C with a controlled *condition number*, the set Ω with special restriction is necessary. Let $|\cdot|$ denotes cardinality and Δ denote symmetric different. Let $m \in \mathbb{Z}^+$, $n = 2^m$, set $\Omega = \{\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n\}$ with $n + 1$ elements is required to create with a restriction that $|\alpha_i| \leq |\alpha_{i+1}|$ and $|\alpha_i\Delta\alpha_{i+1}| \leq 2$. Shown in [6], Ω is proved to be existed, the only thing considering is the implementational way create such kind of set. The following description shows and implemented way.

Let $\alpha_0 = \{\varnothing\}$. We also have $\alpha_1 = \{\varnothing\}$. Suppose we have $\Omega = \{\{\varnothing\}, \{\varnothing\}, \{1\}, \{2\}, ..., \{m\}\}$ initially. Conidering for every time we only take out all the sets with maximum size in Ω, and insert only one element inside by order. That is, for an existed set $\{1\}$, we insert element one by one $2, 3, ..., m$ by order.

With this way of insertion, we only need to consider if the conditions are met between the last old set and the first new set, and the continuous new sets come from different old sets. For all the sets come from the same old set, the conditions are automatically met.

```
Initially Omega = {{}, {}, {1}, {2}, ..., {m}}, take {1}, {2}, ..., {m} out,
and insert only one element each time. Make sure conditions are met between
    1. {m} and {1, _};
    2. {1, _} and {2, _}, {2, _} and {3, _}, ...
```

For the first case above, the only element we can insert is m, which is the first element if we revert the ordered insertion. For the second case, let's say we have set α with size k and β with size $k-1$, if $\alpha \cup \beta = \alpha$, we can insert any element we wish; if $\alpha \cup \beta \neq \alpha$, we can only insert an element $r \in \alpha$. Since we always insert element in order or in reversed order, if the first element of the insertion list is not in α, the last element of the insertion list, or the first element of the reverted list must be in α.

The following pseudo code shows the way generating Ω:

Algorithm 1 Generate Ω

Input: m
Output: Ω

$\Omega \leftarrow \{\{\}, \{\}, \{1\}, \{2\}, ..., \{m\}\}$
for $i \leftarrow 1$ to $m-1$ **do**
    $\Phi \leftarrow$ *sets in* Ω *with size equals* i
    for ϕ in Φ **do**
        $\alpha \leftarrow$ [max(ϕ) + 1, ..., m]
        if α not in Ω[-1] **then**:
            $\alpha \leftarrow \alpha$.reverse()
        end if
        for a in α **do**
            $\Omega \leftarrow \{\Omega..., \{\phi..., a\}\}$ ▷ ϕ... means extend set ϕ
        end for
    end for
end for
return Ω

when $m = 4$, set Ω is shown below:

```
Omega = {set(), set(), {1}, {2}, {3}, {4},
       {1, 4}, {1, 3}, {1, 2}, {2, 4}, {2, 3},
       {3, 4}, {1, 3, 4}, {1, 2, 3}, {1, 2, 4},
       {2, 3, 4}, {1, 2, 3, 4}}
```

2.2 Construct Matrix $A \in \mathcal{A}_n^2$ with Set Ω

Shown in [1], with set $\Omega = \{\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n\}$ satisfies $|\alpha_i| \leq |\alpha_{i+1}|$ and $|\alpha_i \Delta \alpha_{i+1}| \leq 2$, matrix A can be constructed as follows such that $\chi(A) = 2^{\frac{1}{2}n \log n - n(1+o(1))}$:

For every $1 \leq i, j \leq n$:

$$a_{ij} = \begin{cases} -1,\ \alpha_j \bigcap (\alpha_{i-1} \bigcup \alpha_i) = \alpha_{i-1} \Delta \alpha_i \ and\ |\alpha_{i-1} \Delta \alpha_i| = 2 \\ (-1)^{|\alpha_{i-1} \bigcap \alpha_j|+1},\ \alpha_j \bigcap (\alpha_{i-1} \bigcup \alpha_i) \neq \varnothing \ but\ does\ not\ meet\ the\ condition\ above \\ 1,\ \alpha_j \bigcap (\alpha_{i-1} \bigcup \alpha_i) = \varnothing \end{cases} .$$

With the Ω shown in section 2.1, matrix $A \in \mathcal{A}_n^2$ constructed is shown below:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 \end{bmatrix} .$$

In [1], researchers further discussed some properties of the $\{-1, 1\}$ matrix. Let symmetric Hadamard matrix Q be an n by n matrix given by $q_{ij} = (-1)^{|\alpha_i \bigcap \alpha_j|}$, that is $Q^2 = nI_n$. There existed a lower triangular matrix L built by Ω satisfying $A = LQ$. In this report, we build matrix A and show $L = AQ^{-1}$ is a lower triangular matrix.

