Bounded Homopolymer Encoding

Sivakanth Gopi

Microsoft Research

Bounded homopolymer constraints

- DNA strands are made of A,G,C,T bases
- Hard to synthesize and read strands which have long runs of consecutive repeated characters (homopolymers)
  - ATGCCTGGGGTCTACCCCCCTAAAT…
- **Goal**: Encode raw data into DNA strands which do not have long runs of repeated characters

Encoding and Decoding maps

- Fix some input length n and max homopolymer run length k which is allowed
- Find encoding map $E: \{0,1\}^n \to \{A, G, C, T\}^N$ and decoding map $D: \{A, G, C, T\}^N \to \{0,1\}^n$ such that
  - For every x, $E(x)$ has homopolymer runs of length at most k
  - For every x, $D\big(E(x)\big) = x$
  - Rate $R = n/N$ is as large as possible (bits per base)

What is the best possible rate R? (Information theoretic limit)

Max homopolymer run length (k)	Rate (n/N) as $n \to \infty$ (Bits per base)
1	1.5850
2	1.9227
3	1.9823
4	1.9957
5	1.9989
∞	2

What is the best possible rate R?

- Information theoretic limit
- Our current method for no homopolymer encoding ($k = 1$) achieves rate $R = 1.5$
  - Encode 6 bits into 4 bases using base 3 conversion
- We will now show an algorithm to encode and decode efficiently achieving the information theoretic limit for any N

Max homopolymer run length (k)	Rate (n/N) as $n \to \infty$ (Bits per base)	Rate (n/N) when $N = 100$
1	1.5850	1.58
2	1.9227	1.92
3	1.9823	1.98
4	1.9957	1.99
5	1.9989	1.99
∞	2	2

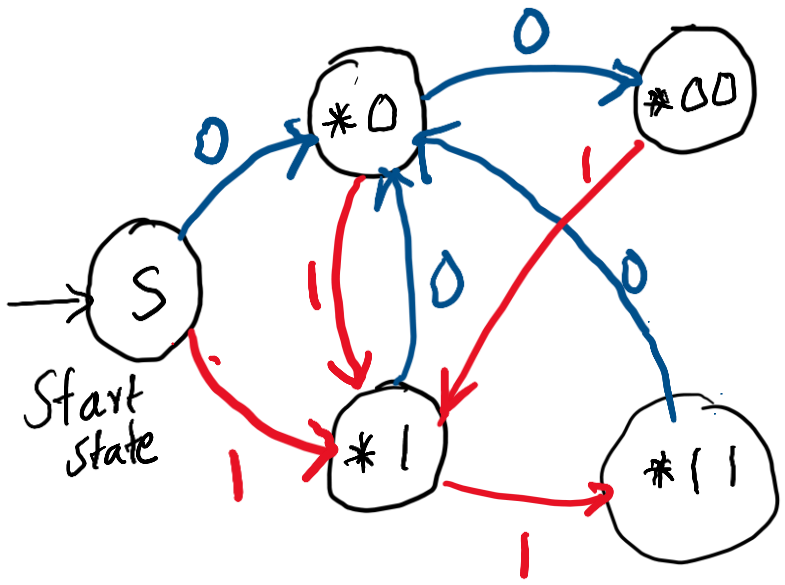
What is the optimal encoding?

- List all strings of $\{A, G, C, T\}$ of length N sorted alphabetically satisfying the run length constraint. Say there are C strings.
- Let $n = \text{ceil}(\log_2 C)$
- Encoding map:
  - E: n bit representation of $i \rightarrow i^{th}$ string in the list
- Decoding map:
  - E: i^{th} string in the list $\rightarrow$ Bit representation of i
- This is super inefficient though!

Index	Message (x)	List of all AGCT strings with run lengths at most k $E(x)$
0	0000	AAACAAATAATA….
1	0001	AAGCATAAATAA…
2	0010	AAGCATAAATAA…
3	0011	AAGCATCAATAA…
4	0100	AAGCATGAATAA…
5	0101	AAGAATAAATAA…
6	0110	AAATATAAATAA…
⋮	⋮	⋮

How to make this efficient?

- Use finite state machine to represent strings which satisfy the bounded homopolymer constraints
  - For simplicity assume that the output alphabet is also $\{0,1\}$
  - $k = 2$, we allow at most two consecutive repeated bits
- Key Observation
  - Allowed strings of length $N \equiv$ Paths of length N starting from the 'start state' in the FSM

