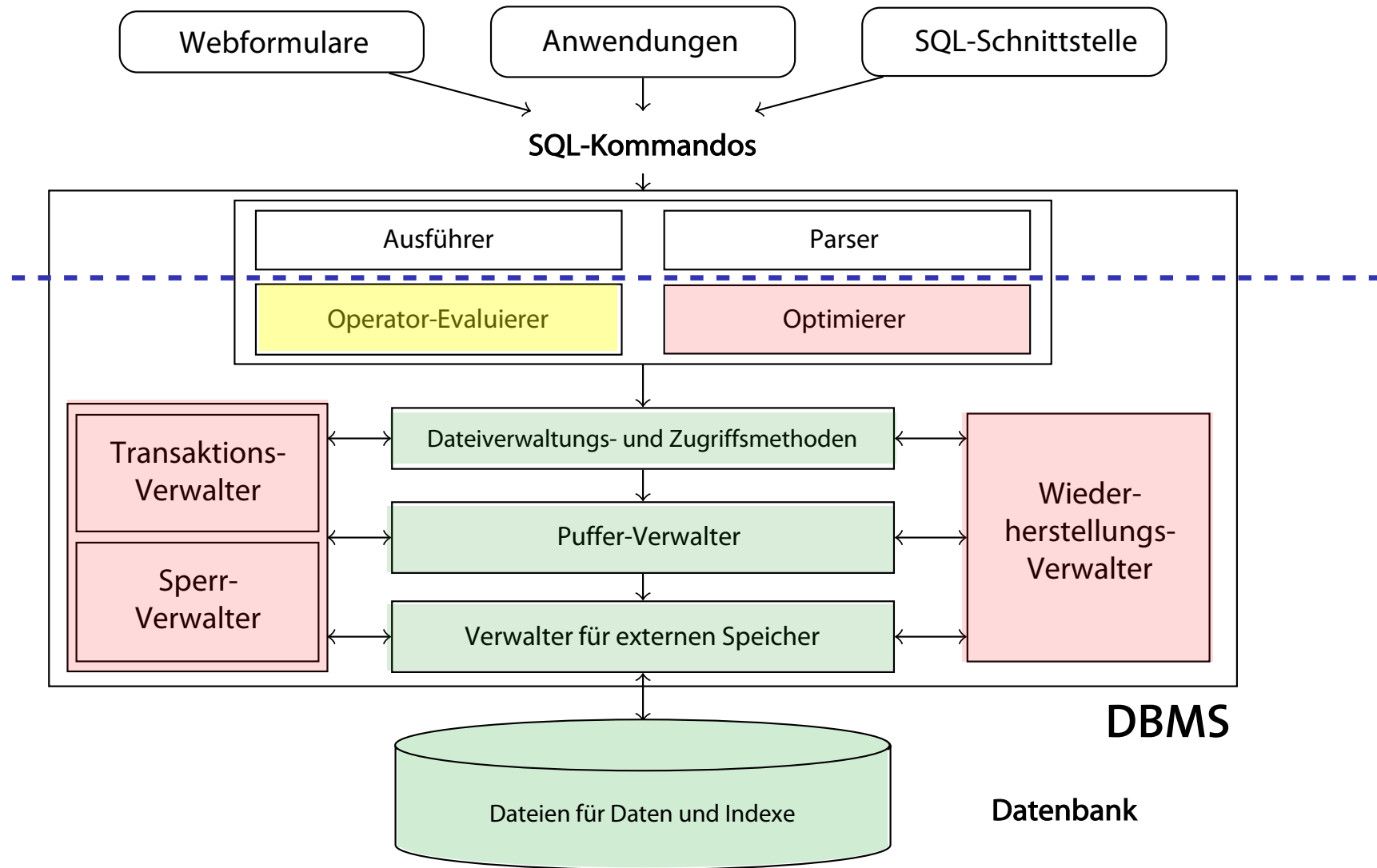

Datenbanken

Anfrageverarbeitung

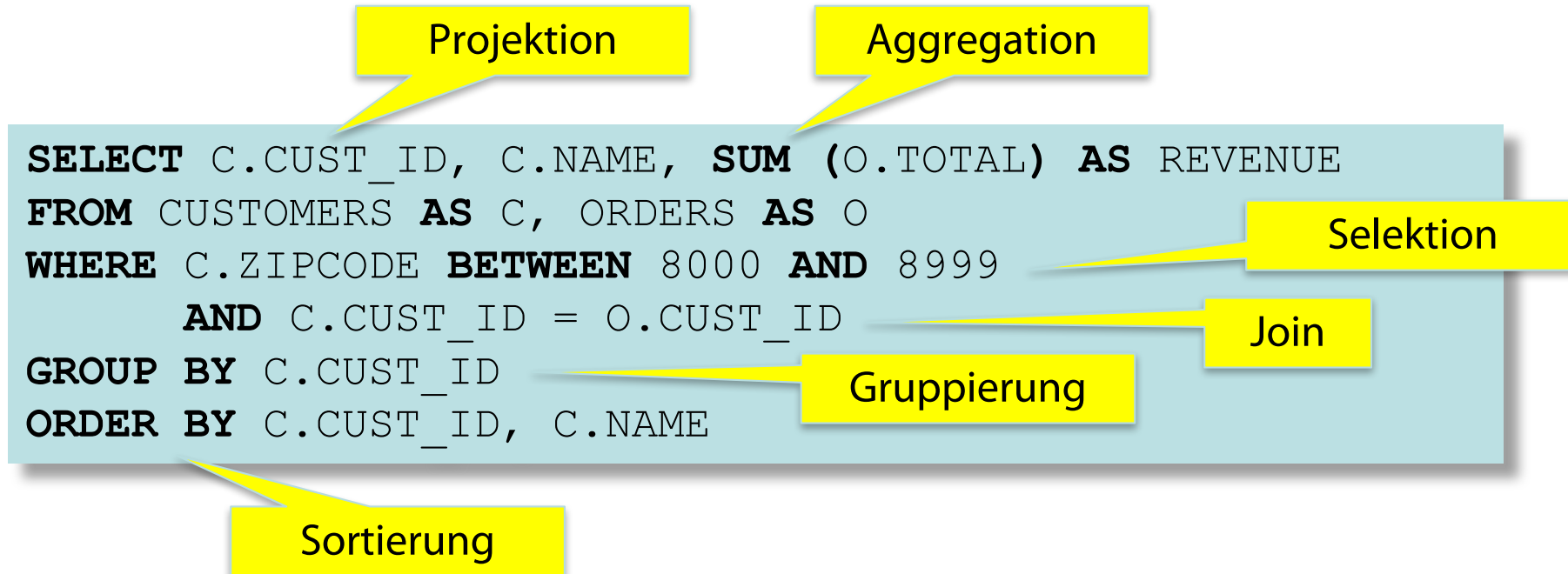
Dr. Özgür Özçep
Universität zu Lübeck
Institut für Informationssysteme



Architektur eines DBMS



Anfragebeantwortung



Ein DBMS muss eine Menge von Aufgaben erledigen:

- mit minimalen Ressourcen
- über großen Datenmengen
- und auch noch so schnell wie möglich

Danksagung

- Diese Vorlesung ist inspiriert von den Präsentationen zu dem Kurs:

„Architecture and Implementation of Database Systems“
von Jens Teubner an der ETH Zürich

- Graphiken und Code-Bestandteile wurden mit Zustimmung des Autors und ggf. kleinen Änderungen aus diesem Kurs übernommen

Sortierung

Wichtige Datenbankoperation mit vielen Anwendungen

- Eine SQL-Anfrage kann **Sortierung anfordern**

```
SELECT A,B,C FROM R ORDER BY A
```

- **Bulk-loading** eines B⁺-Baumes fußt auf sortierten Daten
- **Duplikate-Elimination** wird besonders einfach

```
SELECT DISTINCT A,B,C FROM R
```

- Einige Datenbankoperatoren setzen **sortierte Eingabedateien** voraus (kommt später)

Wie können wir eine Datei sortieren, die nicht in den Hauptspeicher passt (und auf keinen Fall in den vom Pufferverwalter bereitgestellten Platz)?

