\[ a = 0.03; \quad (* \alpha \) \]

\[ H = -a \cdot \log(2, a) - (1-a) \cdot \log(2, 1-a) \quad (* H(X) \) \]

0.194392

(* Exercise 2.1(c) *)

(* The variable "NN" is the variable N from the Exercise 2.1(c), and "r" is the variable r from Exercise 2.1(c) *)

\[
\text{OptCode}[\text{NN}, r_] := \\
\text{Block}[\{lmax, k, totalmessages, psuccess \text{ \indent \text{\(\star\) declare local variables \)}}], \\
lmax = \text{Floor}[r \cdot H \cdot \text{NN}]; \\
\text{\(\star\) maximal length L of the code C to have rate R(C) \leq r \cdot H \)}
\]

For[\[k = 0; \\
\text{totalmessages} = 0; \\
psuccess = 0 \text{ \(\star\) initialize variables \)}, \\
\text{totalmessages} + \text{Binomial}[\text{NN, k}] \leq 2^lmax \text{ \(\star\) condition to check \)}, \\
\text{totalmessages} = \text{totalmessages} + \text{Binomial}[\text{NN, k}]; \\
psuccess = psuccess + \text{Binomial}[\text{NN, k}] \cdot (a^k) \cdot ((1-a)^{(\text{NN} - k)}); \\
k++ \text{ \(\star\) update variables \}]; \\
psuccess = psuccess + (2^lmax - \text{totalmessages}) \cdot (a^k) \cdot ((1-a)^{(\text{NN} - k)}); \\
\text{\(\star\) add probability of the remaining messages with k=} \\
\text{\(\backslash h(a(k)+1) \) occurences of "1" \} \\
\text{return list: \{error probability, number of 1's in sequences of last column \hat{\text{k}}, code length lmax\} \}) \\
];

\text{OptCode}[10, 0.8] \quad (* Examples *)

\text{OptCode}[10, 1.1] \\
\text{OptCode}[100, 1.1] \\
\text{OptCode}[1000, 1.1] \\
\text{OptCode}[10000, 1.1] \\
\text{OptCode}[100000, 1.1] \quad (* this last call takes a few seconds to compute *)

\{0.239769, 0, 1\} \\
\{0.194155, 0, 2\} \\
\{0.268762, 3, 21\} \\
\{0.189857, 34, 213\} \\
\{0.00950799, 340, 2138\} \\
\{2.776 \times 10^{-12}, 3396, 21383\}
The text contains code and formulas, with some annotations in natural language. Here is a transcription of the content:

```plaintext
TableOpt = Table[N[(OptCode[10^e, r]][[1]], 6],
    {e, {1, 2, 3, 4}}, {r, {0.8, 0.9, 1.0, 1.1, 1.2}}]
(* make the required table, first until NN=10^4 (→faster computation) *)

TableOpt = Table[N[(OptCode[10^e, r]][[1]], {e, {1, 2, 3, 4, 5}},
    {r, {0.8, 0.9, 1.0, 1.1, 1.2}}] (* make the required table *)

TableForm[TableOpt, TableHeadings -> {{"10^1", "10^2", "10^3", "10^4", "10^5"},
    {0.8, 0.9, 1.0, 1.1, 1.2}}] (* print the table *)

(* Also make a list of the values of \hat{k},
which basically describes the code C. Note that 0 ≤ \hat{k} ≤ N. *)
(* One can see that one needs \hat{k}/N > alpha to get a small
error probability (this is more and more true for large N) *)

TableOptHatk = Table[N[(OptCode[10^e, r]][[2]], {e, {1, 2, 3, 4, 5}},
    {r, {0.8, 0.9, 1.0, 1.1, 1.2}}] ; (* make the required table *)

TableForm[TableOptHatk, TableHeadings -> {{"10^1", "10^2", "10^3", "10^4", "10^5"},
    {0.8, 0.9, 1.0, 1.1, 1.2}}] (* print the table *)

(* Exercise 2.2 *)
```
AEPCode[NN_, r_] := Block[{Lmax, kAverage, Deltak,
    TotalMessages, Psuccess (* declare local variables *)},
  Lmax = Floor[r*H*NN]; (* maximal length L of the
code C to have rate R(C) ≤ r*H *)
  kAverage = Round[NN*a]; (* for the cases we consider,
NN*a will be an integer anyway, and go symmetrically to left and right *)
  If[Binomial[NN, kAverage] > 2^Lmax, Return[{{1, -1, Lmax, -1}}];
  (* cannot encode any message *)
  TotalMessages = Binomial[NN, kAverage];
  Psuccess = Binomial[NN, kAverage] * (a^kAverage) * ((1 - a)^(NN - kAverage));
  For[Deltak = 1; (* initialize variable *),
    TotalMessages + Binomial[NN, kAverage + Deltak] + Binomial[NN, kAverage - Deltak] ≤
    2^Lmax (* condition to check *),,
    TotalMessages = TotalMessages + Binomial[NN, kAverage + Deltak] +
    Binomial[NN, kAverage - Deltak];
    Psuccess = Psuccess + Binomial[NN, kAverage + Deltak] * (a^(kAverage + Deltak)) * 
    ((1 - a)^(NN - kAverage - Deltak)) + Binomial[NN, kAverage - Deltak] * 
    (a^(kAverage - Deltak)) * ((1 - a)^(NN - kAverage + Deltak)) * 
    (1 - a)^(NN - kAverage + Deltak)];
  Deltak++ (* update variables *)
];

