Поразрядная сортировка

Поразрядные (*radix*) методы сортировки

Часто нет необходимости в обработке ключей в полном объеме

Разумно проводить сравнение по соответствующим частям ключей

Алгоритмы поразрядной сортировки рассматривают ключи как числа, представленные в системе счисления с основанием *R* при различных значениях *R* (*основание системы счисления*), и работают с отдельными цифрами чисел.

Базовые подходы к поразрядной сортировке

- Поразрядная сортировка **MSD** (*most significant digit radix sort - Поразрядная сортировка сначала по старшей цифре*).
- Поразрядная сортировка **LSD** (*least significant digit radix sort — Поразрядная сортировка сначала по младшей цифре*).

MSD/LSD поразрядные сортировки

```
.396465048 .015583409  .0
.353336658 .159072306  .1590
.318693642 .159369371  .1593
.015583409 .269971047  .2
.159369371 .318693642  .31
.691004885 .353336658  .35
.899854354 .396465048  .39
.159072306 .538069659  .5
.604144269 .604144269  .60
.269971047 .691004885  .69
.538069659 .899854354  .8
```

Ключевые условия поразрядной сортировки

- Компьютеры ориентированы на обработку групп битов, называемых *машинными словами*, объединяющихся в небольшие фрагменты, называемые *байтами*;
- Ключи сортировки *также* организуются в последовательности байтов;
- Короткие последовательности байтов могут также служить индексами массивов или машинными адресами.

