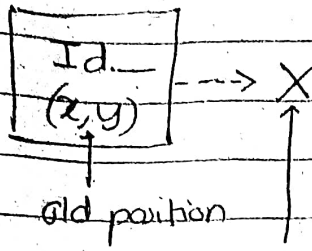
Computer Algorithm Assignment 2

1. Logic : Move recording.

• Each time a tile is moved or rotated, the system records the following information.



i) Tile ID : unique identifier
ii) Oldposition : intial position on grid.
iii) New position : new position on grid.
iv) Rotation : Rotation angle of tile (0, 90, 180, 270)

Compression using Huffman Coding :

→ • Once all moves are recorded, the system compresses the move log to save space.

• Huffman coding is ~~covered~~ chosen for compression because its effective encoding data with variable-length codes based on the frequency of elements/moves

- Frequently occuring moves are given shorter binary codes, while less frequent ones recieve longer codes, minimizing the overall storage size.

Steps : 1) Record all players move in a list
2) Count how often move occurs to build a frequency table
3) Build huffman tree based on frequency of moves (most frequent gets shorter codes)
4) Generate Binary Codes
5) Compress the move log by replacing each move with its corresponding binary code

Example: move 1 : (1, 0, 0, 2, 2, 90)
move 2 : (2, 3, 3, 4, 4, 180)
string → "1020229033344180"
Frequency → '1' → 4 times
0 → 3 times
2 → 6 times
9 → 1 times
3 → 4 times
4 → 2 times
8 → 1 times
Use these to build tree

Example :- '1' → 00
'0' → 01
'2' → 10
'9' → 1100
'3' → 1101
'4' → 1110
'8' → 1111

Encode ↓

"1020229013334418 0"

"00101010101110001101110111010"

Compression : Orignal string had 17 characters.

- Assuming each character is stored as 8 bits, the uncompressed data size is 17 * 8 = 136 bits.
- The compressed data size is 30 bits.
- Compression ratio : 136/30 = 4.53

- Matrix Chain Multiplication

Steps : 1) Collect rotation operations
2) Convert rotations to matrices
3) Use DP to find the optimal way to multiply these matrices, minimizing the computational cost.
4) Trace back optimal sequence

Note / Calculation on Efficiency & Storage Efficiency

1. Initial Move Data Representation:
   - Structure: (tileid, oldx, oldy, newx, newy, relation)
   - Bit requirement → tileid = 10bits
     Coordinates (x,y) = 6 bits each
     Relation = 2 bits.
     Total per move = 32 bits per move

2. Huffman coding for Compression

For 100 moves = 100 × 32 = 3200 (400 bytes)

- Compression – (average relation ≈ 20 bits)
  ∴ 100 × 20 bits = 2000.
- Storage Savings
  Uncompressed – 400 bytes
  Compressed – 250 bytes

(400 – 250) / 400 * 100 = 37.5% reduction in storge spac