# External Sorting

- Used when the data to be sorted is so large that we cannot use the computer's internal storage (main memory) to store it
- We use secondary storage devices to store the data
- The secondary storage devices we discuss here are tape drives. Any other storage device such as disk arrays, etc. can be used

# Two-way Sorting

- Assumptions:
  - Computer's internal storage can hold three records at a time
  - We denote the internal storage capacity by M. Here we have M=3 ← Example (RAM = 128 MB)
  - We only denote the integer key of every record
  - We use four tape drives. <u>One pair of tape drives is denoted by $T_{a1}$ and $T_{a2}$</u> and the other pair is denoted by <u>$T_{b1}$ and $T_{b2}$</u>
  - Initially, all the records that have to be sorted are on $T_{a1}$. $T_{a2}$ , $T_{b1}$ and $T_{b2}$ are empty

# Two-way Sorting Algorithm: Sort Phase

Algorithm:

I. Sort Phase

1. Read M records from one pair of tape drives. Initially, all the records are present only on one tape drive
2. Sort the M records in the computer's internal storage. If M is small (< 10) use insertion sort. For larger values of M use quick sort.
3. Write the M sorted records into the other pair of tape drives (i.e., the pair which does not contain the input records). While writing the records, alternate between the two tape drives of that pair.
4. Repeat steps 1-3 until the end of input

# Two-Way Sort Phase Example

Initially, $T_{a2}$, $T_{b1}$ and $T_{b2}$ are empty and

$T_{a1}$: 81 94 11 96 12 35 17 99 28 58 41 75 15

- Read in 81 94 11 into computer's internal storage and sort them. The output is 11 81 94 which gets written onto $T_{b1}$
- Read in 96 12 35 into computer's internal storage and sort them. The output is 12 35 96 which gets written onto $T_{b2}$
- At the end of the sort phase the contents of the tape drives are:

$T_{a1}$: 81 94 11 96 12 35 17 99 28 58 41 75 15

$T_{a2}$:

$T_{b1}$: 11 81 94 17 28 99 15

$T_{b2}$: 12 35 96 41 58 75

Although $T_{a1}$ contains data, we have sorted and copied the data on the other pair of tape drives. Therefore, $T_{a1}$ is ready to be overwritten

# Two-way Sorting Algorithm: Merge Phase

Algorithm

II. Merge Phase

1. Perform a merge sort reading the data from the input pair of tape drives and writing the data to the output pair of tape drives
2. While writing the data alternate between the two tape drives of the output pair
3. Repeat steps 1 and 2 until nothing is written into one of the output pair of tape drives

# Merge Phase Example

- At the end of the sort phase the contents of the tape drives are:

  $T_{a1}$: 81 94 11 96 12 35 17 99 28 58 41 75

  $T_{a2}$:

  $T_{b1}$: 11 81 94 17 28 99 15

  $T_{b2}$: 12 35 96 41 58 75
- Pass 1 Merge Phase:
  - The input pair of drives for this pass is the b-pair and the output pair is the a-pair
- After pass 1 of the merge phase we get:

  $T_{a1}$: 11 12 35 81 94 96 15

  $T_{a2}$: 17 28 41 58 75 99

  $T_{b1}$: 11 81 94 17 28 99 15

  $T_{b2}$: 12 96 35 41 58 75
- The a-pair now contains the latest merged data and the data in b-pair can be overwritten in the next pass

# Merge Phase Example (contd.)

Pass 2 Merge Phase:

- The input pair of drives for this pass is the a-pair and the output pair is the b-pair
- After pass 2 of the merge phase we get:

$T_{a1}$: 11 12 35 81 94 96 15

$T_{a2}$: 17 28 41 58 75 99

$T_{b1}$: 11 12 17 28 35 41 58 75 81 94 96 99

$T_{b2}$: 15

- The b-pair now contains the latest merged data and the data in a-pair can be overwritten

# Merge Phase Example (contd.)

Pass 3 Merge Phase:

- The input pair of drives for this pass is the b-pair and the output pair is the a-pair
- After pass 3 of the merge phase we get:

