**БИОАЛГОРИТМИКА**

**10.08 Занятие 2.**

1. **Повторение**
2. **Алгоритм слияния «первый-первый»**
   1. Результат – в новом массиве
3. array A[1..M], B[1..N]
4. array C[1..M+N]
5. C[1..M] := A[1..M]
6. p:=1 // указатель на элемент массива C, с которым нужно сравнивать
7. for k from 1 to N
8. **assert** First N+k-1 elements of the array C are incrementally

ordered.

1. **assert** They contain all elements of the array A and first k-1

elements of the array B

1. // find place p of B[k] in C
2. p := p + 1 // номер элемента, с которым нужно сравнивать,

// увеличен на 1 из-за вставки элемента перед ним

10. while B[k] > C[p] AND p < N+k

1. p:=p+1

end\_while

DO: insert B[k] before C[p]

end\_for

Рис.1.

Время работы (количество сравнений) ~ N+M

Объяснение. После каждого выполнению сравнения в строке 10 значение переменной p увеличивается на 1. Если неравенство истинно, то p увеличивается в строке 11. Если ложно – в строке 9. Поэтому проверка неравенства выполняется столько раз, сколько разных значений принимала переменная p (в начале p=1). Максимальное возможное значение p = N+M. Это и дает оценку. Конец объяснения.

* 1. Результат – в том же массиве

algorithm MergeArray(array C[1..L], int Start, NA, NB)

// array С[1..L]

// int Start, NA, NB :

// массив A – C[Start..Start+NA-1]

// массив B – C[Start+NA..Start+NA+NB-1]

p:= Start-1 // (p+1) - указатель на элемент массива C,

// с которым нужно сравнивать очередной элемент массива B

//

for k from Start+NA to Start +NA + NB-1

**assert** Elements of the array C[Start..Start+NA-1+k-1

are incrementally ordered.

**assert** They contain all elements of the array A and first k-1

elements of the array B

**assert** Elememts C[Start..Start+NA-1+k..Start+NA-1+NB] contain last

NB-k+1 elements of the array B

// find place p of B[k] in C

p := p + 1 // номер элемента, с которым нужно сравнивать,

// увеличен на 1 из-за вставки элемента перед ним

while C[Start+NA-1+k] > C[p] AND p < Start+NA-1+k

p:=p+1

end\_while

DO: insert C[Start+NA-1+k] to C[Start..Start+NA-1+k-1] before C[p]

end\_for

end\_algorithm

Рис.2.

1. **Сортировка слиянием для массивов MergeSortArray**

Дано: массив C[1..N]

Получить: массив C[1..N]; все элементы – в возрастающем порядке.

1. аlgorithm MergeSortArray
2. array C[1..N]
3. int k=1 // текущий размер «блока»
4. while k < N
5. s :=1 // начало очередной пары блоков
6. while s < N

// слить два блока длины k

// последний блок может быть короче

1. NA:=k // длина 1-го блока
2. if (s+NA>N) break // остался один блок; ничего не делаем

// есть 2-й блок: вычисляем его длину

1. if (s+2\*k-1 <=N)
2. NB=k
3. else
4. NB = N-s-k+1
5. end\_if

14. MergeArray(C, s, NA, NB)

1. s := s+NA+NB
2. end\_while
3. k := 2\*k
4. end\_while
5. end\_algorithm

Рис.3.

Оценим количество сравнений

При выполнении вызова MergeArray (строка 14) выполняется NA+NB сравнений.

Значит, при одном проходе цикла “while s<N” (строка 6) выполняется всего не более N сравнений.

Так как при каждом выполнении цикла “while k<N” (строка 4) значение k увеличивается в 2 раза, то тело этого цикла (включая тело цикла “while s<N” ) выполняется [log2N] +1 раз.

Итого: общее количество сравнений ~ N\*log2N

Замечание. Важно, что на каждом этапе (прохождении цикла“while k<N” ) у нас все блоки примерно одной длины. Это и позволяет сделать общее количество этапов ~ log2N

.

1. **Структуры данных.**

Структура данных – общее понятие для чего-то, где хранится определенное множество данных. В наших примерах данные – это целые числа, но это не обязательно. Для конкретной структуры данных известно, как устроены «единицы хранения» (ячейки) и как можно осуществлять доступ к ним (извлекать данные и записывать данные). Разные структуры данных позволяют удобно осуществлять разные операции над множествами. Примеры операций: получить элемент (GET), найти максимальный элемент множества (MAX), выяснить, есть ли в множестве заданное число (НАЙТИ); упорядочить элементы множества (SORT); добавить элемент (INSERT), удалить элемент (DELETE).

Пример структуры данных 1. Массив.

Каждая ячейка имеет *индекс* – целое число из определенного интервала. Доступ к ячейке осуществляется указанием имени массива и индекса ячейки.

Если в массиве N элементов, то

- операции MAX и НАЙТИ выполняются за время ~N;

- операция SORT выполняется за время ~N2;

- операция INSERT выполняется за фиксированное время (не зависит от N);

- операция DELETE выполняется за время ~N

- операция GET выполняется за фиксированное время.

1. **Списки.**

Единица хранения в списке - «доминошка». В основной ячейке доминошки хранятся данные, в дополнительной — *указатель,* то есть ссылка на другую доминошку – следующий элемент списка. В последней «доминошке» эта ячейка содержит признак конца списка. Существует специальная доминошка — заголовок списка, которая не хранит никаких данных, ее указатель указывает на 1-й элемент списка, содержащий данные .

Элементы списка не имеют своих имен (например, индексов, как в массиве). Для работы со списком есть отдельный указатель, который указывает на *текущий* элемент списка. В начальный момент указатель списка указывает на заголовок списка. Получить/записать значение можно только для текущего элемента списка. Со списком можно выполнить следующие операции:

* взять значение текущего элемента
* записать в текущий элемент заданное значение
* перейти к следующему элементу
* проверить, не является ли текущий элемент последним
* объявить заголовок текущим элементом (встать в начало списка)
* вставить после текущего новый элемент с заданным значением
* удалить текущий элемент (заголовок удалять нельзя; при удалении заголовка указатель перемещается на предыдущий элемент)

Если в списке N элементов, то

- операции MAX и НАЙТИ выполняются за время ~N;

- операция SORT выполняется за время ~N\*logN;

- операция INSERT выполняется за фиксированное время;

- операция DELETE выполняется за фиксированное время;

- операция GET выполняется за за время ~N.

Пример. Алгоритм MergeSort на списках (аккуратно напишем позже)