Zwei-Wege-Mischsortieren (Merge Sort)

Sortierung von Dateien beliebiger Größe in nur 3 Seiten des Pufferverwalters

- Sortierung von N Seiten in mehreren Durchgängen

Beispiel

n-page run = Datei aus n Seiten

input file

65	43	47	89	52	13	8
----	----	----	----	----	----	---

Zwei-Wege-Mischsortieren

Pass 0 (Input: $N = 2^k$ unsorted pages; Output: 2^k sorted runs)

1. **Read** N pages, **one page at a time**
2. **Sort** records in main memory.
3. **Write** sorted pages to disk (each page results in a **run**).

This pass requires **one page** of buffer space.

Pass 1 (Input: $N = 2^k$ sorted runs; Output: 2^{k-1} sorted runs)

1. Open two runs r_1 and r_2 from Pass 0 for reading.
2. **Merge** records from r_1 and r_2 , reading input page-by-page.
3. **Write** new two-page run to disk (page-by-page).

This pass requires **three pages** of buffer space.

⋮

Pass n (Input: 2^{k-n+1} sorted runs; Output: 2^{k-n} sorted runs)

1. Open two runs r_1 and r_2 from Pass $n - 1$ for reading.
2. **Merge** records from r_1 and r_2 , reading input page-by-page.
3. **Write** new 2^n -page run to disk (page-by-page).

This pass requires **three pages** of buffer space.

I/O-Verhalten

- Um eine Datei mit N Seiten zu sortieren, werden in jedem Durchgang N Seiten gelesen und geschrieben
→ $2 \cdot N$ I/O-Operationen pro Durchgang

- Anzahl der Durchgänge: $1 + \lceil \log_2 N \rceil$
 ↑ ↑
 Durchgang 0 Durchgänge 1..k

- Anzahl der I/O-Operationen:
 $2 \cdot N \cdot (1 + \lceil \log_2 N \rceil)$

Aufgabe:

Anzahl der Durchgänge: $1 + \lceil \log_2 N \rceil$

Anzahl der I/O-Operationen:
 $2 \cdot N \cdot (1 + \lceil \log_2 N \rceil)$

Wie lange dauert die Sortierung einer 8GB Datei bei einer Travelstar 7k200?

- 8KB pro Seite
- Suchzeit $t_s = 10 \text{ ms}$
- Rotationsverzögerung $t_r = 4,17 \text{ ms}$
- Transferzeit $t_{tr} = 0,16 \text{ ms}$ für 8KB-Seite

Lösung

$\approx 7d$

- Wahlfreier Zugriff mit Zugriffszeit =
 $10ms + 4,17ms + 0,16ms = 14,33 \text{ ms}$
- Anzahl Seiten = $8GB / 8KB = 10^6$
- I/O-Operationen = $2 \cdot 10^6 (1 + \lceil 6 \log_2(10) \rceil) = 42 * 10^6$
- Zeit = $42 * 10^6 * 14,33ms \approx 7d$

Externes Mischsortieren

- Bisher freiwillig nur 3 Seiten verwendet
- Wie kann ein großer Pufferbereich genutzt werden:
 B Seiten auf einmal bearbeiten
- Zwei wesentliche Stellgrößen
 - **Reduktion der initialen Runs** durch Verwendung des Pufferspeichers beim Sortieren im Hauptspeicher
 - **Reduktion der Anzahl der Durchgänge** durch Mischen von mehr als 2 Seiten

Reduktion der Anzahl der Durchgänge

Mit B Seiten im Puffer können B Seiten eingelesen und im Hauptspeicher sortiert werden

Pass 0 (Input: N unsorted pages; Output: $\lceil N/B \rceil$ sorted runs)

1. **Read** N pages, B pages at a time
 2. **Sort** records in main memory.
 3. **Write** sorted pages to disk (resulting in $\lceil N/B \rceil$ runs).
- This pass uses B pages of buffer space.