  $T_{a1}$: 11 12 15 17 28 35 41 58 75 81 94 96 99

  $T_{a2}$:

  $T_{b1}$: 11 12 17 28 35 41 58 75 81 94 96 99

  $T_{b2}$: 15
- The a-pair now contains the latest merged data and is $T_{a2}$ empty
- The stopping condition for the merge phase is reached
- **No. of passes in Two-way Sorting = ceil(log ceil((N/M)))**
  - N = # input records
  - ~~M = # records that can fit inside internal storage of computer~~

# Multi-way Merge Sorting

- Sort phase remains the same as in two-way sorting
- In two-way sorting we did a 2-way merge
- In multi-way sorting we make a k-way merge
- For this we need two groups of tape drives
- Each group contains k tape drives giving 2*k tape drives in all

# Multi-way Merge Sorting Example

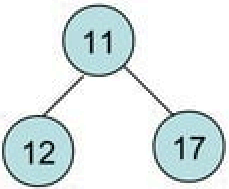
- Problem: Finding the smallest element in the merge phase requires (k-1) comparisons
- Solution: Use a heap to store the elements currently pointed to in each tape drive
- Example: Same data as last example.
- We use 3-way merge that requires 2 groups, each of three tape drives
- At the end of the sort phase we get

$T_{a1}$, $T_{a2}$, $T_{a3}$: can be overwritten

$T_{b1}$: 11 81 94 41 58 75

$T_{b2}$: 12 35 96 15

$T_{b3}$: 17 28 99



# Multi-way Merge Sorting Example

- For the first merge pass
  - The b-tape drives are the input drives
  - The a-tape drives are the output drives
- We store the data pointed to currently on each input tape drive as a heap in the computer's internal storage
- Initially, the heap would contain 11 12 17
- We do a `deletmin()` on the heap and write the record returned by the `deletmin()` into $T_{a1}$
- The cur-pointer to $T_{b1}$ advances by one to point to 81
- We now have to insert 81 inside the heap

# Multi-way Merge Sorting Example

- The heap then becomes 12 17 81
- The next `deletmin()` yields 12 which is written on $T_{a1}$
- We continue to write on $T_{a1}$ until we have written 9 records in it and then we switch to $T_{a2}$ for output
- In this phase we have combined three 3-element data sets from the input b-tape drives into 9-element data sets on the output a-tape drive

$T_{a1}$: 11 12 17 28 35 81 94 96 99

$T_{a2}$: 15 41 58 75

$T_{a3}$:

$T_{b1}$, $T_{b2}$, $T_{b3}$: can be overwritten

# Multi-way Merge Sorting Example

- In the second merge pass we will combine the contents of $T_{a1}$, $T_{a2}$, and $T_{a3}$ and write the merged data on the b-tape drives
- Here, we will be combining three 9-element data sets from the input a-tape drives into one 27-element data set on the output b-tape drives
- We stop when, after a merge pass, (k-1) of the output tape drives are empty
- After the second merge pass $T_{b1}$ contains all the 13 elements of the input data set and $T_{b2}$ and $T_{b3}$ are empty
- The stopping condition is reached
- No. of passes in k-way merge = ceil($\log_k$ ceil(N/M))

# Example 5-Way

**50 110 95|10 100 36|153 40 120|60 70 130|22 140 80**

pass 1

Ta1 50 110 95|10 100 36|153 40 120|60 70 130|22 140 80

Ta2

Ta3

Ta4

Ta5

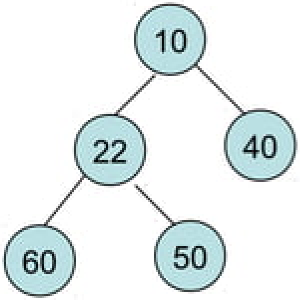
Tb1 50 95 110

Tb2 10 36 100

Tb3 40 120 153

Tb4 60 70 130

Tb5 22 80 140



pass 2

36 40 50 60 70 80 95 100 110 120 130 140 153

Ta1

Ta2

Ta3

Ta4

Ta5

Tb1

Tb2

Tb3

Tb4

Tb5

# Poly-Phase Merge

- K-way merge requires 2*k tape drives
- We can reduce the number of tape drives if we unevenly split the input data set (runs or # of M) for each merge pass
- If there is an available tape drive, stop the merge pass, and begin new merge pass
- For a 2-way merge the ratio of splitting input data (runs) is guided by the Fibonacci number series:
  $a_{i+1} = a_i + a_{i-1}, \quad a_1 = a_2 = 1$

