```
clc
clear
close all
img = imread('lena.tif');
imshow(img)
```

```
img_double = im2double(img);
img_uint8 = uint8(img_double * 255);
img_vector = reshape(img_uint8, 1, []);
img_bits = dec2bin(img_vector, 8);
img_bitstream = reshape(img_bits.', 1, [])
```

```
img_bitstream = 
'10001001100010011000101010000101100000001100000111000001110000011100000011000000110000010100000111000000011
```

```
dictionary = solve(img_bitstream)
```

```
dictionary = 1×35556 cell
'1'          '0'          '00'          '10'          '01'          '100'          '0···
```

```
enc = prefix_locations(dictionary)
```

```
enc = 
'0000000000000000100000000000000000000000000000000001000000000000000001000000000000000010100000000000000100000
```

```
compression_ratio = length(img_bitstream)/length(enc)
```

```
compression_ratio = 0.8674
```

```
function enc = prefix_locations(dictionary)
    n = length(dictionary);
    num_bits = ceil(log2(n));%changed
    enc = [];%changed
    result = zeros(1, n);
    for i = 1:n
        if length(dictionary{i}) == 1
            result(i) = 0;
            enc = [enc dec2bin(result(i), num_bits) dictionary{i}
(end)];%changed
        else
            prefix = dictionary{i}(1:end-1);
            matches = strcmp(prefix, dictionary);
            match_locs = find(matches);
            match_lengths = cellfun(@length, dictionary(matches));
            [~, idx] = max(match_lengths);
            result(i) = match_locs(idx);
            enc = [enc dec2bin(result(i), num_bits) dictionary{i}(end)];
        end
    end
end

function dictionary = solve(s)
    dictionary = {};
    n = length(s);
    start = 1;
    end_ = 1;
    while end_ <= n
        if ismember(s(start:end_), dictionary)
            end_ = end_ + 1;
            continue
        end
        match = find(strfind(s(start:end_), s(end_)), 1);
        if ~isempty(match) && match > 1
            loc = dec2bin(match-1, ceil(log2(end_-start+1))); % binary
location of repeated digit
            bit = num2str(str2double(s(end_)) - str2double(s(end_-1))); %
change in bit
            dictionary{end+1} = [loc bit]; % add new term to dictionary
            start = end_ + 1;
            end_ = start;
        else
            dictionary{end+1} = s(start:end_);
            start = end_ + 1;
            end_ = start;
        end
    end
end
```