- Pass $n > 0$ wie gehabt
- Anzahl der I/O-Operationen:

$$2 \cdot N \cdot (1 + \lceil \log_2 \lceil N/B \rceil \rceil)$$

- Was ist das Zugriffsmuster auf diese Ein-Ausgaben?
- Antwort: Chunks von B Seiten sequenziell gelesen/geschrieben!

Aufgabe:

Wie lange dauert die Sortierung einer 8GB Datei mit $B=1000$ Pufferseiten je 8KB mit einer Travelstar 7k200?

- 8KB pro Seite, $t_s = 10$ ms, $t_r = 4.17$ ms, $t_{tr} = 0,16$ ms für 8KB-Seite

Anzahl der Durchgänge: $1 + \lceil \log_2 \lceil N/B \rceil \rceil$

Anzahl der I/O-Operationen:

$$2 \cdot N \cdot (1 + \lceil \log_2 \lceil N/B \rceil \rceil)$$

Lösung

$\approx 3d$

- Durchgänge = $1 + \lceil \log_2 \lceil N/B \rceil \rceil =$
 $1 + \lceil \log_2 \lceil 10^6 / 10^3 \rceil \rceil = 1 + \lceil \log_2 10^3 \rceil = 11$
- Also: Mergedurchgänge = 10
- Im ersten Durchgang kann Such- und Rotationszeit vernachlässigt werden
- Damit bleibt nur für die Mergedurchgänge eine Such- und Rotationszeit von insgesamt $2 * 10^6 * 10 * 14,17ms \approx 3d$
- Gesamte Transferzeit ergibt sich zu
 $2 * 10^6 * 11 * 0,16ms \approx 1h$
- Zusammen $\approx 3d$

Reduktion der Anzahl der Durchgänge

Mit B Seiten im Puffer können auch $B-1$ Seiten gemischt werden (eine Seite dient als Schreibpuffer)

Pass n (Input: $\frac{\lceil N/B \rceil}{(B-1)^{n-1}}$ sorted runs; Output: $\frac{\lceil N/B \rceil}{(B-1)^n}$ sorted runs)

1. Open $B - 1$ runs $r_1 \dots r_{B-1}$ from Pass $n - 1$ for reading.
2. **Merge** records from $r_1 \dots r_{B-1}$, reading input page-by-page.
3. **Write** new $B \cdot (B - 1)^n$ -page run to disk (page-by-page).

This pass requires B **pages** of buffer space.

($B-1$)-Wege-Mischen:

- Anzahl der I/O-Operationen:

$$2 \cdot N \cdot (1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil)$$

- Was ist das Zugriffsmuster auf diese Ein-Ausgaben?
- Antwort: In den Mergephasen muss ja entschieden werden, welcher der $B-1$ runs den nächsten Eintrag liefert. Daher wahlfreier Zugriff.

Aufgabe:

Wie lange dauert die Sortierung einer 8GB Datei mit $B=1000$ Pufferseiten je 8KB mit einer Travelstar 7k200?

- 8KB pro Seite, $t_s = 10$ ms, $t_r = 4.17$ ms, $t_{tr} = 0,16$ ms für 8KB-Seite

Anzahl der Durchgänge: $1 + \lceil \log_{B-1} N / B \rceil$

Anzahl der I/O-Operationen:

$$2 \cdot N \cdot (1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil)$$

Lösung

$\approx 16h$

- Durchgänge = $1 + \left\lceil \log_{B-1} \left\lceil N/B \right\rceil \right\rceil =$
 $1 + \left\lceil \log_{999} \left\lceil 10^6 / 10^3 \right\rceil \right\rceil = 1 + \left\lceil \log_{999} 10^3 \right\rceil = 3$
- Mergedurchgänge = 2
- Im ersten Durchgang kann Such- und Rotationszeit vernachlässigt werden
- Damit bleibt nur für die Mergedurchgänge eine Such- und Rotationszeit von insgesamt $2 * 10^6 * 2 * 14,17ms \approx 16 h$
- Gesamte Transferzeit ergibt sich zu
 $2 * 10^6 * 3 * 0,16ms \approx 1min$
- Zusammen macht das $\approx 16h$

Blockweise Ein-Ausgabe

Man kann das I/O-Muster verbessern, in dem man Blöcke von b Seiten in den Mischphasen verarbeitet

- Alloziere b Seiten für jede Eingabe (statt nur eine)
- Reduktion der Ein-Ausgabe um Faktor b pro Seite
- Preis: Reduzierte Einfächerung (was in mehr Durchgängen und damit in mehr I/O-Operationen resultiert)

Aufgabe:

Wie lange dauert die Sortierung einer 8GB Datei mit $B=1000$ Pufferseiten je 8KB und blockweisem IO mit $b=32$ bei einer Travelstar 7k200 ?

