Лабораторная работа №1
«Алгоритмы прохода массива»

Основные сведения

Рассмотрим теперь задачу **поиска максимального элемента** в массиве данных, занимающих ячейки начиная от *M[1]* до *M[N]*. Начнем рассмотрение со случая количества процессоров *p=N/2*. Большее количество процессоров не нужно, а в случай *p<N/2* массив данных разбивается на части, в каждой из которых ищется максимум, а затем осуществляется поиск наибольшего из найденных максимальных элементов.

На первом проходе процессор P_i сравнивает значения в ячейках *M[2i]* и *M[2i+1]* и записывает в ячейку *M[i]* большее из них. На втором проходе задействована только половина процессоров, которые аналогично первому проходу обрабатывает ячейки памяти от M[1] до *M[N/2]*, записывая большие элементы пары в ячейки с номерами от *M[1]* до *M[N/4]* и так далее. На каждом проходе длина массива уменьшается вдвое. На последнем проходе будет получено максимальное число.

Другой классической параллельной задачей является **поиск заданного элемента** в массиве. Для простоты будем полагать, что в списке нет дубликатов. Изучение возможности параллельной реализации начнем с простейшего случая равенства количества процессоров количеству элементов в массиве *p=N*. Тогда каждый процессор просто сравнивает искомое значение со значением элемента списка. Если есть совпадение, то процессор записывает номер элемента в специально отведенной ячейке памяти. Если процессоров меньше чем элементов массива, то каждый процессор ищет элемент в своем подмассиве.

Задания

Примечание: Во всех заданиях на вход программе подается имя файла, в первой строке которого через пробел записаны размер массива и количество процессоров, далее в виде строки записаны элементы целочисленного массива. Все программы должны определять общее время решения задачи (без учета времени чтения из файла и записи в файл) на заданном количестве процессоров и на одном процессоре. А также необходимо вычислять коэффициент ускорения.

1-1. Реализуйте программу поиска наименьшего номера элемента массива, равного заданному значению.
1-2. Реализуйте программу поиска наибольшего номера элемента массива, равного заданному значению.
1-3. Реализуйте программу поиска минимального элемента массива.
1-4. Реализуйте программу поиска максимального элемента массива.
1-5. Реализуйте программу подсчета количества элементов массива, равных минимальному элементу массива.
1-6. Реализуйте программу подсчета количества элементов массива, равных максимальному элементу массива.
1-7. Реализуйте программу подсчета количества элементов массива, равных заданному значению.
1-8. Реализуйте программу подсчета количества элементов массива, превышающих заданное значение.
1-9. Реализуйте программу подсчета количества элементов массива, не превышающих заданное значение.
1-10. Реализуйте программу подсчета количества элементов массива, лежащих в заданном интервале.
1-11. Реализуйте программу подсчета количества элементов массива, лежащих вне заданного интервала.

1-12. Реализуйте программу поиска максимального четного элемента массива.
1-13. Реализуйте программу поиска минимального четного элемента массива.
1-14. Реализуйте программу поиска максимального нечетного элемента массива.
1-15. Реализуйте программу поиска минимального нечетного элемента массива.
1-16. Реализуйте программу подсчета количества четных элементов массива, превышающих заданное значение.
1-17. Реализуйте программу подсчета количества нечетных элементов массива, превышающих заданное значение.
1-18. Реализуйте программу подсчета количества четных элементов массива, не превышающих заданное значение.
1-19. Реализуйте программу подсчета количества нечетных элементов массива, не превышающих заданное значение.
1-20. Реализуйте программу подсчета количества четных элементов массива, лежащих в заданном интервале.
1-21. Реализуйте программу подсчета количества нечетных элементов массива, лежащих в заданном интервале.
1-22. Реализуйте программу подсчета количества четных элементов массива, лежащих вне заданного интервала.
1-23. Реализуйте программу подсчета количества нечетных элементов массива, лежащих вне заданного интервала.
1-24. Реализуйте программу подсчета количества элементов массива, не превышающих половину значения максимального элемента массива.
1-25. Реализуйте программу подсчета количества элементов массива, превышающих половину значения максимального элемента массива.

