Experiment No.2

Aim: Demonstration of arithmetic coding.

Equipment/Software: Octave

Theory:

Arithmetic coding overcomes the problem of assigning integer codes to the individual symbols by assigning one (normally long) code to the entire input file. The method starts with a certain interval, it reads the input file symbol by symbol, and it uses the probability of each symbol to narrow the interval. Specifying a narrower interval requires more bits, so the number constructed by the algorithm grows continuously. To achieve compression, the algorithm is designed such that a high-probability symbol narrows the interval less than a low-probability symbol, with the result that high-probability symbols contribute fewer bits to the output. An interval can be specified by its lower and upper limits or by one limit and width. The interval [0, 1] can be specified by the two 1-bit numbers 0 and 1. The interval [0.1, 0.512] can be specified by the longer numbers 0.1 and 0.412. The very narrow interval [0.12575, 0.1257586] is specified by the long numbers 0.12575 and 0.0000086.

The output of arithmetic coding is interpreted as a number in the range [0, 1). [The notation [*a*, *b*)

means the range of real numbers from *a* to *b*, including *a* but not including *b*. The range is "closed" at *a* and "open" at *b*.] Thus the code 9746509 is be interpreted as 0.9746509, although the 0. part is not included in the output file.

The main steps of arithmetic coding are summarized below:

1. Start by defining the "current interval" as [0, 1).

2. Repeat the following two steps for each symbol *s* in the input stream:

2.1. Divide the current interval into subintervals whose sizes are proportional to the symbols'

probabilities.

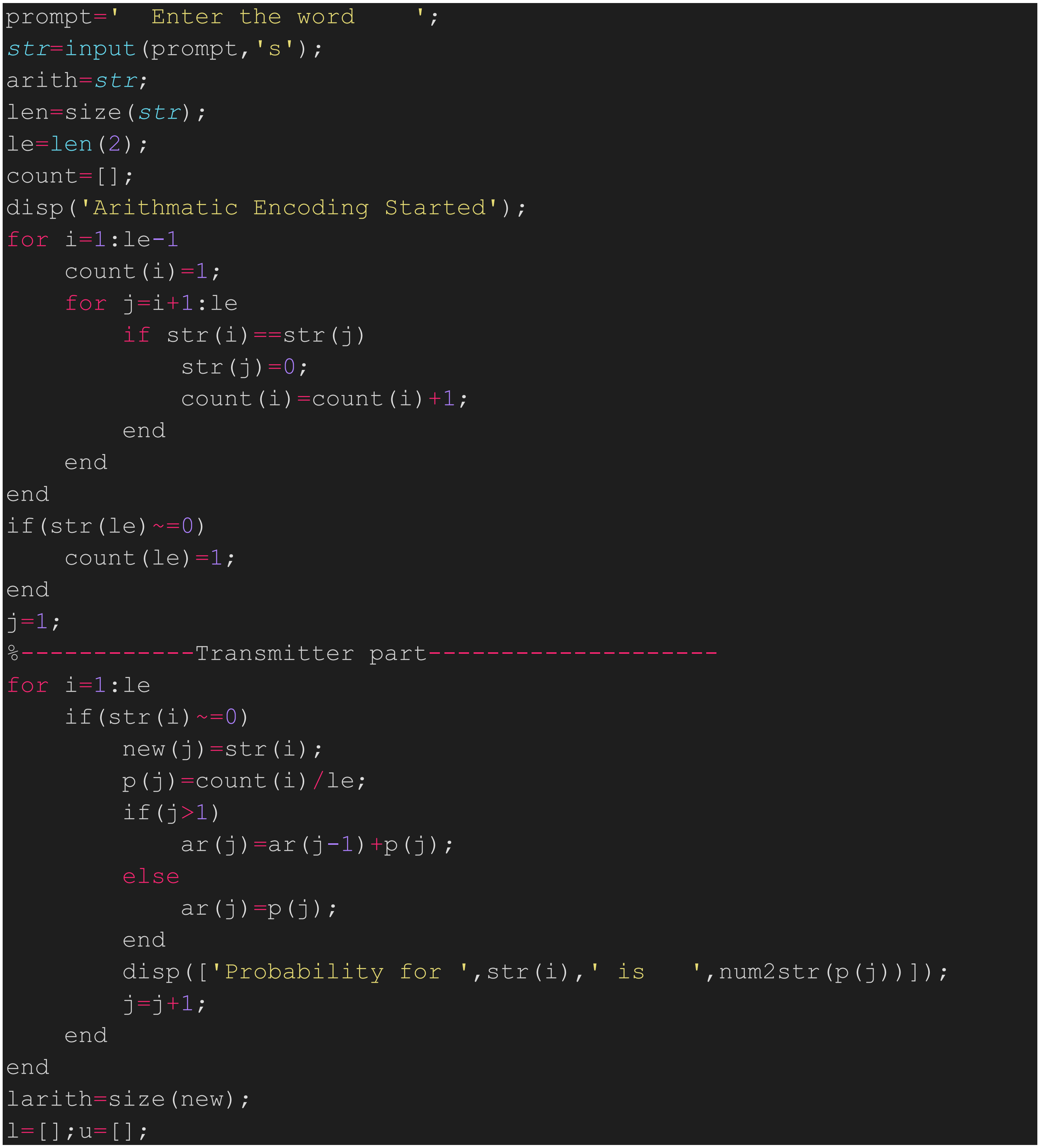
2.2. Select the subintervals for s and define it as the new current interval.