- 8KB pro Seite, $t_s = 10$ ms, $t_r = 4.17$ ms, $t_{tr} = 0,16$ ms für 8KB-Seite
- 63 Sektoren pro Spur, Track-to-Track-Suchzeit $t_{s,track-to-track} = 1$ ms
- Ein Block mit 8 KB benötigt 16 Sektoren

Anzahl der Durchgänge: $1 + \lceil \log_{\lceil B/b \rceil - 1} \lceil N / \lceil B/b \rceil \rceil$

Anzahl der I/O-Operationen:

$$2 \cdot \lceil N/b \rceil \cdot (1 + \lceil \log_{\lceil B/b \rceil - 1} \lceil N / \lceil B/b \rceil \rceil)$$

Lösung

Mit blockweisem I/O ($B=1000$, Blöcke mit $b=32$ Seiten):
ca. $10 \cdot 31250$ Plattenzugriffe mit je 27.42 ms
 $\approx 2,38$ h

(Herleitung: Hausaufgabe)

Hauptspeicher als Ressource

In der Praxis meist genügend Hauptspeicher vorhanden, so dass Dateien in einem (!) Mischdurchgang sortiert werden kann (selbst bei blockweisem I/O).

Aufgabe:

Wieviel Hauptspeicher wird bei Pufferseiten mit je 8KB und blockweisem IO mit $b=32$ benötigt, so dass ein Mischvorgang für eine 8GB Datei reicht?

Anzahl der Durchgänge: $1 + \lceil \log_{\lceil B/b \rceil - 1} \lceil N / \lceil B/b \rceil \rceil$

$$\lceil \log_{\lceil B/b \rceil - 1} \lceil N / \lceil B/b \rceil \rceil = 1$$

Lösung

Platzbedarf: 256MB

(Herleitung: Hausaufgabe)

Bisher nur I/O-Zeit betrachtet

- Auswahl des nächsten Datensatzes für den Output unter $B-1$ (oder $B/b - 1$) Eingabeläufen kann CPU-intensiv sein ($B-2$ Vergleiche, wie genau?)
- Beispiel: $B-1 = 4, <$ -Ordnung

087 503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
(out buffer)

- Aktuelles minimales Element m
- $m_1 = \text{first}(\text{run}_1)$
- $m_{n+1} = \min\{m, \text{first}(\text{run}_{n+1})\}$
- $m_1 = 087$
- $m_2 = \min\{087, 170\} = 087$
- $m_3 = \{087, 154\} = 087$
- $m_4 = \{087, 612\} = 087$
- (drei Vergleiche)

Bisher nur I/O-Zeit betrachtet

- Auswahl des nächsten Datensatzes für den Output unter $B-1$ (oder $B/b - 1$) Eingabeläufen kann CPU-intensiv sein ($B-2$ Vergleiche, wie genau?)
- Beispiel: $B-1 = 4, <$ -Ordnung

087 503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
 (out buffer)

Bisher nur I/O-Zeit betrachtet

- Auswahl des nächsten Datensatzes für den Output unter $B-1$ (oder $B/b - 1$) Eingabeläufen kann CPU-intensiv sein ($B-2$ Vergleiche)
- Beispiel: $B-1 = 4, <$ -Ordnung

087 503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
(out buffer)

503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
087 (out buffer)

Bisher nur I/O-Zeit betrachtet

- Auswahl des nächsten Datensatzes für den Output unter $B-1$ (oder $B/b - 1$) Eingabeläufen kann CPU-intensiv sein ($B-2$ Vergleiche)
- Beispiel: $B-1 = 4, <$ -Ordnung

