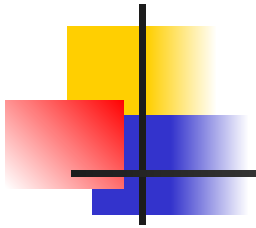
О ПОСТРОЕНИИ ДЕРЕВА ХАФФМАНА

Э. Ю. Джибладзе

Цели и задачи



Цель работы – изучение возможности параллельной реализация алгоритма Хаффмана, основанной на расширении операций матричной алгебры

Задачи

– программная реализация оптимального кода Хаффмана;

– оценка сложности последовательного алгоритма;

– реализация параллельного алгоритма матрично-векторного умножения;

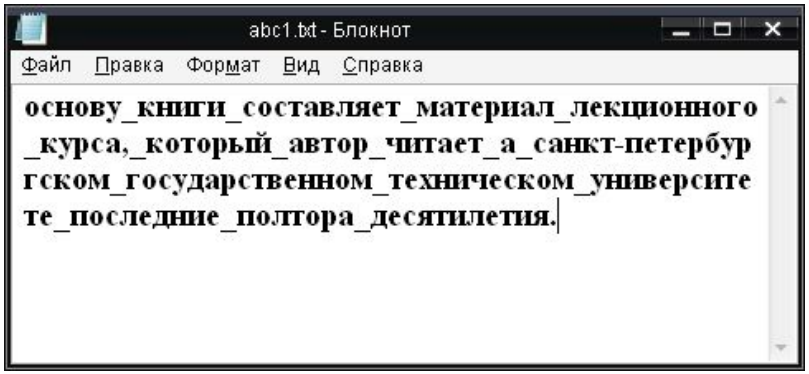
– реализация параллельного алгоритма построения дерева Хаффмана;

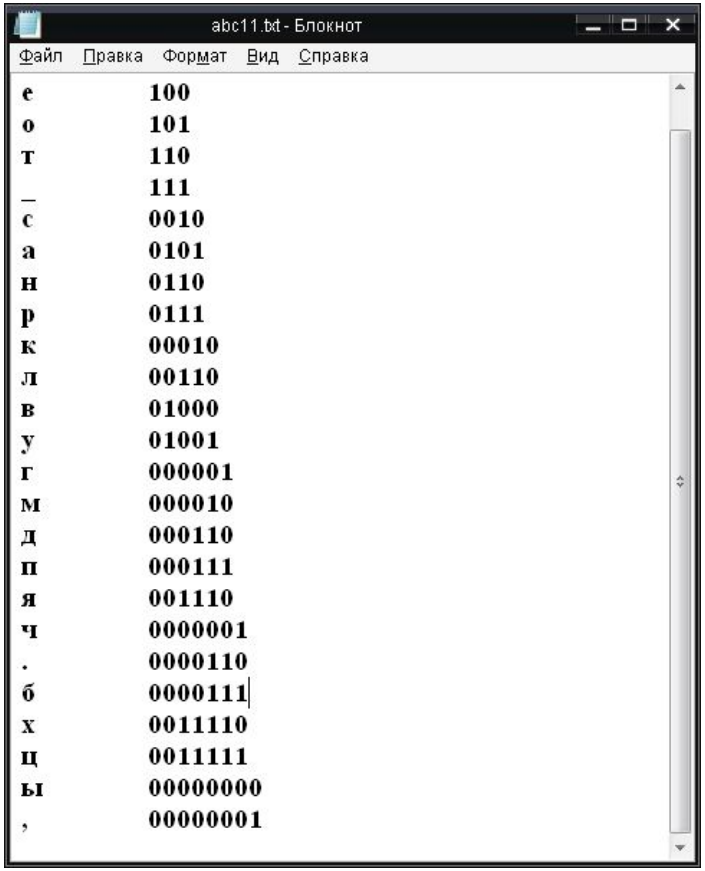
– оценка сложности параллельного алгоритма построения дерева Хаффмана.