Лабораторная работа №2
«Параллельный префикс»

Основные сведения

Вычисление префиксной суммы или просто префикса – это одна из важных абстрактных задач, к которой сводится большое количество алгоритмов. Пусть $X=\{x[1],x[2],...,x[N]\}$ - множество элементов из Y, а $\oplus$ - бинарный ассоциативная операция, замкнутая относительно Y. Результат операции $\pi_N=x[1]\oplus x[2]\oplus...\oplus x[N]$ называют N-м префиксом. Вычисление всех N префиксов:

$\pi_1=x[1]$,
$\pi_2=x[1]\oplus x[2]$,
$\pi_3=x[1]\oplus x[2]\oplus x[3]$,
...
$\pi_N=x[1]\oplus x[2]\oplus x[3]\oplus ... \oplus x[N]$

называют параллельным префиксным вычислением. В качестве примера операции $\oplus$ можно рассмотреть обычные умножение или сложение. При использовании последовательного алгоритма, задача нахождения N префиксов сводится к последовательному вычислению $N-1$ суммы двух слагаемых.

Параллельное вычисление префиксных сумм может быть организовано при использовании следующего алгоритма для количества процессоров $p=N$.

1) Для всех $i=2,...,N$ каждый процессор вычисляет сумму своего элемента и элемента процессора с номером на единицу меньше. $P_i=x[i]+x[i-1]$.

2) Для всех $i=3,...,N$ каждый процессор вычисляет сумму своего элемента и величины полученной процессором с номером на 2 меньше на предыдущем шаге.

$P_i=x[i]+x[i-1]+x[i-2]+x[i-3]$.

...

k) Для всех $i=k+1,...,N$ каждый процессор вычисляет сумму своего элемента и величины полученной процессором с номером на k меньше на предыдущем шаге.

$P_i=x[i]+x[i-1]+...+x[i-2^k-1]$.

Если процессоров меньше чем элементов массива, то каждый процессор на первом этапе ищет префиксные суммы в своем подмассиве.

Во всех заданиях частичной суммой элементов массива обозначена следующая величина $S[k,m]=x[k]+x[k+1]+...+x[m]$.

Задания

Примечание: Во всех задачах массив считывается из файла и результат записывается в файл. Все программы должны определять общее время решения задачи (без учета времени чтения из файла и записи в файл) на заданном количестве процессоров и на одном процессоре. А также необходимо вычислять коэффициент ускорения.

2-1. Используя параллельный префикс, реализуйте алгоритм вычисления частичных сумм S[1,1], S[1,2], S[1,3], ..., S[1,1000] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.

2-2. Используя параллельный префикс, реализуйте алгоритм вычисления частичных сумм S[1000,1000], S[999,1000], S[998,1000], ..., S[1,1000] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.

2-3. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[500,500], S[499,501], S[498,502], ..., S[1,1000] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.

2-4. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,1], S[1,3], S[1,5], S[1,7] ..., S[1,999] массива из 1000 вещественных переменных для

случая двух процессоров с общей памятью.
2-5. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,2], S[1,4], S[1,6], ..., S[1,1000] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.
2-6. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1000,1000], S[998,1000], S[996,1000], ..., S[2,1000] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.
2-7. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[999,1000], S[997,1000], S[995,1000], ..., S[1,1000] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.
2-8. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[500,500], S[498,502], S[496,504], ..., S[2,998] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.
2-9. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[499,501], S[497,503], S[495,505], ..., S[1,999] массива из 1000 вещественных переменных для случая двух процессоров с общей памятью.
2-10. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[499,501], S[497,503], S[495,505], ..., S[1,999] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-11. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,1], S[1,2], S[1,3], ..., S[1,1000] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-12. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1000,1000], S[999,1000], S[998,1000], ..., S[1,1000] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-13. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[500,500], S[499,501], S[498,502], ..., S[1,1000] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-14. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,1], S[1,3], S[1,5], S[1,7] ..., S[1,999] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-15. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,2], S[1,4], S[1,6], ..., S[1,1000] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-16. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1000,1000], S[998,1000], S[996,1000], ..., S[2,1000] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-17. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[999,1000], S[997,1000], S[995,1000], ..., S[1,1000] массива из 1000 вещественных переменных для случая трех процессоров с общей памятью.
2-18. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[500,500], S[498,502], S[496,504], ..., S[2,998] массива из 1000 вещественных переменных для случая трех х процессоров с общей памятью.
2-19. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[499,501], S[497,503], S[495,505], ..., S[1,999] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.
2-20. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,1], S[1,3], S[1,5], S[1,7] ..., S[1,999] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.
2-15. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1,2], S[1,4], S[1,6], ..., S[1,1000] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.