  1 1 2 3 5 8 13 21 34…

# Poly-Phase Merge

- Example:
  - 13 runs split as 8 and 5
  - 34 runs split as 21 and 13
- For non-fibonacci numbers add dummy runs to reach the nearest Fibonacci number
- K-way poly-phase merge uses (k+1) tape drives instead of 2*k tape drives

# Example 1

**50 110 95 | 10 100 36 | 153 40 120 | 60 70 130 | 22 140 80**

pass 1

Ta1

Ta2 50 95 110 | 40 120 153 | 22 80 140

Ta3 10 36 100 | 60 70 130

pass 2

Ta1 10 36 50 95 100 110 | 40 60 70 120 130 153

Ta2 22 80 140

Ta3

pass 3

Ta1 40 60 70 120 130 153

Ta2

10 22 36 50 80 95 100 110 140

Ta3

pass 4

Ta1

36 40 50 60 70 80 95 100 110 120 130 140 153

Ta2

Ta3

# Example 2

**50 110 95 | 10 100 36 | 153 40 120 | 60 70 130**

Pass 1 **Internal Sort**

Ta1

Ta2 50 95 110 | 40 120 153

Ta3 10 36 100 | 60 70 130

Pass 2 **Merge**

Ta1 10 36 50 95 100 110 | 40 60 70 120 130 153

Ta2

Ta3

Pass 3 **Distribute**

Ta1

Ta2 10 36 50 95 100 110

Ta3 40 60 70 120 130 153

Pass 4 **Merge**

Ta1

Ta2

Ta3 36 40 50 60 70 80 95 100 110 120 130 140 153

**Thus, even runs will not increase the speed of sorting**

# Replacement Selection

- Replacement Selection allows for initial runs to contain more records than can fit in memory
- When records are written to tape drive, the internal memory is available
- If next record in the input tape is larger than the record we have just output, then it can be include in the run

# Replacement Selection

Algorithm:

1. Read M records as a heap in the computer's internal storage
2. while (heap is not empty){
   a. Perform deletemin() and send to output
   b. while (next input record > last deletemin )
      insert input record into heap
   a. if (next input record < last deletemin)
      store the record outside the heap
      /* the region outside the heap in the
         computer's internal storage is called
         **dead space** */
1. If there are more input records create a new heap of M records and repeat step 2

Perform an **external sorting with replacement selection technique** on the following data. Assume that the memory can hold 4 records (M = 4) at a time and there are 4 tape drives (Ta1, Ta2, Tb1, and Tb2). Initially all data are stored in tape drive Ta1.

| Tape drive | Data |
|---|---|
| Ta1 | 55 94 11 6 12 35 17 99 28 58 41 75 15 38 19 100 8 80 |
| Ta2 | |
| Tb1 | |
| Tb2 | |

1. Sort the following data using a merge sort. You should use divide and conquer method described in class.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 7 | 1 | 12 | 10 | 8 | 20 | 6 | 9 |

2. Perform the 2-way <u>poly phase merge sort</u> on the following sequence of data which is stored in tape drive Ta1.
55 94 11 6 12 35 17 99 28 58 41 75 15 38 19 100 8 80
Initially, Ta2, Ta3, are empty and M = 3.

| Tape drive | Contents |
|---|---|
| Ta1 | 55 94 11 6 12 35 17 99 28 58 41 75 15 38 19 100 8 80 |
| Ta2 | |
| Ta3 | |

Write the table at each pass. (including sort phase, and merge phase)