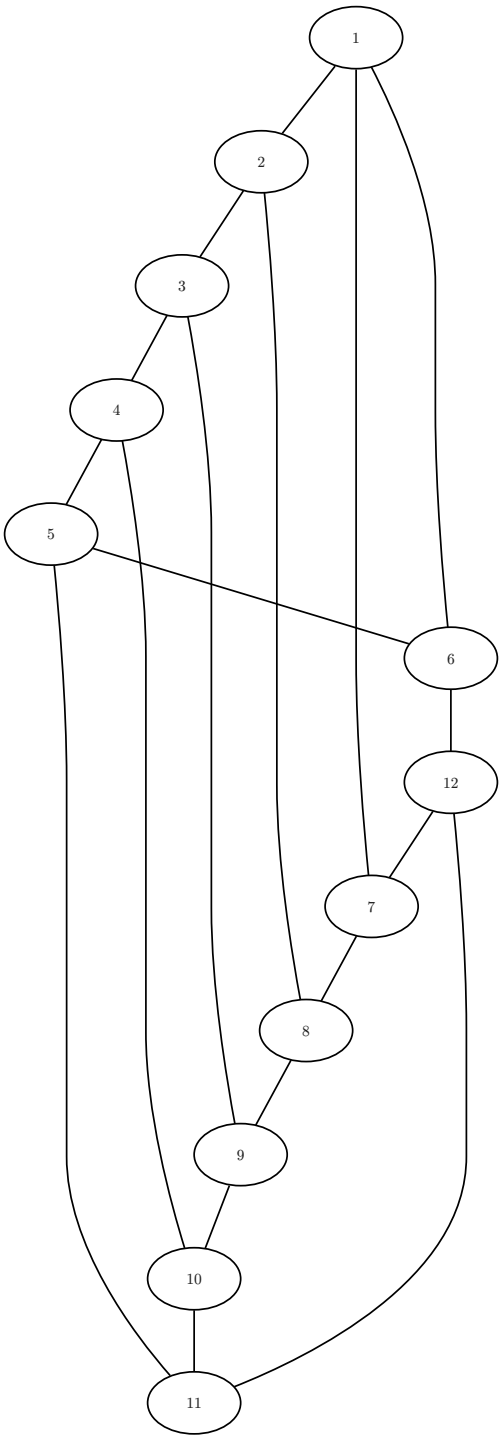
Introduction to Algorithm Engineering

Homework-2

Moida Praneeth Jain, 2022101093

Question 1

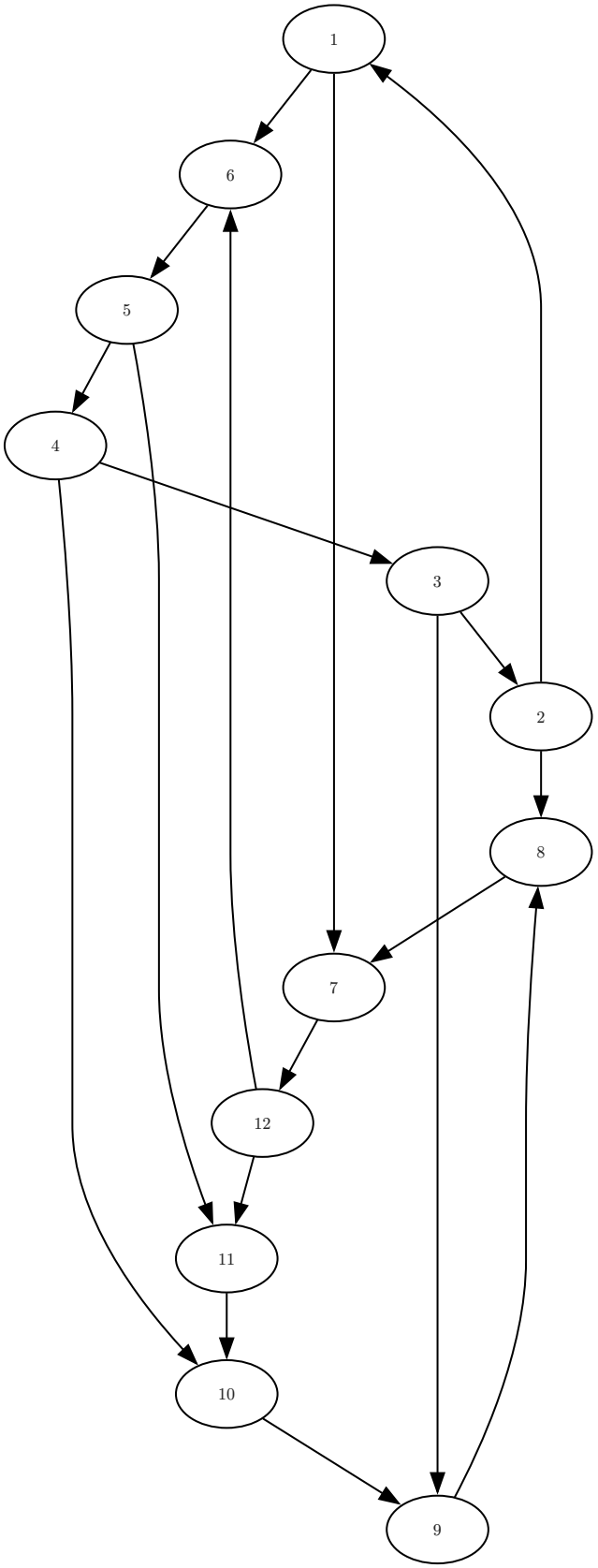
Consider graph G with 12 vertices and 18 edges, as shown below