Алгоритм построения оптимального кода Хаффмана

1. Символы входного алфавита образуют список из *N* свободных узлов. Вес узла равен либо вероятности, либо количеству вхождений элемента алфавита в сжимаемое сообщение.
2. Выбираются два свободных узла дерева с наименьшими весами.
3. Создается их родитель с весом, равным их суммарному весу.
4. Родитель добавляется в список свободных узлов, а двое его детей удаляются из этого списка.
5. Одной дуге, выходящей из родителя, ставится в соответствие бит *1* , а другой – бит *0*.
6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

Реализация последовательного алгоритма





Оценка сложности последовательного алгоритма

Пусть M – число символов в сообщении, кодируемых по Хаффману и принадлежащих исходному алфавиту из N элементов.

Временные сложности этапов:

1. определение весов дерева по исходному сообщению $T1(1)=O(M)$;

2. выбор двух минимальных весов $T2(1)=O((N^3)/2)$;

3. замена исходных символов кодами $T3(1)=O(M)$ или для адаптивных версий алгоритма: $T3(1)=0$.

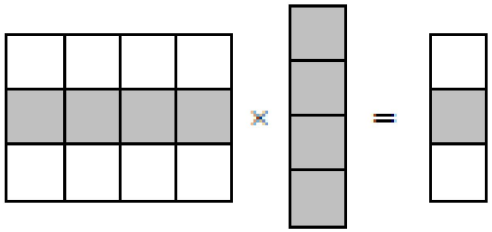
Таким образом, общее время построения дерева и собственно кодирования даже для улучшенных адаптивных версий будет не менее

$$T(1)=T1(1)+T2(1)+T3(1) = M+(N^3)/2.$$

Матрично-векторное умножение

Обычное представление:

$$c_i = (a_i, b) = \sum_{j=1}^{n} a_{i,j} b_j, 1 \le i \le m$$

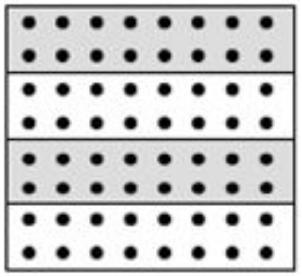


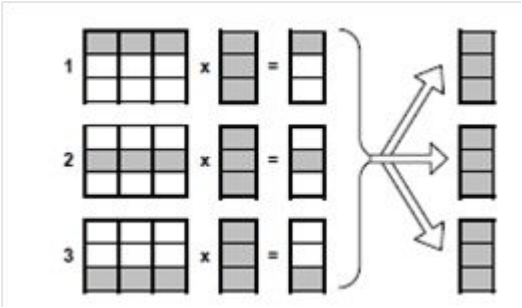
Ленточное разбиение:

Горизонтальное разбиение по строкам:

$$A = (A_0, A_1, \ldots, A_{p-1})^T,\ A_i = (a_{i_0}, a_{i_1}, \ldots, a_{i_{k-1}}),\ i_j = i*k + j,\ 0 \le j < k,\ k = m/p,$$

