## Отчет по лабораторной работе №1. Алгоритм РРМА

Мясников Владислав, М4106

Информация о вероятностном распределении символов представлена в виде дерева, где каждый узел соответствует некоторому контексту, который уже встречался при считывании (корень дерева соответствует пустому контексту длины 0). При этом с каждым узлом связан массив частот всех символов (то есть сколько раз тот или иной символ встречался в считанной последовательности после соответствующего контекста).

Помимо дерева узлов, также имеется дополнительное равномерное распределение тех символов, которые еще не встречались в последовательности. С данным распределением также связан массив частот.

Также вводится дополнительный ESC-символ для каждого узла (данный символ имеет частоту, равную единице), так как впоследствии он также будет кодироваться арифметическим кодировщиком.

## Алгоритм работы архиватора:

- 1. Записываем в выходную последовательность первые 8 байт количество кодируемых символов
- 2. Для каждого символа х из исходной последовательности, который нужно закодировать, выполняем следующее:
- 2.1. Кодируем х при помощи арифметического кодировщика:
- ищем наибольшую длину d <= D для текущего контекста, такую что контекст длины d ранее встречался
- если для найденной длины контекста d символ x ни разу не встречался, то кодируем ESC-символ в текущем контексте длины d и декрементируем значение d (итерируемся до тех пор, пока контекст не станет пустым или частота x не станет положительной)
- затем кодируем символ x, используя одно из распределений в зависимости от величины d и частоты символа x (распределение для непустого или пустого контекста, либо равномерное распределение для символов, которые ранее не встречались)
- 2.2. Если х встретился впервые, то сбрасываем частоту, соответствующую этому х в модели равномерного распределения среди символов, которые ранее не встречались 2.3. Инкрементируем частоту для х:
- проходим по дереву контекстных распределений символов, учитывая текущий контекст, и инкрементируем частоту вхождения х
- 2.4. Обновляем текущий контекст:
- контекст представлен в виде массива байт (символов)
- очередной символ добавляем в начало массива, сдвигая все остальные на одну позицию вправо
- если количество символов стало больше максимальной длины D, то удаляем последний символ из массива (он встретился раньше остальных)

## Алгоритм работы деархиватора

- 1. Считываем первые 8 байт входной последовательности и вычисляем n (количество закодированных символов)
- 2. Выполняем п раз декодирование символа х:

- 2.1. Декодируем символ х из входной последовательности, используя арифметический декодер:
- ищем наибольшую длину d <= D для текущего контекста, такую что контекст длины d ранее встречался
- начиная с контекста длины d, на каждом шаге декодируем следующий символ и смотрим: если это ESC-символ, то уменьшаем d на единицу и повторяем процедуру заново для более короткого контекста; в противном случае, возвращаем полученный символ для последующей записи (итерация может продолжаться, пока d >= 0)
- если на предыдущем шаге символ не был получен, используя равномерное распределение для символов, которые ранее не встречались, декодируем x и возвращаем полученный результат
- 2.2. 2.4. Действия точно такие же, как и в архиваторе
- 2.5. Символ х записывается в выходную последовательность

## Результаты работы алгоритма на тестовом стенде

	bib	book1	book2	geo	news	obj1	obj2	paper1	paper2	pic	progc	progl	progp	trans
H(X)	5.20068	4.52715	4.79263	5.64638	5.18963	5.94817	6.26038	4.98298	4.60143	1.21018	5.19902	4.77009	4.86877	5.53278
H(X X)	3.3641	3.58451	3.74521	4.26418	4.09188	3.4635	3.87036	3.64602	3.52231	0.82365	3.6033	3.21156	3.18748	3.35476
H(X XX)	1.88838	2.73167	2.65226	2.36176	2.84378	1.03003	1.74865	2.09146	2.32189	0.68021	1.86652	1.61133	1.34327	1.55852
Bits per symbol	2.44945	2.6084	2.46903	7.22852	3.07648	5.34561	3.41718	2.94607	2.78836	1.01158	3.08399	2.23834	2.21341	2.06286
Archive size (bytes)	34066	250658	188528	92525	145021	14369	105426	19577	28650	64895	15270	20046	13662	24160
All archive size (bytes)	1016853													