Now, we perform the following DFS traversal, picking the root node as 1

$$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 12 \to 11 \to 10 \to 9 \to 8 \to 7$$

We construct a directed graph, with edges as part of the DFS tree pointing towards the root, and the non-tree edges pointing away from the root



Back Edge 1 - 6

$P_0 = 1 - 2$

$P_1 = 1 - 6 - 5 - 4 - 3 - 2$

$\text{visited} = \{1, 2, 3, 4, 5, 6\}$

Back Edge 1 - 7

$P_2 = 1 - 7 - 12 - 6$

$\text{visited} = \{1, 2, 3, 4, 5, 6, 7, 12\}$

Back Edge 2 - 8

$P_3 = 2 - 8 - 7$

$\text{visited} = \{1, 2, 3, 4, 5, 6, 7, 8, 12\}$

Back Edge 3 - 9

$P_4 = 3 - 9 - 8$

$\text{visited} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 12\}$

Back Edge 4 - 10

$P_5 = 4 - 10 - 9$

$\text{visited} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12\}$

Back Edge 5 - 11

$P_6 = 5 - 11 - 10$

$\text{visited} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

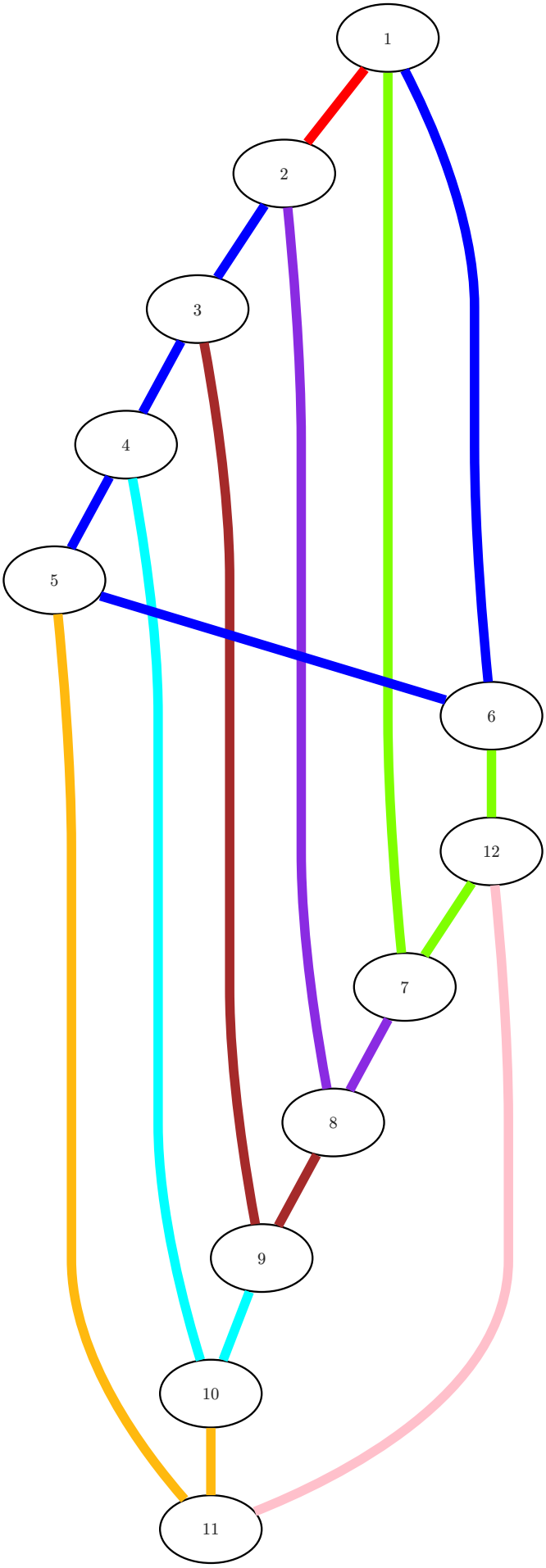
Back Edge 12 - 11

$P_7 = 12 - 11$

$\text{visited} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

$\{P_i\}$ is the ear decomposition

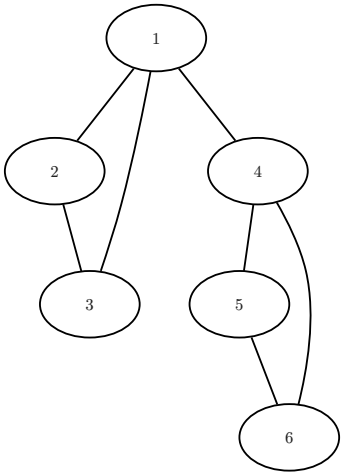
Below is a visual representation of the ear decomposition of the graph


