Externspeicher-Sortieren demogr.

Sei $B = M^{1/3}$ und $n = M^{1/2}$. Wieviele Mergephasen braucht dann das verbesserte Externspeicher-Sortieren?

- (1) $\Theta(1)$ Phasen
- (2) $\Theta(\log \frac{M}{B})$ Phasen
- (3) $\Theta(\log n)$ Phasen
- (4) $\Theta(n)$ Phasen
- (5) ist mir nicht mehr klar.

Externspeicher-Sortieren demogr.

Sei $B = M^{1/3}$ und $n = M^{1/2}$. Wieviele Mergephasen braucht dann das verbesserte Externspeicher-Sortieren?

- (1) $\Theta(1)$ Phasen
- (2) $\Theta(\log \frac{M}{B})$ Phasen
- (3) $\Theta(\log n)$ Phasen
- (4) $\Theta(n)$ Phasen
- (5) ist mir nicht mehr klar.

Auflösung:

Sei $B = M^{1/3}$ und $n = M^{1/2}$. Wieviele Mergephasen braucht dann das verbesserte Externspeicher-Sortieren?

- (1) $\Theta(1)$ Phasen
- (2) $\Theta(\log \frac{M}{B})$ Phasen
- (3) $\Theta(\log n)$ Phasen
- (4) $\Theta(n)$ Phasen
- (5) ist mir nicht mehr klar.

Auflösung: (1) $\Theta(1)$ Phasen

Sei in Z[1..n] eine Permutation der Zahlen $1, 2, \ldots, n$ abgelegt.
Wieviel I/O produziert folgendes Codestück im Worst-Case

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Externspeicher - Umsortieren demogr.

Sei in Z[1..n] eine Permutation der Zahlen $1, 2, \ldots, n$ abgelegt.
Wieviel I/O produziert folgendes Codestück im Worst-Case

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

- (1) $\Theta(1)$ I/Os.
- (2) $\Theta(n/B)$ I/Os.
- (3) $\Theta(\mathrm{sort}(n))$ I/Os.
- (4) $\Theta(n)$ I/Os.
- (5) $\Theta(n \log n)$ I/Os.
- (6) Θhne meinen Anwalt sag ich gar nichts!

Sei in Z[1..n] eine Permutation der Zahlen $1, 2, \ldots, n$ abgelegt.
Wieviel I/O produziert folgendes Codestück im Worst-Case

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

- (1) $\Theta(1)$ I/Os.
- (2) $\Theta(n/B)$ I/Os.
- (3) $\Theta(\mathrm{sort}(n))$ I/Os.
- (4) $\Theta(n)$ I/Os.
- (5) $\Theta(n \log n)$ I/Os.
- (6) Θhne meinen Anwalt sag ich gar nichts!

Auflösung:

Sei in Z[1..n] eine Permutation der Zahlen $1, 2, \ldots, n$ abgelegt.
Wieviel I/O produziert folgendes Codestück im Worst-Case

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

- (1) $\Theta(1)$ I/Os.
- (2) $\Theta(n/B)$ I/Os.
- (3) $\Theta(\mathrm{sort}(n))$ I/Os.
- (4) $\Theta(n)$ I/Os.
- (5) $\Theta(n \log n)$ I/Os.
- (6) Θhne meinen Anwalt sag ich gar nichts!

Auflösung: (4), mit ein wenig Sortieren geht es aber besser:

Externspeicher - Umsortieren

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

Externspeicher - Umsortieren

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

```
SCAN Z:      (Z[1]=17,1),   (Z[2]=5,2),    ...
```

Externspeicher - Umsortieren

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

```
SCAN Z:       (Z[1]=17,1),  (Z[2]=5,2),   ...
SORT(1st):    (Z[73]=1,73), (Z[12]=2,12), ...
```

Externspeicher - Umsortieren

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

```
SCAN Z:       (Z[1]=17,1),  (Z[2]=5,2),    ...
SORT(1st):    (Z[73]=1,73), (Z[12]=2,12),  ...
par. SCAN :   (Y[1],73),    (Y[2],12),     ...
```

Externspeicher - Umsortieren

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

```
SCAN Z:       (Z[1]=17,1),   (Z[2]=5,2),     ...
SORT(1st):    (Z[73]=1,73),  (Z[12]=2,12),   ...
par. SCAN :   (Y[1],73),     (Y[2],12),      ...
SORT(2nd):    (Y[Z[1]],1),   (Y[Z[2]],2),    ...
```

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

```
SCAN Z:       (Z[1]=17,1),   (Z[2]=5,2),    ...
SORT(1st):    (Z[73]=1,73), (Z[12]=2,12),  ...
par. SCAN :   (Y[1],73),     (Y[2],12),     ...
SORT(2nd):    (Y[Z[1]],1),   (Y[Z[2]],2),   ...
par. SCAN :  X[1]=Y[Z[1]], X[2]=Y[Z[2]],  ...
```

Externspeicher - Umsortieren

```
int[1..n] X,Y,Z;
for i=1 to n do X[i]:=Y[Z[i]];
```

Wende folgendes Vorgehen an:

```
SCAN Z:       (Z[1]=17,1),   (Z[2]=5,2),    ...
SORT(1st):    (Z[73]=1,73),  (Z[12]=2,12),  ...
par. SCAN :   (Y[1],73),     (Y[2],12),     ...
SORT(2nd):    (Y[Z[1]],1),   (Y[Z[2]],2),   ...
par. SCAN :  X[1]=Y[Z[1]],  X[2]=Y[Z[2]],   ...
```

Das braucht nur $O(\mathrm{sort}(n))$ I/Os, unabhängig von Permutation in Z[].

Es seien n Zahlen aus dem Bereich $\{0, \ldots, n^x\}$ zu sortieren. Bei welchem Parameter x ist Mergesort in etwa genauso schnell oder schneller als Radixsort?

- (1) $x = \Omega(1)$.
- (2) $x = \Omega(\log n)$.
- (3) $x = \Omega(\sqrt{n})$.
- (4) $x = \Omega(n)$.
- (5) niemals.

Es seien n Zahlen aus dem Bereich $\{0, \dots, n^x\}$ zu sortieren. Bei welchem Parameter x ist Mergesort in etwa genauso schnell oder schneller als Radixsort?

- (1) $x = \Omega(1)$.
- (2) $x = \Omega(\log n)$.
- (3) $x = \Omega(\sqrt{n})$.
- (4) $x = \Omega(n)$.
- (5) niemals.

Auflösung:

Sortieren demogr.

Es seien n Zahlen aus dem Bereich $\{0, \ldots, n^x\}$ zu sortieren. Bei welchem Parameter x ist Mergesort in etwa genauso schnell oder schneller als Radixsort?

- (1) $x = \Omega(1)$.
- (2) $x = \Omega(\log n)$.
- (3) $x = \Omega(\sqrt{n})$.
- (4) $x = \Omega(n)$.
- (5) niemals.

Auflösung: (2) $x = \Omega(\log n)$.