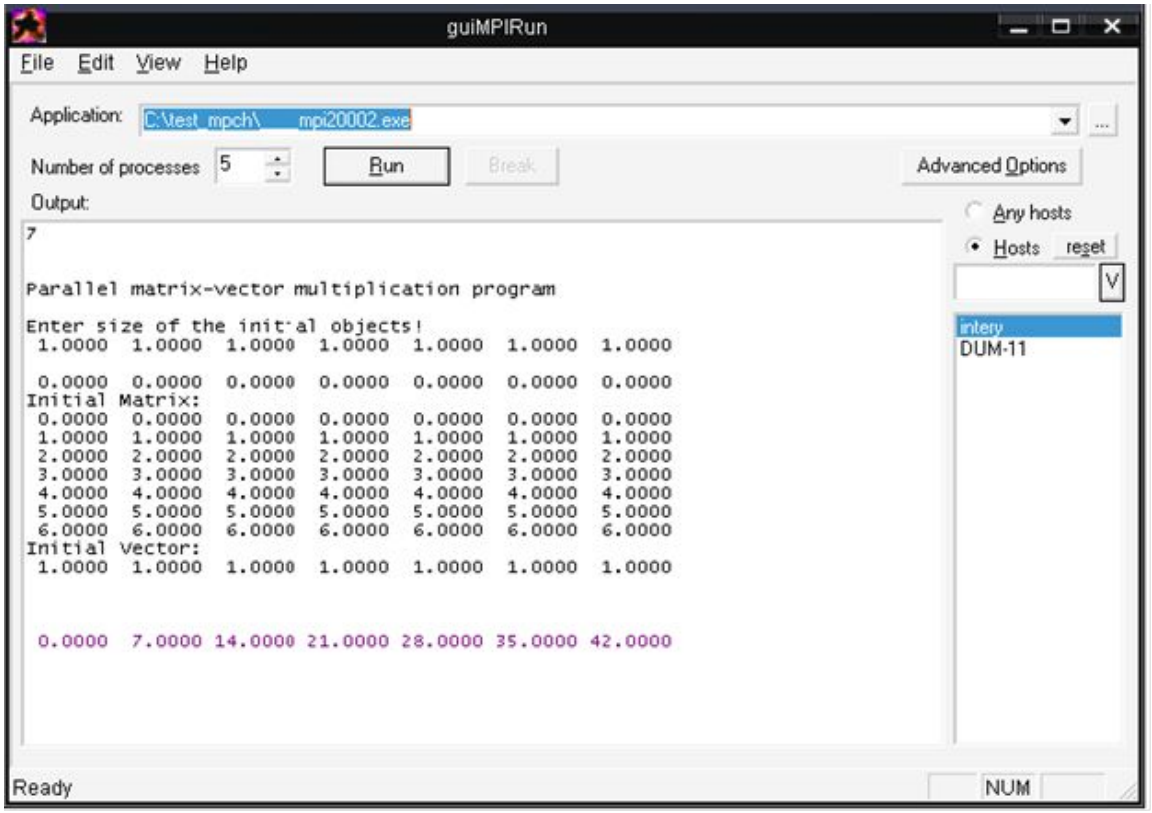
где $a_i = (a_{i_1}, a_{i_2}, \ldots, a_{i_n}),\ 0 \le i < m$ - *i-я* строка матрицы *A* (предполагается, что количество строк *m* кратно числу процессоров *p*, т.е. $m = k \cdot p$)





Результаты работы программы

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 7 \\ 14 \\ 21 \\ 28 \\ 35 \\ 42 \end{pmatrix}$$



Использование матричных операций при построении дерева алгоритма Хаффмана

Алгоритм:

Определить частоты встречаемости символов в сообщении, составляющих его алфавит. N ненулевых элементов алфавита – исходное множество узлов дерева.

- Пока число новых узлов меньше N-1
  - Упорядочить узлы.
  - Добавить новый узел, соответствующий двум минимальным.
  - Для всех символов алфавита добавить очередной разряд в код Хаффмана.
- Конец Пока
- Заменить символы на их коды.

Использование матричных операций при построении дерева алгоритма Хаффмана

Рассмотрим множество, состоящее из элементов 0,1, ω. Пусть *T* – множество, которому принадлежат элементы матрицы, и *P, P1, P2* – предикаты, определенные на *T*. Тогда операции умножения и сложения любых $a, b \in T$ определим следующим образом

$$a \otimes b = \begin{cases} 1, P1 \\ 0, P2 \\ \omega, \neg P1 \wedge \neg P2, \end{cases} \quad (1) \qquad a \oplus b = \begin{cases} b, P \\ a, b = \omega \wedge \neg P. \end{cases} \quad (2)$$