2-16. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[1000,1000], S[998,1000], S[996,1000], ..., S[2,1000] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.
2-17. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[999,1000], S[997,1000], S[995,1000], ..., S[1,1000] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.
2-18. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[500,500], S[498,502], S[496,504], ..., S[2,998] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.
2-25. Используя параллельный префикс реализуйте алгоритм вычисления частичных сумм S[500,500], S[499,501], S[498,502], ..., S[1,1000] массива из 1000 вещественных переменных для случая четырех процессоров с общей памятью.

Лабораторная работа №3
«Распараллеливание матричных операций»
Основные сведения

Рассмотрим одну из самых распространенных операций умножения матриц. Пусть необходимо найти матрицу произведение матрицы A размером $n_1 \times n_2$ на матрицу B размером $n_2 \times n_3$. Результирующая матрица C=A·B будет иметь размеры $n_1 \times n_3$. Элементы же матрицы будут находится по формуле

$$C_{ij} = \sum_{k=1}^{n2} A_{ik} B_{kj} .$$

Разрежем предварительно матрицу *A* на *p* горизонтальных полос. Для равномерной загрузки процессоров необходимо, чтобы полосы были одинаковой ширины, но это достижимо только при делимости n_1 на *p*. В противном случае первые $n_1 \bmod p$ полос будут включать в себя $[n_1/p]+1$ строк, остальные - по $[n_1/p]$ строк. Аналогично разрезаем матрицу *B* на *p* вертикальных полос. Введем нумерацию процессоров $P_1,...,P_p$. Введем также нумерацию полос матрицы *A*: $A_1,...,A_p$, и полос матрицы *B*: $B_1,...,B_p$. Рассмотрим теперь непосредственно алгоритм вычисления произведения матриц:

1) В память процессора P_i загружается полосы A_i и B_i ($i=1,2,...,p$).
2) Каждый процессор P_i вычисляет подматрицу $C_{ii}=A_i \cdot B_i$ ($i=1,2,...,p$).

For (k=0; k<p; k++) {

3) Процессор P_i осуществляет передачу, хранящейся у него вертикальной полосы $B_{(i+k) \bmod p}$, соседнему процессору ($i=1,2,...,p$).
4) Каждый процессор P_i вычисляет подматрицу $C_{i,(i+k) \bmod p}=Ai \cdot B_{(i+k) \bmod p}$.

}

5) Матрица *C* собирается из подматриц на процессоре P_1.

В представленном алгоритме пересылка вертикальных полос B_i может быть заменена пересылкой горизонтальных полос A_i. Перед шагом 5 матрица *C* будет разрезана на p^2 подматриц.

Рассмотрим еще один классический алгоритм поиска решения системы линейных уравнений методом Гаусса. Пусть требуется найти решение системы линейных алгебраических уравнений:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1, \\ a_{21}x_2 + a_{22}x_2 + ... + a_{2n}x_n &= b_2, \\ . . . . . & \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n &= b_n. \end{aligned}$$

Метод Гаусса основан на последовательном исключении неизвестных. В матричной форме задача имеет вид

$$A \cdot x = b$$

Рассмотрим случай сильно связанной системы из $p=n$ процессоров. Пусть *i*-ая строка матрицы *A* хранится в процессоре с номером *i*. На первом шаге первая строка рассылается всем процессорам, после чего процессоры могут параллельно производить следующие вычисления:

$$l_{i1}=a_{i1}/a_{11},\ a_{ij}=a_{ij}-l_{i1}a_{1j},\ j=2,3,\ ...,n.$$

На втором шаге вторая строка приведенной матрицы рассылается из процессора P_2 процессорам $P_3,...,P_n$, после чего все процессоры кроме первого и второго осуществляют параллельную обработку своих строк.

Такой алгоритм является не очень удачным, так как он требует большого количества обмена данных и на каждом следующем этапе на один увеличивается число простаивающих процессоров.

Рассмотрим случай, когда количество процессоров меньше числа строк матрицы ($p<N$), тогда возможны два подхода к распределению данных по процессорам. Начнем со **слоисто блочной** схемы хранения строк. Разрежем исходную матрицу коэффициентов *A* и

вектор правых частей b на горизонтальные полосы шириной $[n/p]$ - $A^{(i)}$ и $b^{(i)}$. Каждая полоса загружается в соответствующий процессор: $A^{(i)}$ и $b^{(i)}$ в P_i. При прямом ходе матрица приводится к треугольному виду последовательно по процессорам. Вначале к треугольному виду приводятся строки в первом процессоре, при этом первый процессор последовательно, строка за строкой, по мере обработки, передает свои сроки остальным процессорам. Затем к треугольному виду приводятся строки во втором процессоре, передавая свои строки остальным процессорам и т.д. Аналогично, последовательно по процессорам, начиная с последнего по номеру компьютера, осуществляется обратный ход.