3. When the entire input stream has been processed in this way, the output should be any number that uniquely identifies the current interval (i.e., any number inside the current interval).

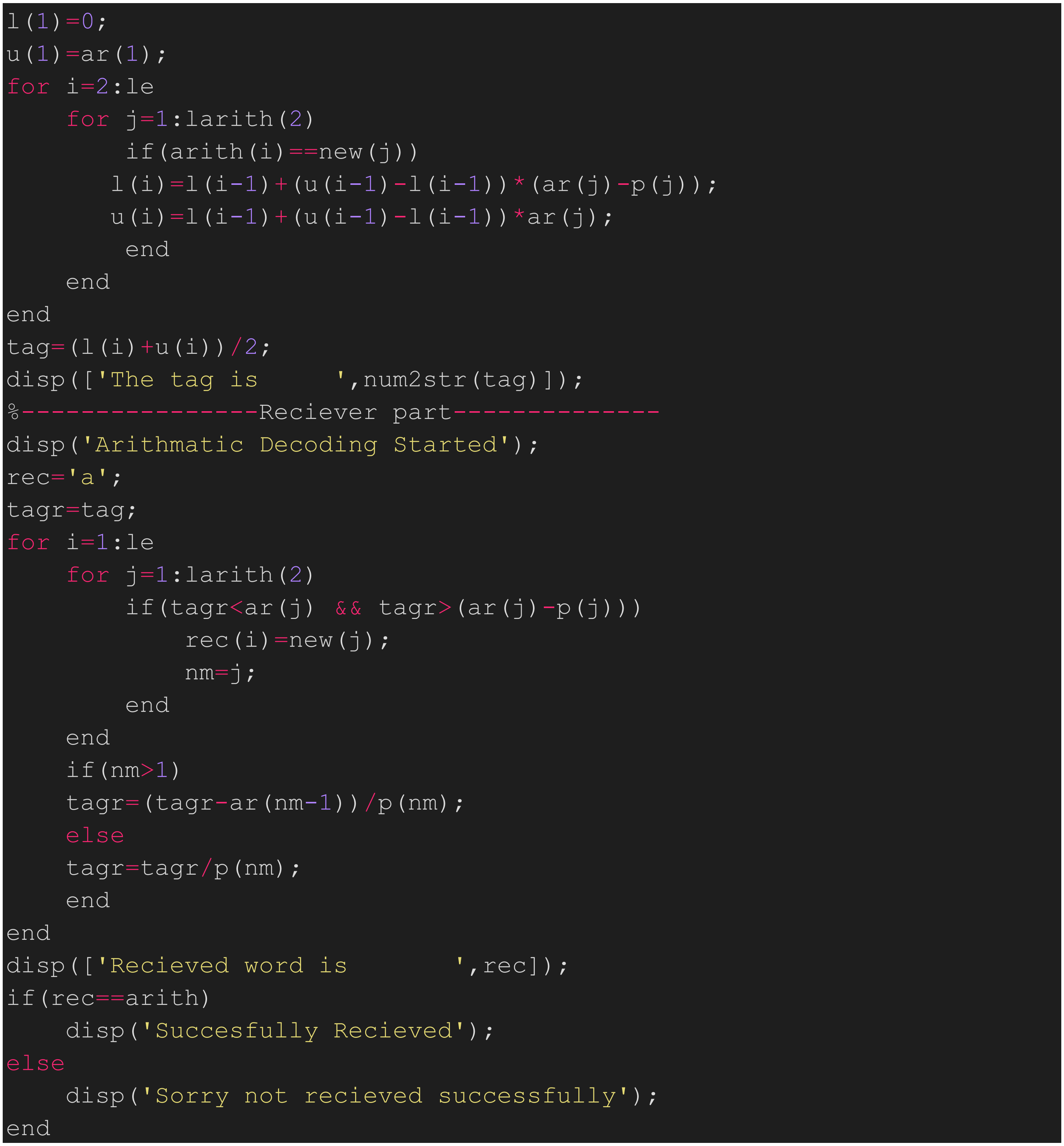
For each symbol processed, the current interval gets smaller, so it takes more bits to express it, but