With matrix A shown above, matrix Q and L is shown below:

$$Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix} ,$$

$$Q \times Q = \begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{bmatrix} ,$$

$$L = \begin{bmatrix}
1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.5 & 0.5 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.5 & 0.5 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.5 & 0.5 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.5 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.0 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.25 & 0.25 & 0.0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.0 & 0.0 & 0.0 & -0.75 & 0.25 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.25 & 0.0 & 0.25 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.25 & -0.75 & 0.25 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.0 & 0.25 & 0.25 & 0.0 & 0.0 & -0.75 & 0.25 & 0.0 & 0.0 \\
0.0 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.25 & 0.0 & 0.0 & -0.75 & 0.25 & 0.0 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -0.875 & 0.125
\end{bmatrix}.$$

2.3 Mapping Matrix $A \in \mathcal{A}_n^2$ to $B \in \mathcal{A}_{n-1}^1$

With matrix $A \in \mathcal{A}_n^2$, $\chi(A) = 2^{\frac{1}{2}n\log n - n(1+o(1))}$ being constructed, a mapping can be implemented to generate a matrix $A \in \mathcal{A}_{n-1}^1$ such that $\chi(B) = 2^{\frac{1}{2}n\log n - n(1+o(1))}$. Consider the map Φ which assigns to any matrix $B \in \mathcal{A}_{n-1}^1$ a matrix $\Phi(B) \in \mathcal{A}_n^2$ in the following way:

$$\Phi(B) = \begin{pmatrix} 1 & 1_{n-1} \\ -1_{n-1}^T & 2B - J_{n-1} \end{pmatrix}.$$

We can see that $\Phi(B)$ is a mapping $\mathcal{A}_n^2 \to \mathcal{A}_{n-1}^1$ with a series of linear operations. Therefore, we have the following reversing way to construct matrix $B \in \mathcal{A}_{n-1}^1$ with $A = \{a_{ij}\} \in \mathcal{A}_n^2$:

$$B = \frac{1}{2}(J_{n-1} + \{a_{ij}\}_{2\le i\le n, 2\le j\le n}).$$

Notice that in the above section, the $A \in \mathcal{A}_n^2$ we constructed has it's first column as:

$$\begin{pmatrix} 1 \\ 1_{n-1}^T \end{pmatrix},$$

so we need to negative the first column, and have the relation between matrix $B \in \mathcal{A}_{n-1}^1$ and $A \in \mathcal{A}_n^2$ in the implementation shown as follows:

$$B = \frac{1}{2}(J_{n-1} - \{a_{ij}\}_{2\le i\le n, 2\le j\le n}).$$

With the matrix $A \in \mathcal{A}_n^2$ constructed above, a constructed matrix B is shown below:

$$B = \begin{bmatrix}
1&0&0&0&1&1&1&0&0&0&1&1&1&0&1 \\
0&1&0&0&0&1&1&1&0&0&1&1&1&1&1 \\
0&0&1&0&0&1&0&0&1&1&1&1&0&1&1 \\
0&0&0&1&1&0&0&1&0&1&1&0&1&1&1 \\
1&0&0&0&0&1&1&0&0&0&0&1&0&0&0 \\
0&0&1&0&1&0&0&0&1&1&1&0&1&1&1 \\
0&1&0&0&0&1&0&1&1&0&1&1&0&1&1 \\
0&0&0&1&1&0&1&0&0&1&1&1&1&0&1 \\
0&0&1&0&0&1&0&1&0&1&1&0&1&1&1 \\
0&0&0&1&1&0&0&1&1&0&0&1&1&1&1 \\
1&0&0&0&0&0&1&0&0&1&1&0&0&1&1 \\
0&1&0&0&1&1&0&1&0&1&0&1&1&1&0 \\
0&0&0&1&0&1&1&0&1&1&1&0&1&1&0 \\
0&0&1&0&1&1&1&1&0&0&1&1&0&1&0 \\
1&0&0&0&0&0&0&1&1&1&1&1&1&0&0
\end{bmatrix}.$$

Shown in [1], matrix B preserves the ill-conditions property, that is $\chi(B) = 2^{\frac{1}{2}n\log n - n(1+o(1))}$.

2.4 Generate and Verify Ill-conditioned Matrix C

Before constructing matrix C, [1] shows a way concatenating that has a special property. Let S and T be two non-singular matrices of order n_1 and n_2. Define $S \diamond T$ as follows:

$$R = \begin{bmatrix} s_{11} & \dots & s_{1n_1} & 0 & \dots & 0 \\ s_{21} & \dots & s_{2n_1} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ s_{n_1 1} & \dots & s_{n_1 n_1} & 0 & \dots & 0 \\ 0 & 0 \dots 0 & 1 & t_{11} & \dots & t_{1n_2} \\ 0 & 0 \dots 0 & 0 & t_{21} & \dots & t_{2n_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 \dots 0 & 0 & t_{n_2 1} & \dots & t_{n_2 n_2} \end{bmatrix}.$$

It's shown that R has the following property:

$$\chi(S \diamond T) \geq \chi(S)\chi(S),$$

Which shows an increased *condition number* leading to construct the ill-conditioned C. Consider the $(0,1)$ matrix $C = A_1 \diamond (A_2 \diamond (...(A_{r-1} \diamond A_r))...)$ with order $\sum_{i=1}^{r} n_i = n$, with the definition of the operator $\diamond$, we have the following conclusion that

$$\chi(C) \geq \prod_{i=1}^{r} \chi(A_i) > 2^{\frac{1}{2} n \log n - n(2+o(1))}$$

This conclusion describes the lower bound of $\chi(C)$, and we can say matrix C is ill-conditioned with respect to n.

