"Verdana"

CmSc 250 Intro to Algorithms

External Sorting

1. Characteristics

1. Processing large files, unable to fit into the main memory
2. Restrictions on the access, depending on the external storage medium
3. Primary costs – for input-output
4. Main concern: minimize the number of times each piece of data is moved between the external storage and the main memory.

2. General strategy - Sort-Merge

- Break the file into blocks about the size of the internal memory
- Sort these blocks
- Merge sorted blocks

Usually several passes are needed, creating larger sorted blocks until the whole file is sorted

3. Basic Algorithm

Assumptions:
four tapes:

two for input - Ta1, Ta2,
two for output - Tb1, Tb2.

Initially the file is on Ta1.

N records on Ta1
M records can fit in the memory

Step 1: Break the file into blocks of size **M, [N/M]+1** blocks

Step 2: Sorting the blocks:

- read a block, sort, store on Tb1
- read a block, sort, store on Tb2,
- read a block, sort, store on Tb1,
- etc, alternatively writing on Tb1 and Tb2

Each sorted block is called a **run.**
Each output tape will contain half of the runs

Step 3: Merge:

a. **From Tb1, Tb2 to Ta1, Ta2.**
Merge the first run on Tb1 and the first run on Tb2, and store the result on Ta1:

> Read two records in main memory, compare, store the smaller on Ta1
>
> Read the next record (from Tb1 or Tb2 - the tape that contained the record stored on Ta1) compare, store on Ta1, etc.

Merge the second run on Tb1 and the second run on Tb2, store the result on Ta2.
Merge the third run on Tb1 and the third run on Tb2, store the result on Ta1.
Etc, storing the result alternatively on Ta1 and Ta2.

Now Ta1 and Ta2 will contain sorted runs twice the size of the previous runs on Tb1 and Tb2

b. **From Ta1, Ta2 to Tb1, Tb2.**
Merge the first run on Ta1 and the first run on Ta2, and store the result on Tb1.
Merge the second run on Ta1 and the second run on Ta2, store the result on Tb2
Etc, merge and store alternatively on Ta1 and Ta2.

c. Repeat the process until only one run is obtained. This would be the sorted file

Example of a basic external sorting

4. Analysis of two-way merge

The algorithm requires [log(N/M)] passes plus the initial run-constructing pass.

Each pass merges runs of length r to obtain runs of length 2*r.
The first runs are of length M. The last run would be of length N.

Let's assume that N is a multiple of M.

Initial situation:

> 1st tape contains N records = M records * N/M runs

After storing the runs on two tapes, each contains half of the runs:

> 2 tapes * M records_per_run * (1/2)(N/M) runs = N records

After merge 1st pass - double the length of the runs, halve the number of the runs:

> 2 tapes * 2M records_per_run * (1/2)(1/2)(N/M) runs = N records

After merge 2nd pass :

> 2 tapes * 4M records_per_run * (1/4)(1/2) (N/M) runs = N records

......

After merge s-th pass:

> 2 tapes * 2^s M records_per_run * ($1/2^s$)(1/2)(N/M) runs = N records

......

Thus the length of the runs after the s-th merge is 2^sM.

After the last merge there is only one run equal to the whole file:

> $2^sM = N$
>
> $2^s = N/M$

s = log(N/M)

if s is the last merge, s = log(N/M)

At each pass we process N records, so the complexity is **O(Nlog(N/M))**

5. Multi-way merge

The basic algorithm is 2-way merge - we use 2 output tapes.

Assume that we have **k** tapes - then the number of passes will be reduced - $\log_k(N/M)$
At a given merge step we merge the first **k** runs, then the second **k** runs, etc.

The task here: to find the smallest element out of **k** elements

Solution: priority queues

Idea: Take the smallest elements from the first **k** runs,
store them into main memory in a heap tree.

Then repeatedly output the smallest element from the heap.
The smallest element is replaced with the next element from the run from which it came.

When finished with the first set of runs, do the same with the next set of runs.

Example of a multi-way merge

Learning Goals

- Be able to explain how the basic two-way sort-merge algorithm works.
- Be able explain how the multiway sort-merge algorithm works
  and why its efficiency is improved over the basic algorithm.

Back to Contents page

Created by Lydia Sinapova