the point is that the final output is a single number and does not consist of codes for the individual symbols. The average code size can be obtained by dividing the size of the output (in bits) by the
size of the input (in symbols).

Octave Code:

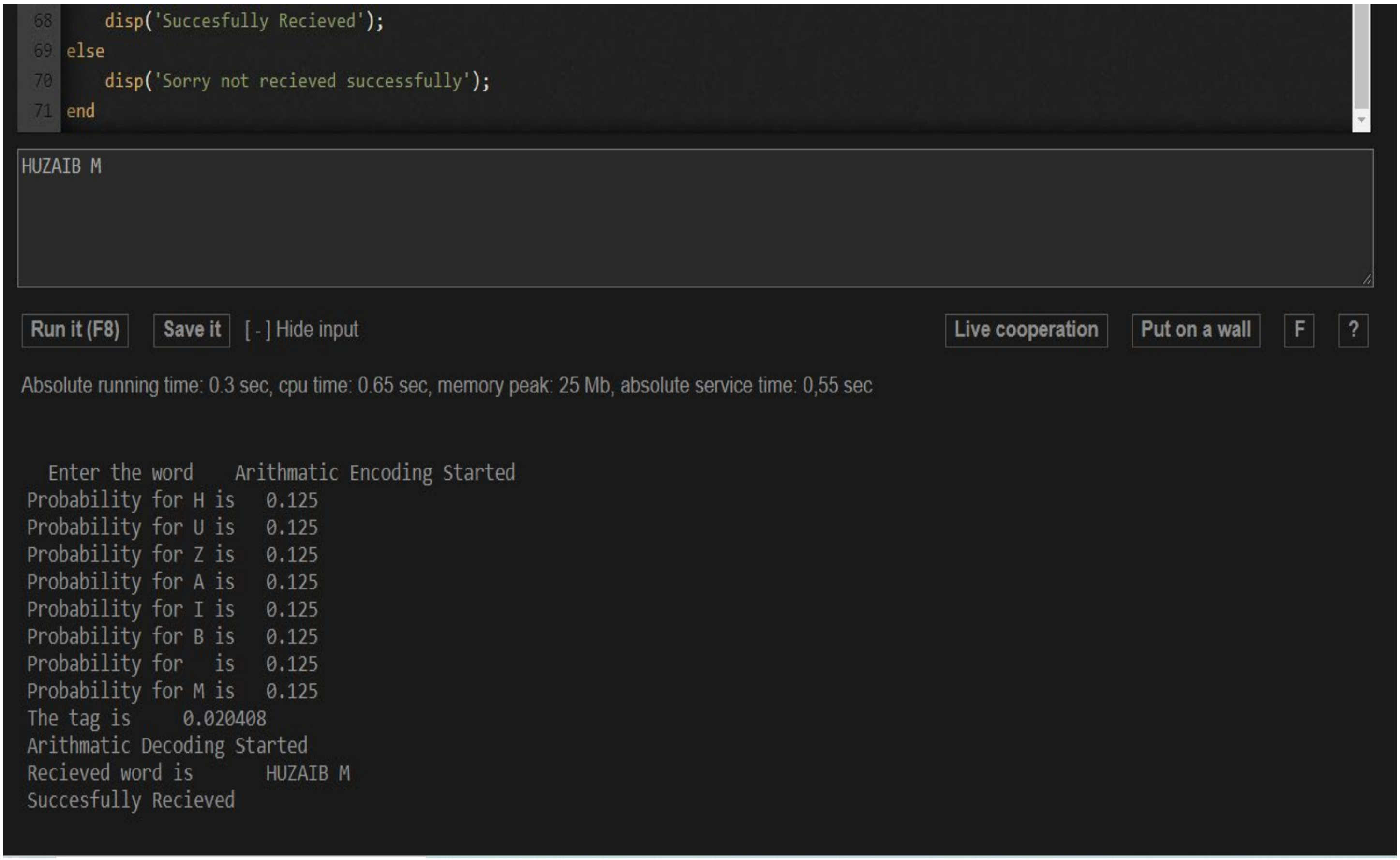


```
prompt='  Enter the word     ';
str=input(prompt,'s');
arith=str;
len=size(str);
le=len(2);
count=[];
disp('Arithmatic Encoding Started');
for i=1:le-1
    count(i)=1;
    for j=i+1:le
        if str(i)==str(j)
            str(j)=0;
            count(i)=count(i)+1;
        end
    end
end
if(str(le)~=0)
    count(le)=1;
end
j=1;
%------------Transmitter part--------------------
for i=1:le
    if(str(i)~=0)
        new(j)=str(i);
        p(j)=count(i)/le;
        if(j>1)
            ar(j)=ar(j-1)+p(j);
        else
            ar(j)=p(j);
        end
        disp(['Probability for ',str(i),' is   ',num2str(p(j))]);
        j=j+1;
    end
end
larith=size(new);
l=[];u=[];
```



```
l(1)=0;
u(1)=ar(1);
for i=2:le
    for j=1:larith(2)
        if(arith(i)==new(j))
       l(i)=l(i-1)+(u(i-1)-l(i-1))*(ar(j)-p(j));
       u(i)=l(i-1)+(u(i-1)-l(i-1))*ar(j);
        end