Другой подход получил название **слоисто циклической** схемы хранения строк. Распределим исходную матрицу коэффициентов по процессорам циклическими горизонтальными полосами с шириной полосы в одну строку. То есть процессор с номером k получает строки с номерами $k+p \cdot i,\ i=0,1,...,[n/p]$. Строку, которая вычитается из всех остальных строк (после предварительного деления на нужные коэффициенты), назовем текущей строкой. Алгоритм прямого хода заключается в следующем. Сначала текущей строкой является строка с индексом 1 в процессоре 1, затем строка с индексом 1 в процессоре 2 и т. д., и наконец, строка с индексом 1 в процессоре P_p. После чего цикл по процессорам повторяется и текущей строкой становится строка с индексом 2 в процессоре 1, затем строка с индексом 2 в процессоре 2 и т. д. Аналогично, последовательно по узлам, начиная с последнего по номеру процессора, осуществляется обратный ход.

Задания

Примечание: Во всех заданиях матрицы считываются из файла. Все программы должны определять общее время решения задачи (без учета времени чтения из файла и записи в файл) на заданном количестве процессоров и на одном процессоре. А также необходимо вычислять коэффициент ускорения.

3-1. Реализуйте параллельный алгоритм умножения матриц размером 1000×1000 для случая двух процессоров с общей памятью.
3-2. Реализуйте параллельный алгоритм вычисления определителя матрицы размером 1000×1000 разложением по заданной строке для случая двух процессоров с общей памятью.
3-3. Реализуйте параллельный алгоритм решения системы из 1000 линейных алгебраических уравнений с 1000 неизвестных методом Гаусса для случая двух процессоров с общей памятью.
3-4. Реализуйте параллельный алгоритм поиска определителя матрицы размером 1000×1000 методом Гаусса для случая двух процессоров с общей памятью.
3-5. Реализуйте параллельный алгоритм сложения матриц размером 1000×1000 для случая двух процессоров с общей памятью.
3-6. Реализуйте параллельный алгоритм поиска собственных значений матрицы размером 1000×1000 для случая двух процессоров с общей памятью.
3-7. Реализуйте параллельный алгоритм вычисления определителя матрицы размером 1000×1000 разложением по заданному столбцу для случая двух процессоров с общей памятью.
3-8. Реализуйте параллельный алгоритм умножения матрицы размером 1000×1000 на вектор-строку для случая двух процессоров с общей памятью.
3-9. Реализуйте параллельный алгоритм умножения матрицы размером 1000×1000 на вектор-столбец для случая двух процессоров с общей памятью.
3-10. Реализуйте параллельный алгоритм умножения матрицы размером 1000×1000 на вектор-столбец для случая трех процессоров с общей памятью.
3-11. Реализуйте параллельный алгоритм умножения матриц размером 1000×1000 для случая трех процессоров с общей памятью.
3-12. Реализуйте параллельный алгоритм вычисления определителя матрицы размером 1000×1000 разложением по заданной строке для случая трех процессоров с общей памятью.

3-13. Реализуйте параллельный алгоритм решения системы из 1000 линейных алгебраических уравнений с 1000 неизвестных методом Гаусса для случая трех процессоров с общей памятью.
3-14. Реализуйте параллельный алгоритм поиска определителя матрицы размером 1000×1000 методом Гаусса для случая трех процессоров с общей памятью.
3-15. Реализуйте параллельный алгоритм сложения матриц размером 1000×1000 для случая трех процессоров с общей памятью.
3-16. Реализуйте параллельный алгоритм поиска собственных значений матрицы размером 1000×1000 для случая трех процессоров с общей памятью.
3-17. Реализуйте параллельный алгоритм вычисления определителя матрицы размером 1000×1000 разложением по заданному столбцу для случая трех процессоров с общей памятью.
3-18. Реализуйте параллельный алгоритм умножения матрицы размером 1000×1000 на вектор-строку для случая трех процессоров с общей памятью.
3-19. Реализуйте параллельный алгоритм умножения матрицы размером 1000×1000 на вектор-столбец для случая четырех процессоров с общей памятью.
3-20. Реализуйте параллельный алгоритм умножения матриц размером 1000×1000 для случая четырех процессоров с общей памятью.
3-21. Реализуйте параллельный алгоритм вычисления определителя матрицы размером 1000×1000 разложением по заданной строке для случая четырех процессоров с общей памятью.
3-22. Реализуйте параллельный алгоритм решения системы из 1000 линейных алгебраических уравнений с 1000 неизвестных методом Гаусса для случая четырех процессоров с общей памятью.
3-23. Реализуйте параллельный алгоритм поиска определителя матрицы размером 1000×1000 методом Гаусса для случая четырех процессоров с общей памятью.
3-24. Реализуйте параллельный алгоритм сложения матриц размером 1000×1000 для случая четырех процессоров с общей памятью.
3-25. Реализуйте параллельный алгоритм поиска собственных значений матрицы размером 1000×1000 для случая четырех процессоров с общей памятью.