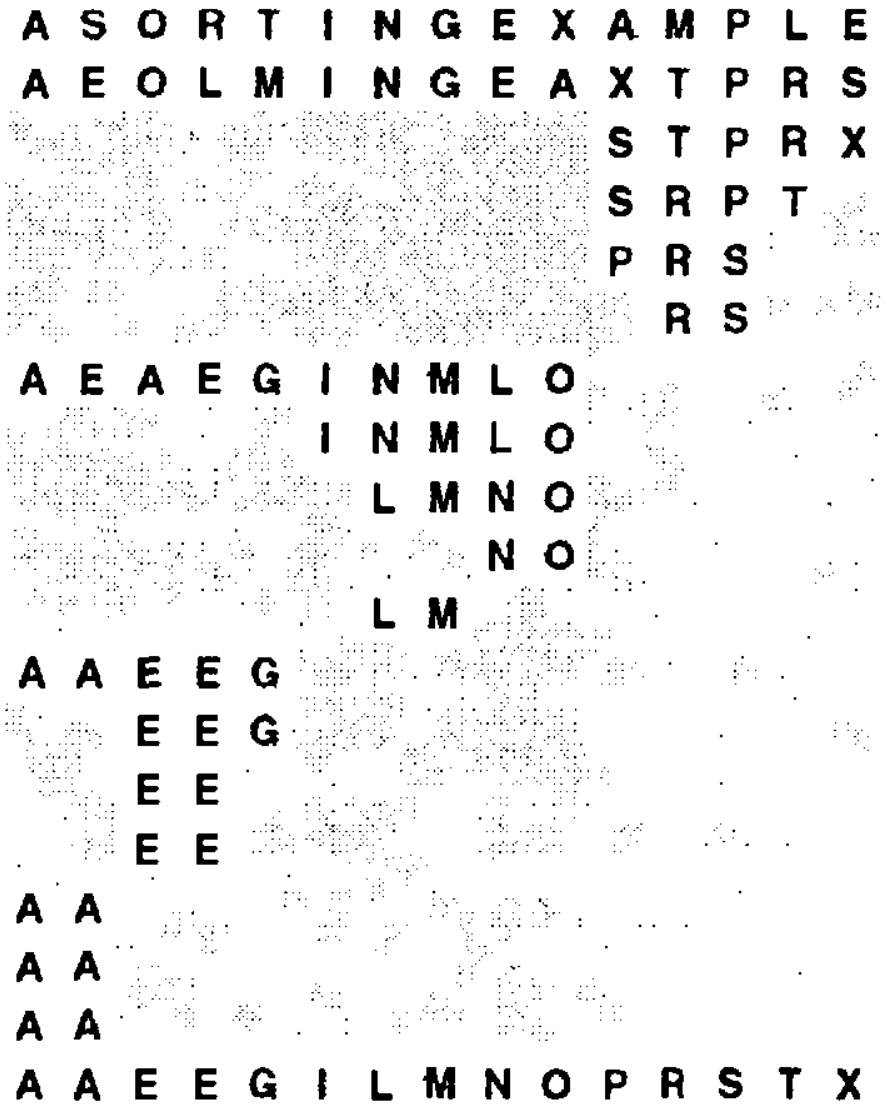
Абстракции для работы

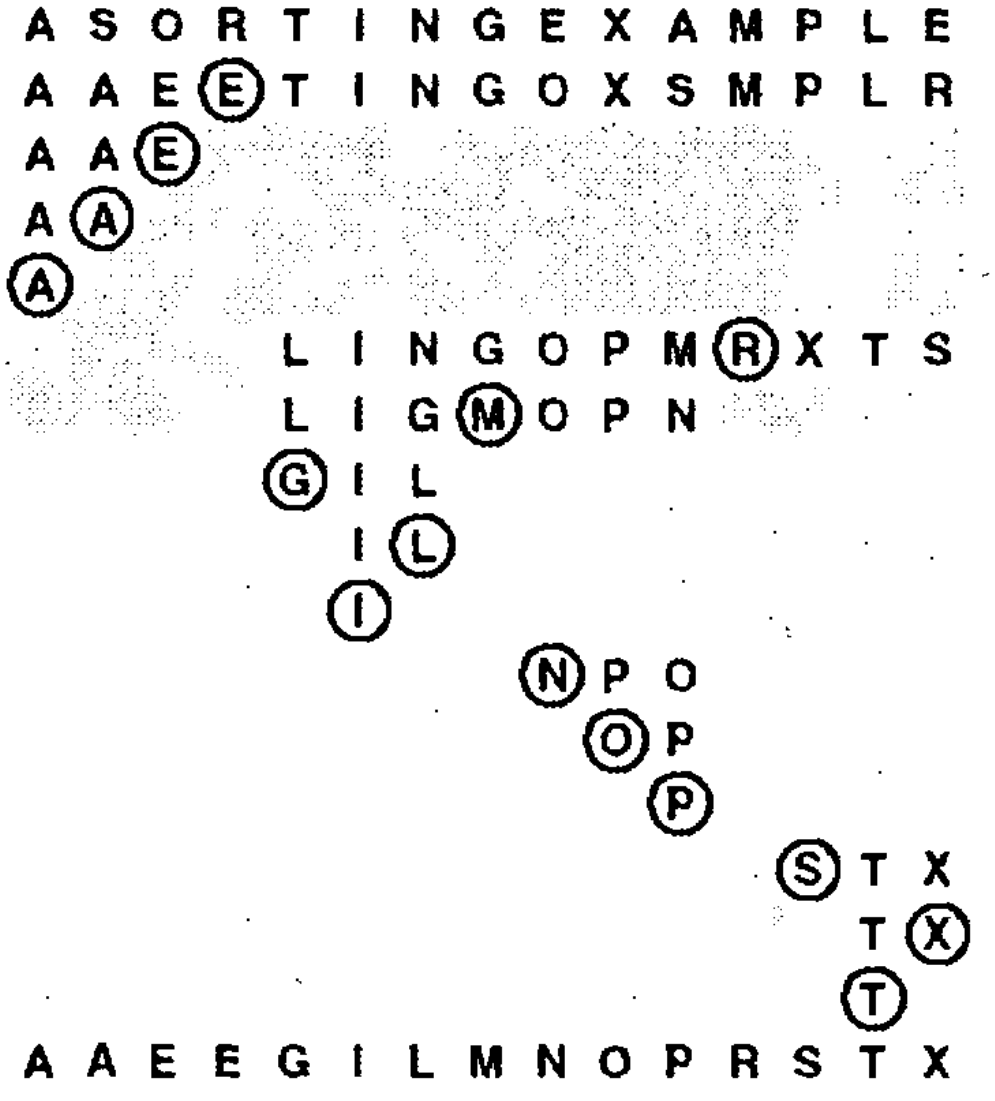
- ***Байт*** представляет собой последовательность битов фиксированной длины,
- ***Строка*** есть последовательность байтов переменной длины,
- ***Слово*** есть последовательность байтов фиксированной длины.
- ***Ключ*** есть число в системе счисления с основанием *R*, цифры которого пронумерованы слева (начиная с 0).