Since the concatenation rapidly enlarge the order of the matrix, we could not show the matrix C constructed with all the steps described in pervious sections. Instead, we show some result of $\chi(C)$ with C generate in the same steps above. Remind that $C = A_1 \diamond (A_2 \diamond (...(A_{r-1} \diamond A_r))...)$, where r is the biggest order of all the $\{0,1\}$ matrices A:

```
r = 2, order of C = 4, χ(C) = 1.0
r = 3, order of C = 11, χ(C) = 2.0
r = 4, order of C = 26, χ(C) = 260.0
r = 5, order of C = 57, χ(C) = 106491641548.6
```

With the result above, We can clearly observe that as order r grows, $\chi(C)$ may grow rapidly to infinity.

3 Complexity Analysis of Generating Matrix C

3.1 Time Complexity of Creating Ω

In this section we discuss the complexity of constructing matrix C, and further more show the reason not being able to generate matrix C with large r. Since we go through all these steps with code, we assume that we have enough memory and focus on the time complexity.

In order to generate set Ω, everytime we only need to take out one generated set and insert one new element inside. The insertion operation is $O(\log n)$, but notice that we always insert an element greater than the maximum of a set, we can consider it an $O(1)$, without considering the maintenance of the internal tree data structure. Therefore, the most ideally time complexity is $O(n)$, which cannot be reach but will be used.

A more detailed complexity analysis is shown below. In the implementation, there are 2 for-loop with $O(m)$ and at least $O(m)$ time complexity. With all the assignment, set insertion, set reversion operation and an extra for-loop are neglected, a rough time complexity estimation would be $O(n^2)$.

Algorithm 2 Time Complexity of Generating Ω

Input: m
Output: Ω

$\Omega \leftarrow \{\{\}, \{\}, \{1\}, \{2\}, ..., \{m\}\}$
for $i \leftarrow 1$ to $m-1$ **do** ▷ $O(m)$ loop
  $\Phi \leftarrow$ *sets in* Ω *with size equals* i ▷ $O(1)$ with special data structure
  for ϕ in Φ **do** ▷ $\binom{m}{i} \rightarrow$ at least $O(m)$
    $\alpha \leftarrow$ [max(ϕ) + 1, ..., m] ▷ the max(...) of an ordered set is $O(1)$
    if α not in Ω[-1] **then**:
      $\alpha \leftarrow \alpha$.reverse() ▷ a set reversion operation
    end if
    for a in α **do** ▷ $m-\max(\phi)-1$, more than $O(1)$
      $\Omega \leftarrow \{\Omega..., \{\phi..., a\}\}$ ▷ a set insertion operation
    end for
  end for
end for
return Ω

3.2 Time Complexity of Creating $\{0, 1\}$ Matrix C

To generate every entry of matrix $A \in \mathcal{A}_n^2$, a visit of three continuous element in set Ω is necessary. Let n be the order of matrix A, without considering the complexity of set accessing, the total ideally time complexity is $O(n) + O(n^2) = O(n^2)$.

Generating matrix B is a simple matrix operation, let $n-1$ be the order of B, the time complexity is $O((n-1)^2) + O(n^2) = O(n^2)$.

In order to generate matrix C, we need a series of matrix A_1, A_2, ... , A_r. As shown above, for matrix A_r with order r, the time complexity is $O(r^2)$. Therefore the time complexity generating these matrices is $\sum_{i=1}^{n} O(r^2) = O(n^3)$

Therefore, when r grows, with an $O(n^3)$ time complexity, the time required to generate matrix C grows rapidly.

4 Conclusion

References

[1] Noga Alon and Văn H Vũ. Anti-hadamard matrices, coin weighing, threshold gates, and indecomposable hypergraphs. *Journal of Combinatorial Theory, Series A*, 79(1):133–160, 1997.

[2] G. H. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the jordan canonical form. *SIAM Review*, 18(4):578–619, 1976.

[3] JH Wilkinson. Note on matrices with a very ill-conditioned eigenproblem. *Numerische Mathematik*, 19:176–178, 1972.

[4] Arnold Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM review*, 40(3):636–666, 1998.

[5] R.L. Graham and N.J.A. Sloane. Anti-hadamard matrices. *Linear Algebra and its Applications*, 62:113–137, 1984.

[6] Johan Håstad. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 7(3):484–492, 1994.

Code

Add notebook here