1
2
3
4
5
6
12
7
8
9
10
11

The algorithm fails if the input graph is not biconnected due to the following reasons:

Case 1: The graph is not connected
In this case, the DFS tree itself cannot be formed, instead a forest will be formed. Thus, the back edges will not necessarily form cycles. Thus, the algorithm will fail.

Case 2: The graph is connected but not biconnected
In this case, the DFS tree will be formed, but since the graph is not biconnected, there will exist a bridge. This bridge cannot be part of any cycle, and thus will never be visited in the above algorithm. Because of this, the algorithm will fail. For example, consider the graph



Here, the edge $1 - 4$ will not be part of any cycle, and thus there will exist no ear decomposition.

Question 2

Square Matrices

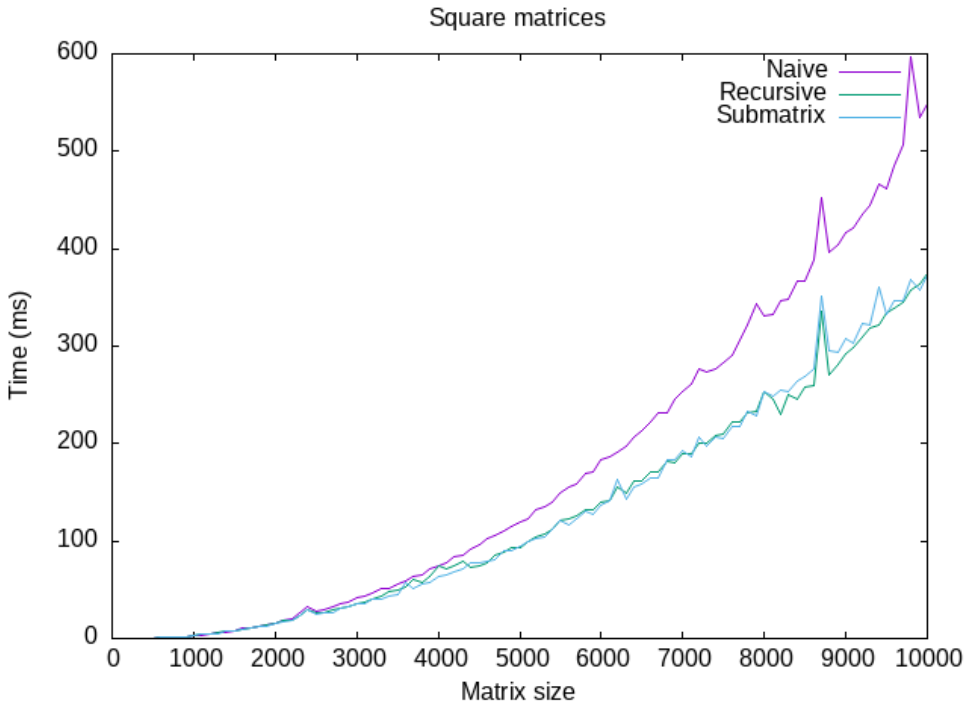


Figure 1: Variation in transpose time with size of square matrices

First, let us consider the case of square matrices. As we can observe from Figure 1, the trend is quadratic, as was noted in Homework 1. In the naive case, we get $O(N^2)$, and for the recursive and submatrix case, we get $O\left(\frac{N^2}{B}\right)$