Лабораторная работа №4
«Расщепление последовательности ГПСП»

Основные сведения

1. Генераторы ПСП

1.1 Линейный конгруэнтный генератор (LCD)

Псевдослучайная последовательность целых чисел определяется начальным значением U_0 и рекуррентным соотношением

$$U_{n+1}=aU_n+c \bmod m,$$

где $m>0$ модуль последовательности, a - множитель ($0<a<m$) и c - аддитивная константа.

1.2. Прямой инверсный конгруэнтный генератор (EICG)

Псевдослучайная последовательность формируется по рекуррентной формуле

$$U_n = \overline{an + b} \mod m.$$

Здесь a и b – константы, а чертой сверху обозначено число, обратное к данному по *mod m*.

Основной вычислительной трудностью для генераторов данного типа является нахождение обратного элемента. В общем случае обращение числа по *mod m* требует машинного времени порядка $O(\log m)$. Используем алгоритм Евклида нахождения наибольшего общего делителя (n,m) чисел n и m Раскручивая алгоритм Евклида в обратную сторону мы можем найти такие целые числа x и y, что будет верно равенство:

$$(n,m)=xn+ym.$$

Полагая, что m - простое число, получаем:

$$xn+ym=1 \bmod m \Rightarrow x_n=1 \bmod m \Rightarrow x=\overline{n} \bmod m.$$

2. Методы расщепления последовательности

2.1 Блочное разбиение

Пользователь выбирает размер блока B, который задает число членов подпоследова-тельности. Если общая последовательность имеет вид U_0, U_1, ..., то на i-ом процессоре вычисляются члены U_{iB}, U_{iB+1}, ... Размер блока может быть выбран произвольно. Однако для эффективного использования вычислительной системы требуется равномерная загрузка процессоров, что приводит к однозначному выбору размера блока B. Основная проблема при таком способе распараллеливания состоит в вычислении зерен подпоследовательностей.

2.1.1 Линейный конгруэнтный генератор

Зерна генераторов отдельных процессоров могут быть вычислены по формуле

$$U_n = a^n U_0 + \frac{a^n - 1}{a - 1} c \mod m.$$

2.1.2 Прямой обратный конгруэнтный генератор

Блочное разбиение EICG на подпоследовательности не встречает никаких трудностей в силу возможности вычисления любого члена последовательности по его номеру.

2.2 «Прыжок лягушки»

Этот алгоритм также разбивает последовательность на подпоследовательности, но более гибко чем блочное разбиение. А именно процессор с номером i вырабатывает подпоследовательность U_i, U_{i+p}, U_{i+2p}, ..., где p - общее количество процессоров. Следует отметить, что такой способ распараллеливания требует не только вычисления зерен подпоследовательностей, но и получения дополнительных рекуррентных соотношений, позволяющих выполнять «прыжок», что не всегда возможно.

Введем функцию $f(x)$, определяющую рекуррентное соотношение для генератора случайных чисел $U_n=f(U_{n-1})$. Для удобства введем обозначение для суперпозиции k функций:

$$f^k(x)= f(f(f...f(x)...)) \ (k \text{ раз}).$$

Тогда k-ый член последовательности случайных чисел находится как $U_k=f^k(x)$.

Рассмотрим расщепление последовательности на примере двух процессоров ($k=2$):

P_0: $U_0, U_2, U_4,...$

P_1: $U_1, U_3, U_5,...$

Другими словами процессор P_0 генерирует члены последовательности начиная с U_0, а процессор P_1 - члены последовательности начиная с U_1, при этом оба используют рекуррентное соотношение $U_n=F(U_{n-1})$, где $F(x)=f^2(x)$.

