Sorting

- efficient evaluation for many operations
- required by query:
  - SELECT cid,name FROM student ORDER BY name
- implementations
  - internal sorting (if records fit in memory)
  - external sorting

External Sort-Merge Algorithm (1/3)

- Sort stage: create sorted *runs*

```
i=0;
repeat
    read $M$ pages of relation R into memory
    sort the $M$ pages
    write them into file $R_i$
    increment i
until no more pages
N = i            // number of runs
```

External Sort-Merge Algorithm (2/3)

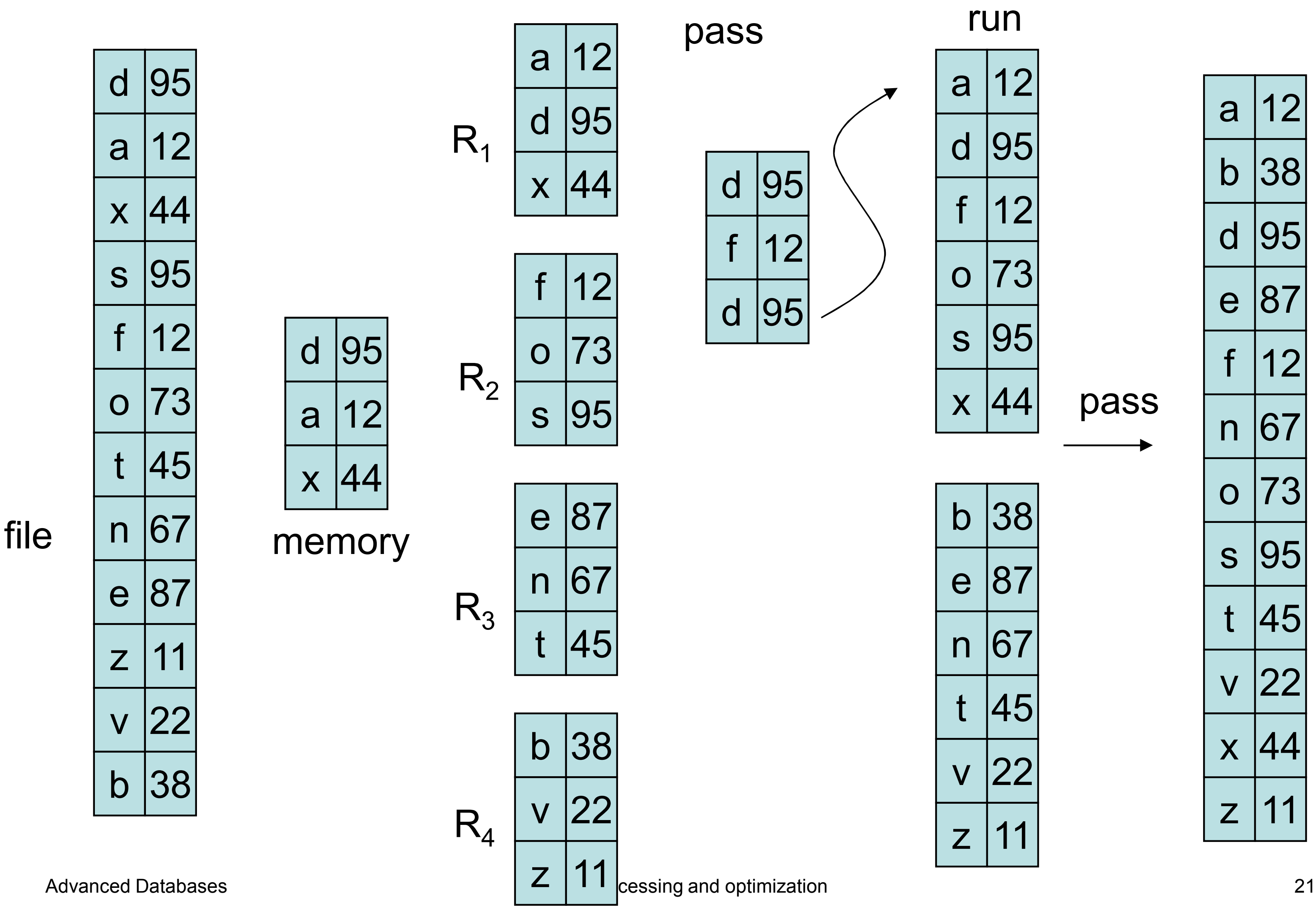
- Merge stage: merge sorted *runs*

```
//assuming N < M
allocate a page for each run file $R_i$          // N pages allocated
read a page $P_i$ of each $R_i$
repeat
    choose first record (in sort order) among N pages, say from page $P_j$
    write record to output and delete from page $P_j$
    if page is empty read next page $P_j$' from $R_j$
until all pages are empty
```

External Sort-Merge Algorithm (3/3)

- Merge stage: merge sorted *runs*
- What if N > M ?
  - perform multiple *passes*
  - each *pass* merges M-1 runs until relation is processed
  - in next pass number of runs is reduced
  - final *pass* generated sorted output

Sort-Merge Example



file

d	95
a	12
x	44
s	95
f	12
o	73
t	45
n	67
e	87
z	11
v	22
b	38

memory

d	95
a	12
x	44

R_1

a	12
d	95
x	44

R_2

f	12
o	73
s	95

R_3

e	87
n	67
t	45

R_4

b	38
v	22
z	11

pass

d	95
f	12
d	95

run

a	12
d	95
f	12
o	73
s	95
x	44

b	38
e	87
n	67
t	45
v	22
z	11

pass

a	12
b	38
d	95
e	87
f	12
n	67
o	73
s	95
t	45
v	22
x	44
z	11

Sort-Merge cost

- B_R the number of pages of R
- Sort stage: 2 * B_R
  - read/write relation
- Merge stage:
  - initially $\left\lceil \frac{B_R}{M} \right\rceil$ runs to be merged
  - each *pass* M-1 runs sorted
  - thus, total number of passes: $\left\lceil \log_{M-1}\left(\frac{B_R}{M}\right) \right\rceil$
  - at each pass 2 * B_R pages are read
    - read/write relation
    - apart from final write
- Total cost:
  - $2 * B_R + 2 * B_R * \left\lceil \log_{M-1}\left(\frac{B_R}{M}\right) \right\rceil - B_R$