Computer Algarithm Assignment 2
Lagic: Move recording
· Each time a tile is rooved or rotated, the system
Tile records the following linformation.
Tilo 7D: unique identifier
(2,y) (2) (iii) New position in the position on grid (2,y) (iii) New position in new position on grid (2,y) (iv) Rotation is Rotation angle of tile (0,96,180,270)
now pointion!
Compression using Huffman Coding
-> Once all moves are recorded the system
compresses the move lag to save space.
· Huffaman coding is covered chosen for compression
because its effective encoding data with variable
-length codes based on the frequency of elements/move
- Frequently occuring moves are given shorter binary
codes unité less frequent orles l'rédieve longier codes,
minimizing the overall storage size.
Sully year a data trada to
Steps : 1) Record all playes move in a list
(2) Count how often above occurs to build a
frequency I table It william
3) Build huffman thee board on frequency of
moves (most frequent gets schovier codes)
g) Generale Binny Codos
5) Compress the move log by replacing eachmove
with its corresponding binory code
Example: more 1: (1,00,2,3,90)
move 2: (2,3,3,4,4,180)
string 73 11020 2290   333 44 180"
Frequency - 1 - 4 times 7  0 - 3 times Use those to build tree
2 -> 6 History
4 + 2 times
1. Hines

Example: - 11 >00 12'-10\_ 19:->1100 3 - 1101 141-) 1110 1817 1111 Encode + 10202290133344 180" 0010101010100011011101010" Compression Orignal string had 17 eminoctors · Assuming each character is stored as 8 bits, the uncompressed data size is 17 8 = 136 bib · 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 motrices 3) Use DP to find the optimal way to multiply these motaices, minimizing the computational cost line 4) Trace back appined sequence

Initial Move Data Representation:  Structure: ('hiterid, aldx, aldy, newx, newy, relative)  Bit requirement -> tilend = 10bits  Coordinates (2,y) = 5 bits can  Rebon = 2 bits  Tatal per move = 32 bits per move  2. Huftman coding for Campressian  For 100 moves = 100 × 32 = 3200 (400 bytes)  *Conspression - (average reaction = 20 trit)  \$\frac{100 \times \times 20 \times 100 \times 20 \times 100}{100 \times 2000}  . Storage Savings  Uncompressed - 400 bytes  (400 - 250) / 400 * 100 = 37.5% reduction in storage.	Note / Calculation on Efficiency & Storage Fifti	ac
Structure ('litte id, oldx, oldy, new x, new y, redelian)  Bit requirement -> tilend = 10bit  Coordinates (2,y) = 5 bits can  Rebian = 2 bits  Total per move = 32 bits per make  2. Huftman coding for Campressian  for 100 moves = 100 × 32 = 3200 (400 bytes)  "Compression - (average rancian = 20 bits)  to 100 x 20 bits = 2000.  Storage Savings  Uncompressed - 400 bytes  Campressed - 250 bytes	Initial Move Data Representation:	
Bit requirement -> tilerd = 10bit  (Coordinates (2,y) = \( \) bits can  Rebion = 2 bits.  Total per move = 30 bits per move  2. Huftman coding for Compression  For 100 moves = 100 \times 32 = 3200 (400 bytes)  "Compression - (average rewaion \times 20 bits)  took \( \) 20bits = 2000.  Storage Sowings  Uncompressed - 400 bytes  (am pressed - 250 bytes		turne garage
Coordinates (2, y) = Ebit Can  Retion = 2 bits  Total per move = 32 bits per move  2. Huftman coding for Campvession  For 100 moves = 100 × 32 = 3200 (400 bytes)  * Campression - (average reaction = 20 bits)  \$\frac{1}{2}\$ 100 \times 20bits = 2000.  Storage Savings  Uncompressed - 400 bytes  Compressed - 250 bytes	Bit requirement -> tileid = 10bits	
Rabon = 2 bits  Total per move = 30 bits per move  2. Huftman coding for Campressian  for 100 moves = 100 × 32 = 3200 (400 bytes)  "Compression - (average reaction = 20 bits)  + 100 x x 20 bits = 2000  . Storage Savings  Uncompressed - 400 bytes  Compressed - 250 bytes		San and the
Total per move = 32 bits per move  2. Huftman coding for Compression  for 100 moves = 100 × 32 = 3200 (400 by tes)  * Compression - (average reaction = 20 bit)  \$\frac{1}{7}\$ 100 \text{ x 20 bits = 2000}\$  Storage Savings  Uncompressed - 400 by tos		Certifications
2. Huftman coding for Campressian  For 100 moves = 100 × 32 = 3200 (400 bytes)  * Compression - (average reaction = 20 bit)  \$\frac{100 \times \times 20 \times 1 \times 2000}{\times 2000 \times 2000}  . Storage Savings  Uncompressed - 400 bytes  Campressed - 250 bytes		Acres de la constante de la co
For 100 moves = 100 × 32 = 3200 (400 bytes)  "Compression - (average rewaiting 20 bit)  \$\frac{100 \times \times 20bits = 2000}{\times 2000}\$  Storage Savings  Uncompressed - 400 bytes  Compressed - 250 bytes		ette etimpotipumilini
For 100 moves = 100 × 32 = 3200 (400 bytes)  "Compression - (average rewaiting 20 bit)  \$\frac{100 \times \times 20bits = 2000}{\times 2000}\$  Storage Savings  Uncompressed - 400 bytes  Compressed - 250 bytes	2. Huftman coding for Compression	glica, proposition (see
· Storage Savings Uncompressed - 400 by to		
· Storage Savings Uncompressed - 400 by to	· Compression - (average requirem = 20 bit)	mana y provincento e
· Storage Savings Uncompressed - 400 by to	\$ 100 x x 20 bit = 2000	فلتعامل وإستان
Compressed - 400 by to		-
Commpressed - 250 by to		
(400 - 250) / 400 * 100 = 37.57.7 chuchions in storge.		
in storge:	(400 -250) / 400 \$ 100 = 37.5% reduct	7'07
	in storge	
		(control of the
		and a contract
	- 1987 -	