Двоичная быстрая сортировка

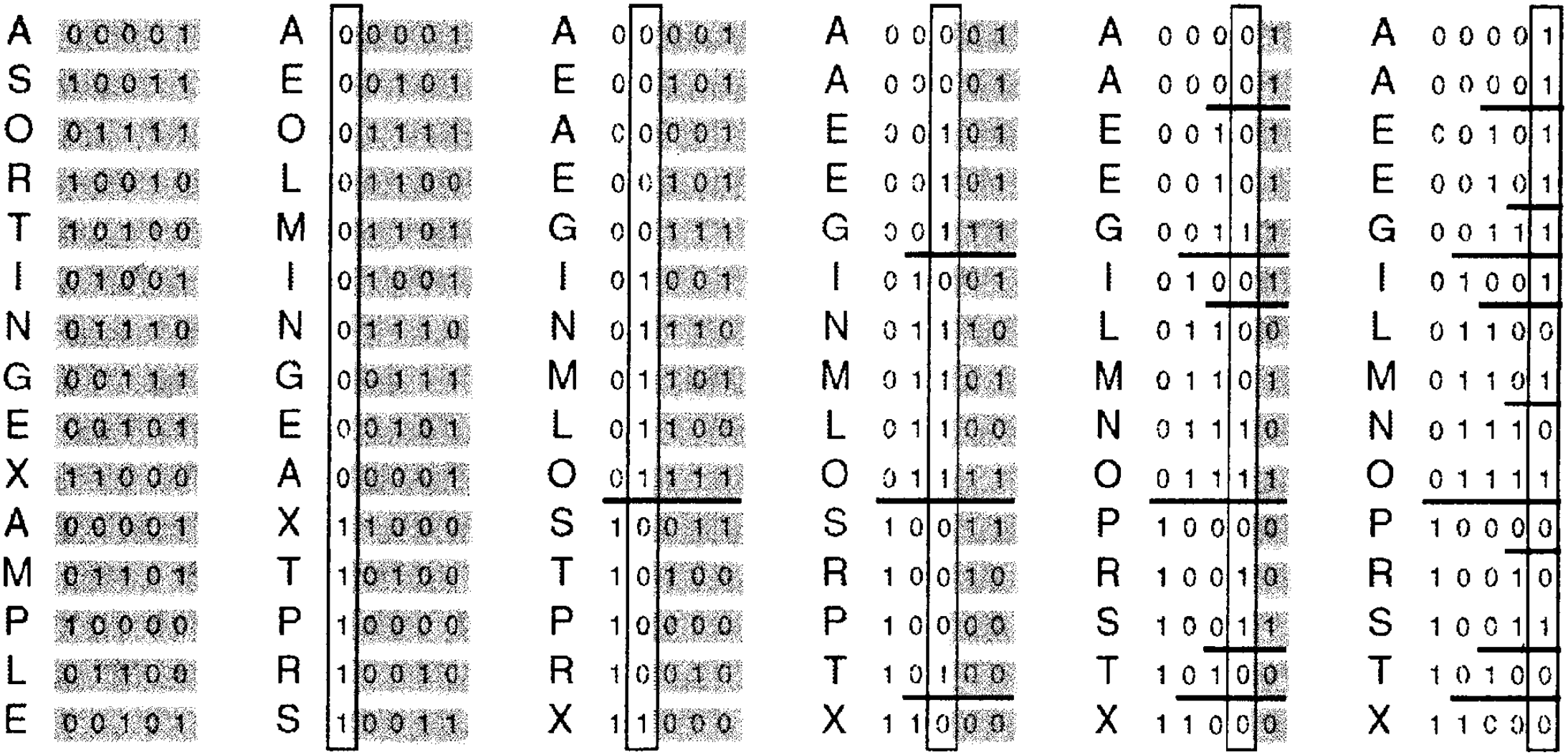
```
void quicksortB(Item a[], int l, int r, int d)
 { int i = l, j = r;
   if (r <= l || d > bitsword) return;
   while (j != i)
     {
      while (digit(a[i], d) == 0 && (i < j)) i++;
      while (digit(a[j], d) == 1 && (j > i)) j--;
      exch(a[i], a[j]);
     }
   if (digit(a[r], d) == 0) j++;
   quicksortB(a, l, j-1, d+1);
   quicksortB(a, j, r, d+1);
 }
```