Row Matrices

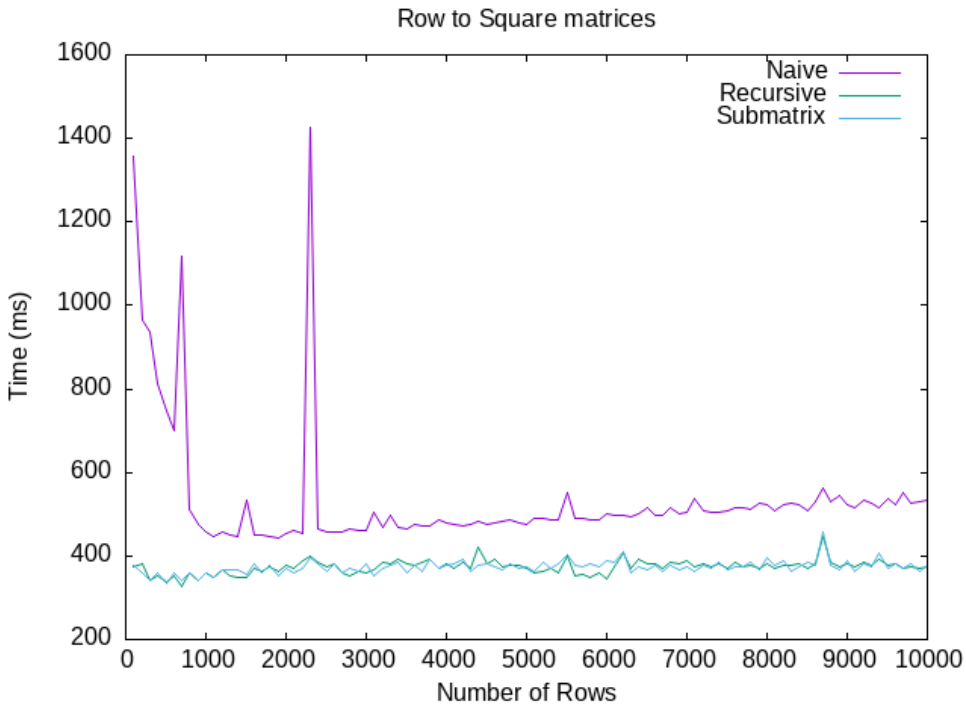


Figure 2: Variation in transpose time with rows of matrix. (1e8 matrix entries)

We fix the total number of elements in the matrix to `1e8`, and increase the number of rows to `1e4` while decreasing the number of columns from `1e8`, i.e, the plot represents the transpose time as the graph transitions from a row matrix to a square matrix.

From Figure 2 we observe that the times are approximately constant. This is because in these matrices, each row has more elements than the cache can fit, thus the cache hit/miss rate remains constant throughout.

Column Matrices

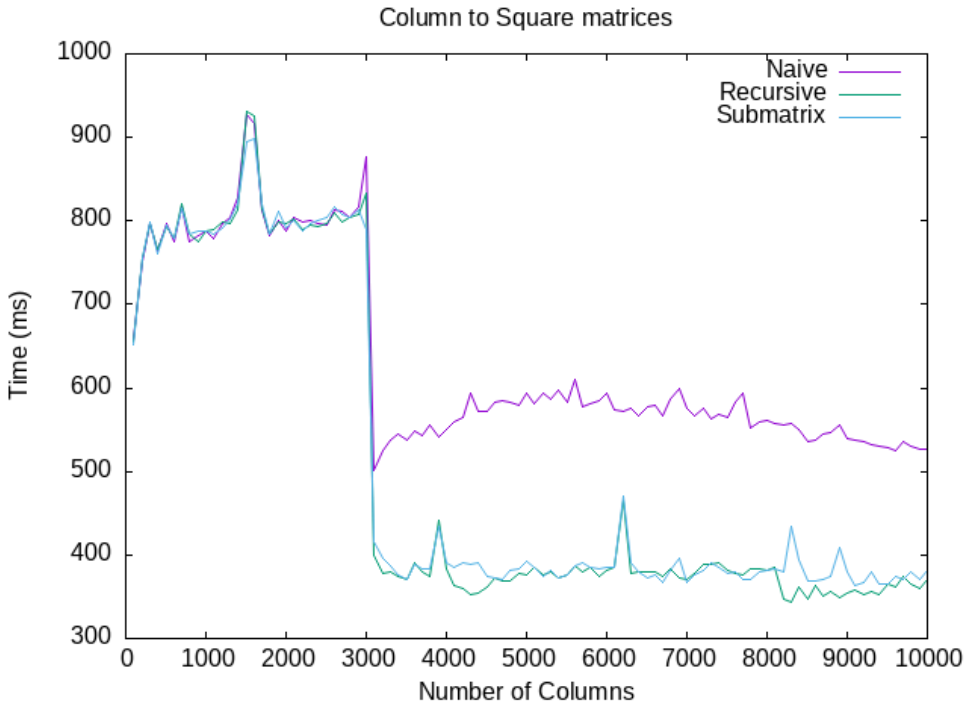


Figure 3: Variation in transpose time with columns of matrix. (1e8 matrix entries)

We fix the total number of elements in the matrix to `1e8`, and increase the number of columns to `1e4` while decreasing the number of rows from `1e8`, i.e, the plot represents the transpose time as the graph transitions from a column matrix to a square matrix.

The key point to note from Figure 3 is that these times are much higher than the row matrices. This is because column matrices are much worse for cache access than row matrices. At around `3000` columns, the times go back down. This is where the number of elements in each row (the number of columns) is large enough for the cache to get filled up.