2.2.1 Линейный конгруэнтный генератор

Если функция имеет вид $f(x)=ax+c \ mod \ m$, то для любого целого $k>0$ суперпозиция k таких функций определяется соотношением

$$f^k(x) = a^k x + \frac{a^k - 1}{a - 1} c \quad \text{mod} \quad m$$

2.1.2 Прямой обратный конгруэнтный генератор

Не требует вычисления функции $F(x)$, так как возможно прямое вычисление членов последовательности.

Задания

Примечание: Во всех заданиях необходимо реализовать параллельный генератор псевдослучайных чисел. Далее на базе полученного генератора псевдослучайных чисел необходимо реализовать поточное шифрование, путем побитового «ИСКЛЮЧАЮЩЕГО ИЛИ» псевдослучайной последовательности с исходным текстом. Процесс шифрования также необходимо распараллелить на два процессора. Все программы должны определять общее время решения задачи (без учета времени чтения из файла и записи в файл) на заданном количестве процессоров и на одном процессоре. А также необходимо вычислять коэффициент ускорения.

4-1. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-2. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом leapfrog. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-3. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом блочного разбиения. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-4. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом блочного разбиения. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-5. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-6. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом leapfrog. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-7. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом блочного разбиения. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-8. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на два процессора методом блочного разбиения. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-9. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом leapfrog. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-10. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом блочного разбиения. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-11. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-12. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом leapfrog. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-13. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом блочного разбиения. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-14. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом блочного разбиения. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-15. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-16. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом leapfrog. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-17. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом блочного разбиения. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-18. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на три процессора методом блочного разбиения. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-19. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом блочного разбиения. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-20. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-21. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-22. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом leapfrog. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-23. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом блочного разбиения. В качестве модуля генератора m сгенерируйте простое трехзначное число.

4-24. Реализуйте линейный конгруэнтный генератор псевдослучайных чисел, распараллелив его на четыре процессора методом блочного разбиения. В качестве модуля генератора используйте число 2^w, где число w выбирается пользователем.

4-25. Реализуйте прямой инверсный конгруэнтный генератор псевдослучайных чисел, распараллелив его на пять процессоров методом leapfrog. В качестве модуля генератора m сгенерируйте простое трехзначное число.

Лабораторная работа №5
«Параметризация ГПСП»
Основные сведения

Задания

Примечание: Во всех заданиях необходимо провести спектральный тест распределения байтов в конечной случайной последовательности. Все программы должны определять общее время решения задачи (без учета времени чтения из файла и записи в файл) на заданном количестве процессоров и на одном процессоре. А также необходимо вычислять коэффициент ускорения.

5-1. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех линейных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.
5-2. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех линейных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.
5-3. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.
5-4. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.
5-5. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех прямых обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.
5-6. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех прямых обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.
5-7. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех аддитивных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.
5-8. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех аддитивных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.
5-9. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех мультипликативных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов

используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-10. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех мультипликативных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-11. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех линейных конгруэнтных генераторов. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-12. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех линейных конгруэнтных генераторов. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-13. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-14. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-15. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех прямых обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-16. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех прямых обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-17. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех аддитивных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-18. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех аддитивных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-19. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех мультипликативных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой

конкатенацией последовательностей генераторов.

5-20. Реализуйте параллельный генератор псевдослучайных чисел, на базе четырех мультипликативных генераторов Фибоначчи с запаздыванием. Для выбора зерен генераторов используйте параметризацию. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-21. Реализуйте параллельный генератор псевдослучайных чисел, на базе двух линейных конгруэнтных генераторов и двух генераторов аддитивных генераторов Фибоначчи. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-22. Реализуйте параллельный генератор псевдослучайных чисел, на базе двух линейных конгруэнтных генераторов и двух генераторов аддитивных генераторов Фибоначчи. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-23. Реализуйте параллельный генератор псевдослучайных чисел, на базе двух линейных конгруэнтных генераторов и двух обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.

5-24. Реализуйте параллельный генератор псевдослучайных чисел, на базе двух линейных конгруэнтных генераторов и двух обратных конгруэнтных генераторов. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите слиянием последовательностей генераторов.

5-25. Реализуйте параллельный генератор псевдослучайных чисел, на базе трех линейных конгруэнтных генераторов и одного обратного конгруэнтного генератора. Для выбора зерен генераторов используйте дерево Лехмера. В качестве модулей линейных генераторов используйте простые трехзначные числа. Конечную случайную последовательность получите простой конкатенацией последовательностей генераторов.