Элементы C_{ij} матрицы $C = A \times B$ будем вычислять по следующим формулам

$$C_{i.j} = A_{i,1} \otimes B_{1,j} \oplus A_{i,2} \otimes B_{2,j} \oplus \ldots \oplus A_{1,n} \otimes B_{n,j} \quad (3)$$

$$C_{i,j} = A_{i,1} \otimes B_{1,j} + A_{i,2} \otimes B_{2,j} + \ldots + A_{i,n} \otimes B_{n,j}, \quad (3')$$

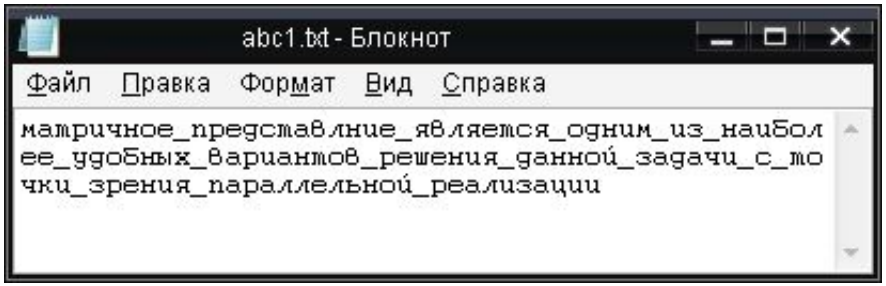
Определение частот встречаемости символов в сообщении

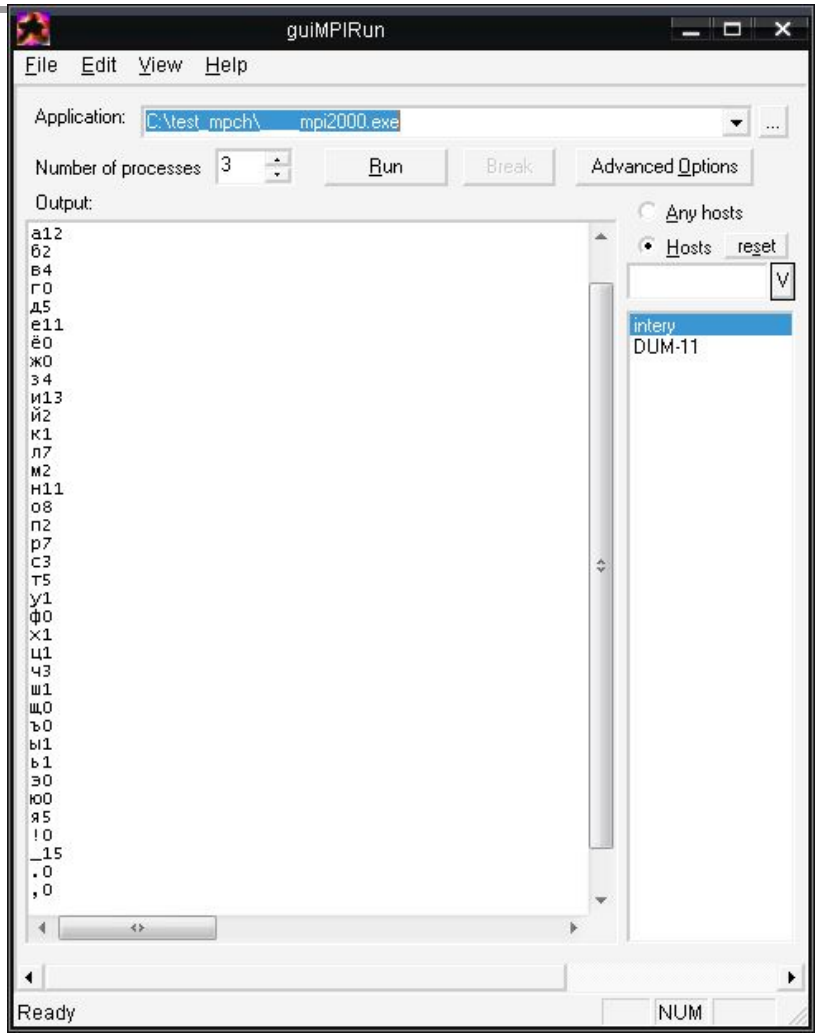
- Представим исходное сообщение, символы которого принадлежат множеству входного алфавита $S = \{s_1, s_2, \ldots s_N\}$, матрицей A размера $L * K$. Пусть X – искомый вектор из *N* чисел, каждое из которых равно числу вхождений соответствующего элемента алфавита в исходное сообщение. Для нахождения его значений образуем новую матрицу B размера $K * N$, каждая строка которой состоит из элементов алфавита S, так что $b_{i,1} = s_1, \; b_{i,2} = s_2, \ldots \; b_{1,N} = s_N$. Выполним умножение матриц $A \times B$, определив в нем операцию умножения компонент