    end
end
tag=(l(i)+u(i))/2;
disp(['The tag is     ',num2str(tag)]);
%----------------Reciever part--------------
disp('Arithmatic Decoding Started');
rec='a';
tagr=tag;
for i=1:le
    for j=1:larith(2)
        if(tagr<ar(j) && tagr>(ar(j)-p(j)))
            rec(i)=new(j);
            nm=j;
        end
    end
    if(nm>1)
    tagr=(tagr-ar(nm-1))/p(nm);
    else
    tagr=tagr/p(nm);
    end
end
disp(['Recieved word is       ',rec]);
if(rec==arith)
    disp('Succesfully Recieved');
else
    disp('Sorry not recieved successfully');
end
```

Output:



```
    disp('Succesfully Recieved');
else
    disp('Sorry not recieved successfully');
end
```

```
HUZAIB M
```

Absolute running time: 0.3 sec, cpu time: 0.65 sec, memory peak: 25 Mb, absolute service time: 0,55 sec

```
  Enter the word    Arithmatic Encoding Started
Probability for H is   0.125
Probability for U is   0.125
Probability for Z is   0.125
Probability for A is   0.125
Probability for I is   0.125
Probability for B is   0.125
Probability for   is   0.125
Probability for M is   0.125
The tag is     0.020408
Arithmatic Decoding Started
Recieved word is       HUZAIB M
Succesfully Recieved
```