Пример двоичной быстрой сортировки



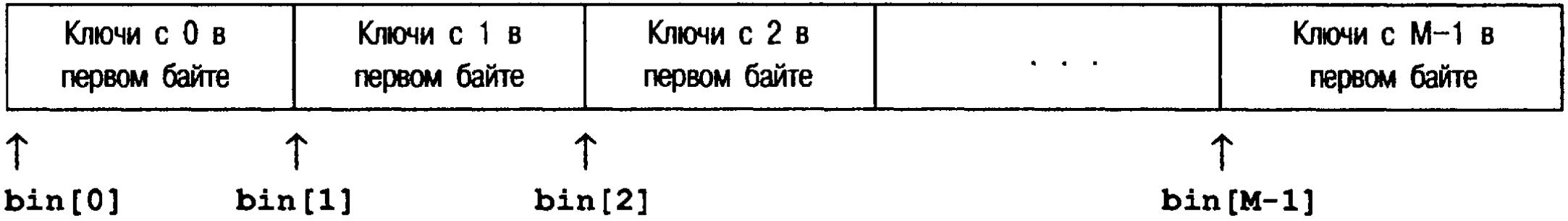


Пример двоичной быстрой сортировки



Поразрядная сортировка **MSD**

- ***Двоичная поразрядная сортировка –*** использование одного разряда
- ***MSD*** – обобщение в системе счисления *R*, при рассмотрении в первую очередь наиболее значащих байт



Используем основание R, соответствующее размеру байта. Для извлечения цифры загружаем байт, чтобы переместиться к следующей цифре увеличиваем на единицу указатель строки. Это при использовании строки фиксированной длины, но возможно использование ключей в виде строк переменной длины.

Поразрядная сортировка **MSD**
проблемы настройки

Выбор R

- если R принимает чрезмерно большое значение, то большая часть стоимости сортировки приходится на инициализацию и проверку корзин,
- если R недостаточно велико, то метод не использует своих потенциальных выгод, что достигается, если разделить исходный файл на максимально возможное число фрагментов.

Поразрядная сортировка **MSD**

```
void radixMSD(Item a[], int l, int r, int d)
  { int i, j, count[R+1];
    static Item aux[maxN];
    if (d > bytesword) return;
    if (r-l <= M) { insertion(a, l, r); return; }
    for (j = 0; j < R; j++) count[j] = 0;
    for (i = l; i <= r; i++)
       count[digit(a[i], d) + 1]++;
    for (j = 1; j < R; j++)
       count[j] += count[j-1];
    for (i = l; i <= r; i++)
       aux[l+count[digit(a[i], d)]++] = a[i];
    for (i = l; i <= r; i++) a[i] = aux[i];
    radixMSD(a, l, bin(0)-1, d+1);
    for (j = 0; j < R-1; j++)
       radixMSD(a, bin(j), bin(j+1)-1, d+1);
}
```

Проблема пустых корзин поразрядной сортировки **MSD**

Пути решения

- Разработка более сложной реализации абстрактной операции доступа к конкретным байтам, которая учитывает любые специальные знания о сортируемых строках.
- *Эвристика в масштабах корзины* (bin-span-heuristics)

Трехпутевая поразрядная быстрая сортировка

Основная идея

Приспособить быструю сортировку для MSD, используя трехпутевое разделение ключей по старшим байтам с переходом к следующему байту только в среднем подфайле

Трехпутевая поразрядная быстрая сортировка

Логика алгоритма

1. Берем опорный элемент.
2. Разделяем массив на три части, сравнивая элементы с опорным по старшему разряду — на меньшие, равные и большие.
3. В каждой из трех частей процедуру повторяем, начиная с шага №1, до тех пор, не дойдем до пустых частей или частей с 1 элементом.

Сложность сортировки — O(n logn).

Дополнительная память — O(1).