$$a_{i,k} \otimes b_{k,j} = \begin{cases} 1, a_{i,k} = b_{k,j}, \\ 0, a_{i,k} \neq b_{k,j}. \end{cases} \quad (4)$$

- Каждая строка произведения $A \times B$ будет соответствовать числу вхождений соответствующего символа алфавита в строку матрицы исходного сообщения A. Свертка произведения по столбцам (сверху – вниз) позволит получить искомый вектор X. При представлении входной матрицы как вектора размера $1 * K$ свертки не потребуется.

Использование матричных операций при построении дерева алгоритма Хаффмана





Упорядочивание узлов дерева

- Рассмотрим возможность использования введенной операции матричного умножения для упорядочения элементов, составляющих вектор исходных значений A длины *N*. Назовем вектором упорядоченности C последовательность индексов, соответствующую расположению элементов A в порядке их возрастания. Для нахождения значений этого индексного вектора упорядоченности образуем новую матрицу B размера $N * N$, каждая строка которой состоит из элементов исходного вектора A. Выполним умножение матриц $A \times B$, определив в нем только операцию умножения компонент

$$a_{i,k} \otimes b_{k,j} = \begin{cases} 0, (a_{i,k} > b_{k,j}) \vee ((a_{i,k} = b_{k,j}) \wedge (i \le j)), \\ 1, (a_{i,k} < b_{k,j}) \vee ((a_{i,k} = b_{k,j}) \wedge (i > j)) \end{cases} \quad (5)$$

- Чтобы получить индексный вектор упорядоченности $C = [c_1, c_2, c_3, \ldots c_N]$ выполним умножение исходного вектора A на матрицу B, с учетом (5) и при арифметическом сложении:

$C = A \times B$, где $B = \begin{bmatrix} \overline{A} \\ \overline{A} \\ \overline{A} \\ \overline{A} \\ \overline{A} \end{bmatrix}$

Добавление нового узла

- Для выбора двух минимальных узлов и добавления соответствующего им нового узла–родителя преобразуем вектор C, добавив в него незначащие разряды для всех пока не созданных вершин.
- Выполним над C две операции. Первая, матричная, заключается в создании для каждого нового узла вектора кода D длины $2*N-1$, разряды которого соответствуют полному множеству как исходных, так и новых узлов дерева. Вектор D содержит коды *1,0* в разрядах двух минимальных вершин и код *1* в разряде новой родительской вершины. Вторая операция состоит в добавлении к вектору A разряда для значения веса новой вершины, вычисления этого значения и удаления двух минимальных значений. Для нахождения значений D умножим вектор C на матрицу B, значение которой формируются разверткой C. При этом операцию умножения определим как

$$c_{i,k} \otimes b_{k,j} = \begin{cases} 1, ((c_{i,k}=1) \vee (c_{i,k}=2)) \wedge (k \geq j) \vee (k = R) \\ 0, ((c_{i,k}=1) \vee (c_{i,k}=2)) \wedge (k < j) \\ \_, \neg(c_{i,k}=1) \vee \neg(c_{i,k}=2) \end{cases} \quad (6)$$