Return[{{1 - Psuccess, {kAverage - Deltak + 1, kAverage + Deltak - 1}},
  Lmax, N[Log[2, TotalMessages], 6]] (* return a list *)
];

AEPCode[10, 0.8] (* Examples *)
AEPCode[10, 1.1]
AEPCode[100, 1.1]
AEPCode[1000, 1.1]
AEPCode[10000, 1.1]
AEPCode[100000, 1.1] (* this last call can take a few seconds to compute *)
{0.262576, {0, 0}, 1, 0}
{0.262576, {0, 0}, 2, 0}
{0.772526, {3, 3}, 21, 17.3030}
{0.40392, {26, 34}, 213, 210.275}
{0.0175627, {260, 340}, 2138, 2135.26}
{2.810 × 10⁻¹², {2604, 3396}, 21383, 21380.7}
tableAEP = Table[N[(AEPCode[10^e, r]][[1]], 6],
{e, {1, 2, 3, 4, 5}}, {r, {0.8, 0.9, 1.0, 1.1, 1.2}}]
(* make the required table, first until NN=10^4 (→faster computation) *)

TableForm[tableAEP, TableHeadings →
{{"10^1", "10^2", "10^3", "10^4", "10^5"}, {0.8, 0.9, 1.0, 1.1, 1.2}}]
(* print the table *)
(* compare to results from Exercise 1 *)

TableForm[tableOpt, TableHeadings →
{{"10^1", "10^2", "10^3", "10^4", "10^5"}, {0.8, 0.9, 1.0, 1.1, 1.2}}]

(* The following gives the k-range of the maximum typical set
A_\_epsilon((N)) which just fits into the code space of size 2^L *)
(* It is striking to compare (for N≥1000 and r>1) the k-range here to the k-range from Exercise 2.1, see above *)
(* One can see (entry=-1) that for r=0.8 or r=0.9 and N≥100,
the code space is too small to even it
all the "most typical" sequences with k=N*alpha. *)
tableAEPkRange = Table[(AEPCode[10^e, r]][[2]],
{e, {1, 2, 3, 4, 5}}, {r, {0.8, 0.9, 1.0, 1.1, 1.2}}];

TableForm[tableAEPkRange, TableHeadings →
{{"10^1", "10^2", "10^3", "10^4", "10^5"}, {0.8, 0.9, 1.0, 1.1, 1.2}}]
\[ \text{NN} = 10000 \]
\[ r = 0.9 \]
\[ \text{Floor}[\text{NN} \times 0.9] \quad (\text{* largest possible size of the code *}) \]
\[ \text{Log}[2, \text{Binomial}[\text{NN}, \text{NN} \times r]] / \text{N} \]
\[ (\text{* logarithm of the size of the "most typical" set, i.e. with } k = \text{N} \times \text{alpha} \quad (\text{*}) \]
\[ (\text{* If the last number is larger than the size of the code, one cannot encode any typical subset, and the error probability of the code is 1 *}) \]

10000
0.9
1749
1938.5