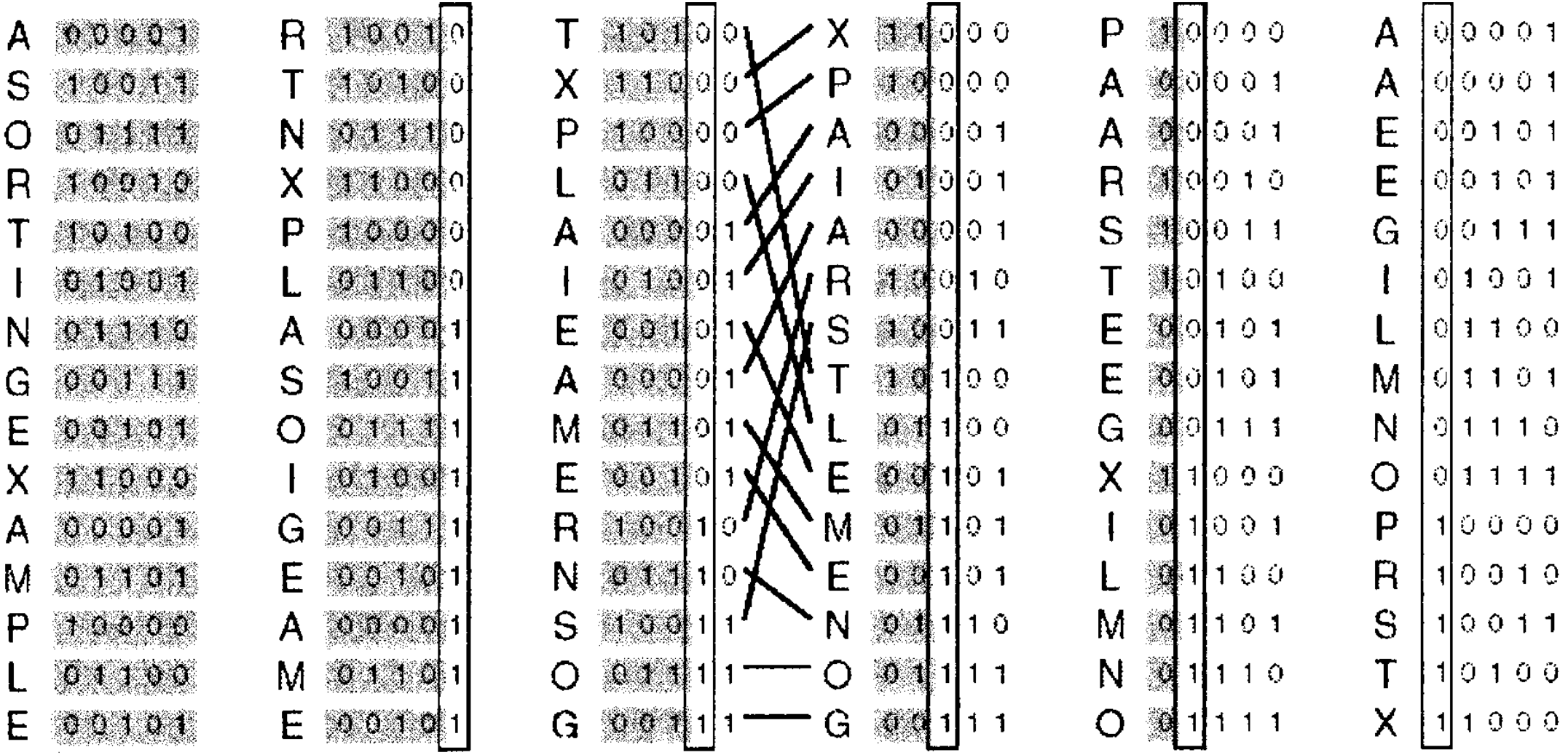
Поразрядная сортировка **LSD**

Альтернативный метод поразрядной сортировки

- Сортируем по последней букве (используем метод подсчета индексных ключей)
- Сортируем по средней букве
- и т.д.

Работает только в том случае, если доказана ***устойчивость*** метода сортировки

Пример поразрядной сортировки **LSD**



Анализ поразрядной сортировки
ключи в виде целых чисел

	4-разрядные байты			8-разрядные байты			16-разрядные байты		
N	Q	M	L	M	L	L*	M	L	M*
12500	2	7	11	28	4	2	52	5	8
25000	5	14	21	29	8	4	54	8	15
50000	10	49	43	35	18	9	58	15	39
100000	21	77	92	47	39	18	67	30	77
200000	49	133	185	72	81	39	296	56	98
400000	102	278	377	581	169	88	119398	110	297
800000	223	919	732	6064	328	203	1532492	219	2309

Q – быстрая сортировка
L – поразрядная сортировка LSD L* – поразрядная сортировка LSD на разрядах MSD.
M – поразрядная сортировка, стандартная
M* - поразрядная сортировка MSD, основание системы счисления адаптируется под размер файла

Анализ поразрядной сортировки
строковые ключи

N	Q	T	M	F	R	X	X*
12500	7	6	9	9	8	6	5
25000	14	12	18	19	15	11	10
50000	34	26	39	49	34	25	24
100000	83	61	87	114	71	57	54

Q – быстрая сортировка
T – быстрая сортировка с трехпутевым разбиением
M – сортировка слиянием
F – пирамидальная сортировка с усовершенствованием Флойда
R – поразрядная сортировка MSD
X – трехпутевая поразрядная быстрая сортировка MSD
X* – трехпутевая поразрядная быстрая сортировка MSD (с отсечениями)