  где R – номер добавляемой вершины.
- Операцию сложения $S_k \oplus S_{k+1} = c_{i,k} \otimes b_{k,j} \oplus c_{i,k+1} \otimes b_{k+1,j}$ определим в виде

$$S_k \oplus S_{k+1} = \begin{cases} S_{k+1}, (S_{k+1}=1) \vee (S_{k+1}=0) \wedge (S_k \neq 1), \\ \overline{S_k}, S_{k+1} = '\_'. \end{cases}$$

Формирование кодовых разрядов

- При добавлении очередной K-ой вершины разобьем сформированный вектор Dна два: вектор Q, соответствующий элементам исходного алфавита и вектор P, состоящий из элементов родительских разрядов.
- Пусть H – матрица кодов, столбцы которой – это векторы Q, каждый из которых соответствует добавленной вершине. Размер матрицы кодов равен $N * R$, где R – порядковый номер добавляемого узла-родителя. Тогда R-ый вектор-столбец матрицы кодов Хаффмана может быть получен умножением матрицы H на вектор P. При этом операции умножения и сложения определим следующим образом:

$$h_{i,k} \times p_{k,1} = \begin{cases} 1, (h_{i,k} \neq '\_') \wedge (p_{k,1} = 1) \wedge (k \neq R), \\ 0, (h_{i,k} \neq '\_') \wedge (p_{k,1} = 1) \wedge (k = R) \\ \_, (h_{i,k} = '\_') \end{cases} \quad (7)$$

$$S_k \oplus S_{k+1} = \begin{cases} S_{k+1}, (S_{k+1} = 1) \vee (S_{k+1} = 0) \wedge (S_k \neq 1) \\ \overline{S_k}, S_{k+1} = '\_' \end{cases} \quad (8)$$

Суммарная оценка эффективности распараллеливания

Определение частот встречаемости символов в сообщении (общее число сложений $L*K=M$):

$$T_1(p) = \log_2(M).$$

Упорядочение узлов дерева:

$$T_2(p) = \log_2(N)$$

Добавление нового узла:

$$T_3(p) = \log_2(N)$$

Формирование кодовых разрядов:

$$T_4(p) = \log_2(N)$$

Замена символов сообщения кодами:

$$T_5(p) = \log_2(M)$$

Суммарная оценка эффективности распараллеливания:

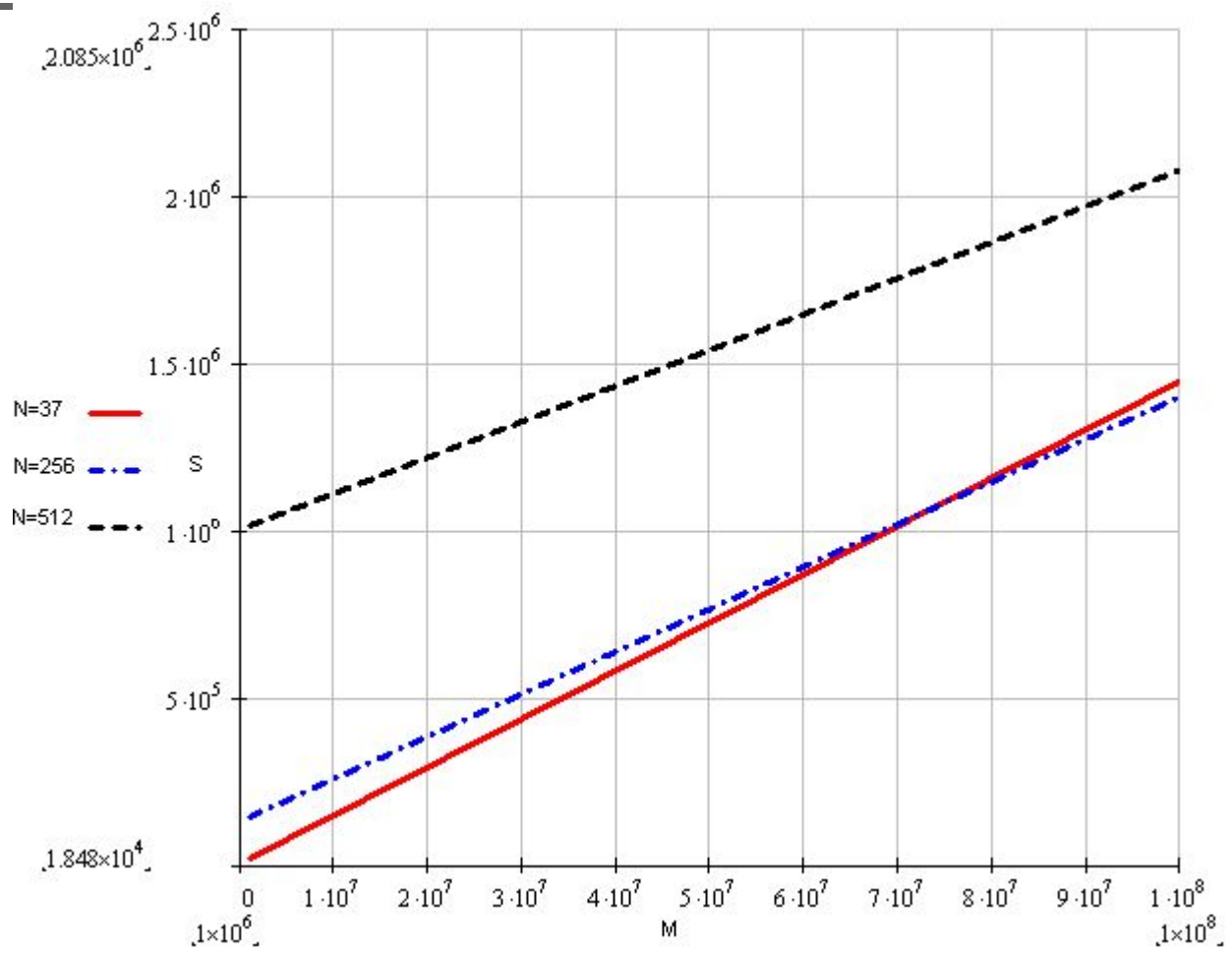
$$T(p) = T_1(p) + (N-1) * (T_2(p) + T_3(p) + T_4(p)) + T_5(p)$$

(9) $$T(p) = 2 * \log_2(M) + 3 * \log_2(N)$$

Суммарная оценка эффективности распараллеливания

$$S_p(n) = \frac{T_1(n)}{T_p(n)}$$

$$S = \frac{M + N^3/2}{2 * \log_2(M) + 3 * \log_2(N)} \quad (10)$$



Пример

Пусть задано следующее множество элементов входного алфавита (N=3) и соответствующие им веса: а–5, б–3, в–7.

Упорядочим узлы:

$$C = [5 \quad 3 \quad 7] \otimes \begin{bmatrix} 5 & 3 & 7 \\ 5 & 3 & 7 \\ 5 & 3 & 7 \end{bmatrix} = [2 \quad 1 \quad 3]$$

где

$$C_{1,1} = 1 + 1 + 0 = 2$$
$$C_{1,2} = 0 + 1 + 0 = 1$$
$$C_{1,3} = 1 + 1 + 1 = 3$$

Добавления нового узла:

$R = 1;$

$$v_1 \quad v_2 \quad a \quad б \quad в$$
$$[\_ \quad \_ \quad 2 \quad 1 \quad 3] \otimes \begin{bmatrix} \_ & \_ & 2 & 1 & 3 \\ \_ & \_ & 2 & 1 & 3 \\ \_ & \_ & 2 & 1 & 3 \\ \_ & \_ & 2 & 1 & 3 \\ \_ & \_ & 2 & 1 & 3 \end{bmatrix};$$