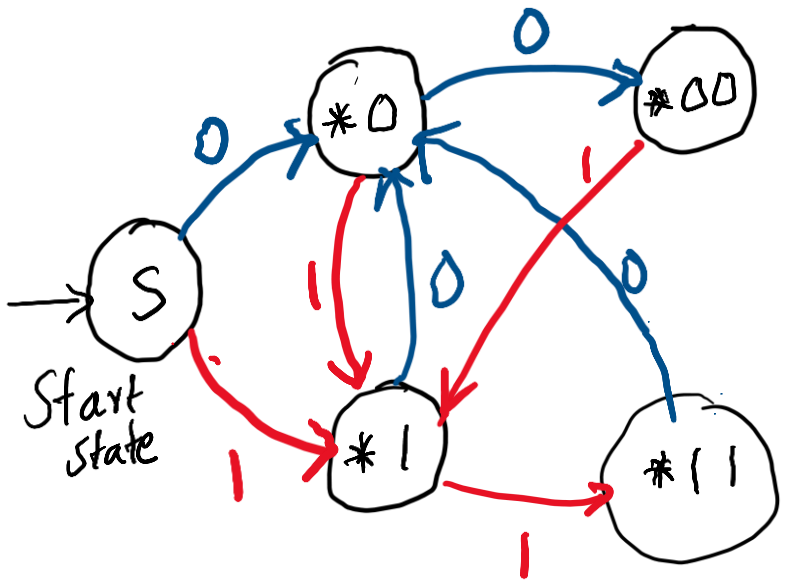


Efficient encoding

- Given i, find the i^{th} path P_i from the start state in alphabetical order
- Suppose we already calculated

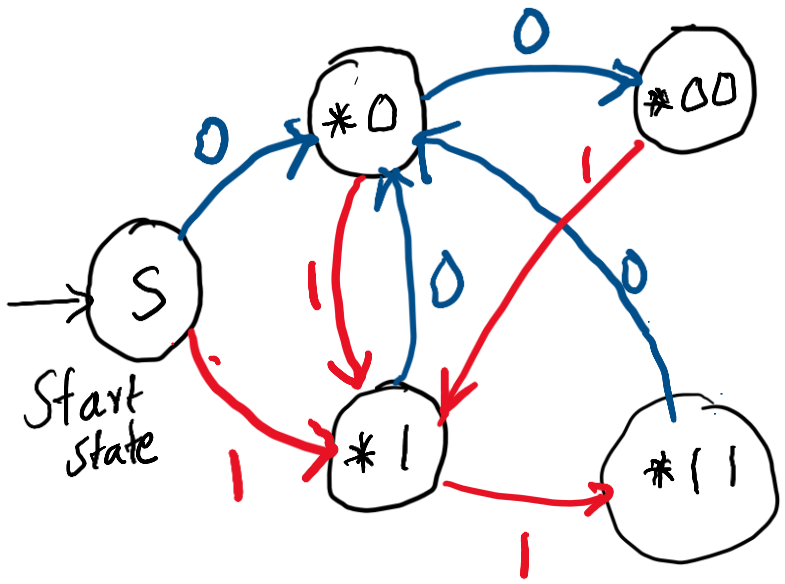
$F(s, L)$ = number of length-L paths from state s for all states s and lengths $L \leq N$

- These can be calculated by a dynamic program efficiently
  - E.g.: F(start,L) = F(*0,L-1)+F(*1,L-1)



Efficient encoding

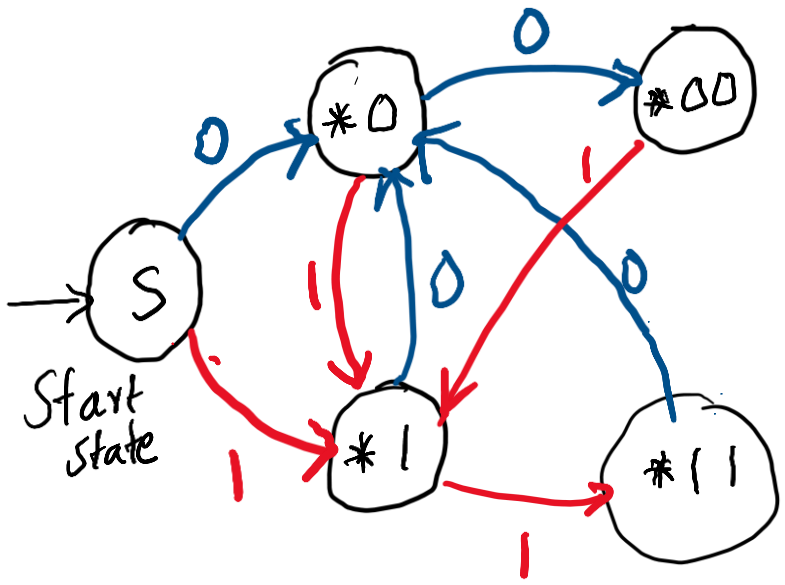
- F(start,N)=F(*0,N-1)+F(*1,N-1)
  - If $i <$ F(*0,N-1), then P_i should start with 0
  - If $i \geq$ F(*0,N-1), then P_i should start with 1
- So we can figure out the first bit using one comparison
- Now suppose $i <$ F(*0,N-1), the first bit is one
  - The second bit of P_i can be obtained similarly
    - F(*0,N-1)=F(*00,N-2)+F(*1,N-2)
    - If $i <$ F(*00,N-2), the second bit of P_i is also 0, else the it is 1.
- Continuing this way we can find the each bit of P_i using a constant number of comparisons.



Efficient decoding

- Given a path P of length N, find its position in the alphabetically sorted list of all paths of length N started from the start state
- Let i be the position of P
- If $P_1 = 0$, then $i < F(*0, N-1)$
- If $P_1 = 1$, then $i > F(*0, N-1)$

- Continuing this way, we can find i by comparisons.



Speeds

- About 50Mbps for encoding and 80 Mbps for decoding on a single core.
- Max run length 1 is about 3 times faster, because it is basically base 2 to base 3 conversion.
- Uses GMP multi precision library for maintaining large integers.