$R = 2:$

$$v_1 \quad v_2 \quad a \quad б \quad в$$
$$[2 \quad \_ \quad \_ \quad \_ \quad 1] \otimes \begin{bmatrix} 2 & \_ & \_ & \_ & 1 \\ 2 & \_ & \_ & \_ & 1 \\ 2 & \_ & \_ & \_ & 1 \\ 2 & \_ & \_ & \_ & 1 \\ 2 & \_ & \_ & \_ & 1 \end{bmatrix}.$$

	v_1			v_2		
	A	C	D	A	C	D
v_1			1	8	2	1
v_2						1
а	5	2	1			
б	3	1	0			
в	7	3		7	1	0

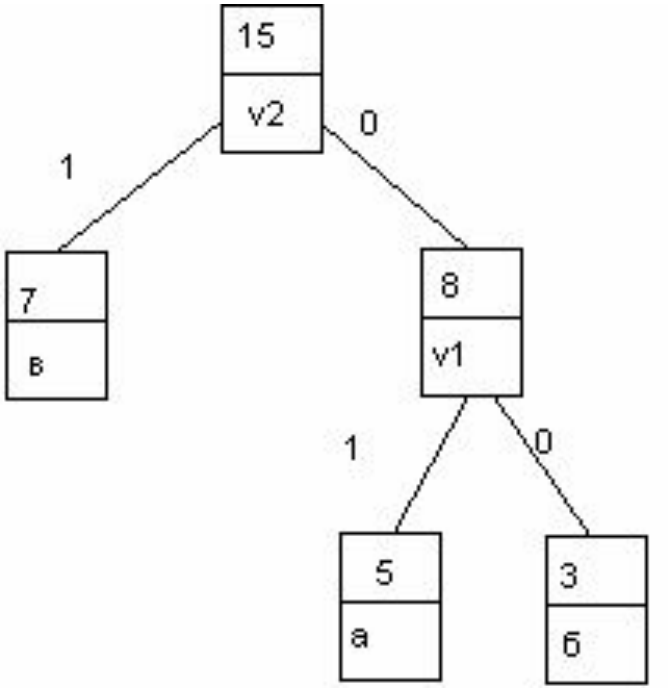
Пример

Формируем кодовые разряды:

$$R = 2$$

$$\begin{matrix} v_1 & v_2 \end{matrix}$$

$$\begin{bmatrix} 1 & \_ \\ 0 & \_ \\ \_ & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus \_ \\ 1 \oplus \_ \\ \_ \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Получили итоговую матрицу:

$$\begin{matrix} а \\ б \\ в \end{matrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \_ & 1 \end{bmatrix}.$$

M=3, N=37 :

$$S = \frac{M + N^3 / 2}{2 * \log_2(M) + 3 * \log_2(N)} = \frac{25329,5}{18,79829} = 1347,437$$

Список используемых источников

1 Алексеев Е. Р. Учимся программировать на Microsoft Visual C++ и Turbo C++ Explorer / Е. Р. Алексеев, под ред. О. В. Чесноковой. – М. : НТ Пресс, 2007 –352 с.

2 Антонов А. С. Параллельное программирование с использованием технологии MPI: учебное пособие / А. С. Антонов. – М. : МГУ, 2004. –71 с.

3 Ахо А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М. : Мир, 1979. – С. 255-283

4 Гергель В. П. Теория и практика параллельных вычислений / В. П. Гергель. – М. : Бином. Лаборатория знаний , 2007. – 424 с.

5 История развития теории сжатия информации [Электронный ресурс]. – Режим доступа: http://compression.ru

6 Новиков Ф. А. Дискретная математика для программистов: учебник для вузов. 2-е изд. / Ф. А. Новиков. – СПб. : Питер, 2005. – С. 171-215

7 Самойлов М. Ю. Использование матричных операций при построении дерева Хаффмана / М. Ю.Самойлов, Т. А. Самойлова. – Смоленск: СГМА Математическая морфология. Электронный математический и медико-биологический журнал. Русская версия 2.0. –Том 2. – Вып.2, 1997. – 246 с.

8 Хокни Р. Параллельные ЭВМ / Р. Хокни, К. Джессхоуп. – М. : Радио и связь, 1986. – С. 253-255, 264-269