087 503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
(out buffer)

503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
087 (out buffer)

503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
426 653
612 613 700 ... (page Run B-1)
087 154 (out buffer)

Bisher nur I/O-Zeit betrachtet

- Auswahl des nächsten Datensatzes für den Output unter $B-1$ (oder $B/b - 1$) Eingabeläufen kann CPU-intensiv sein ($B-2$ Vergleiche)
- Beispiel: $B-1 = 4, <$ -Ordnung

087 503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
(out buffer)

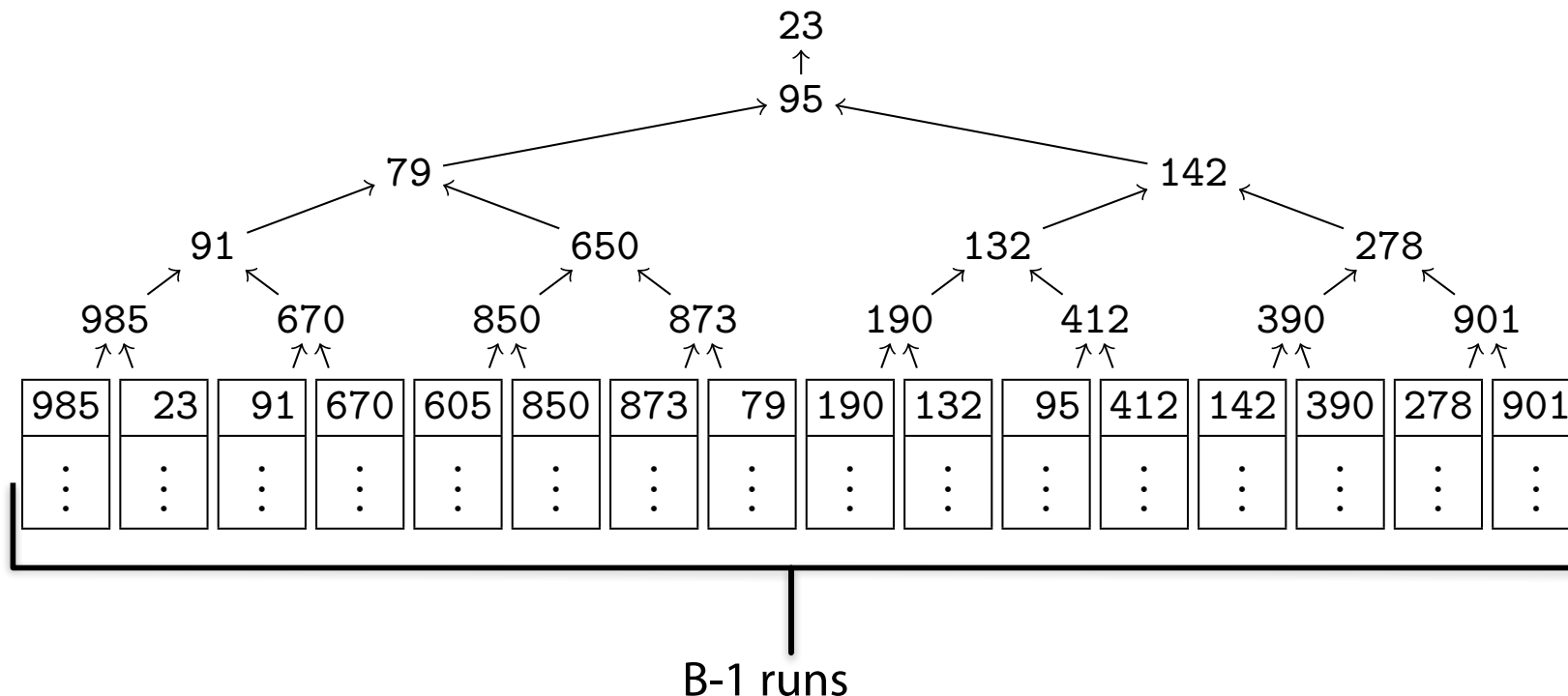
503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
426 653
612 613 700 ... (page Run B-1)
087 154 (out buffer)

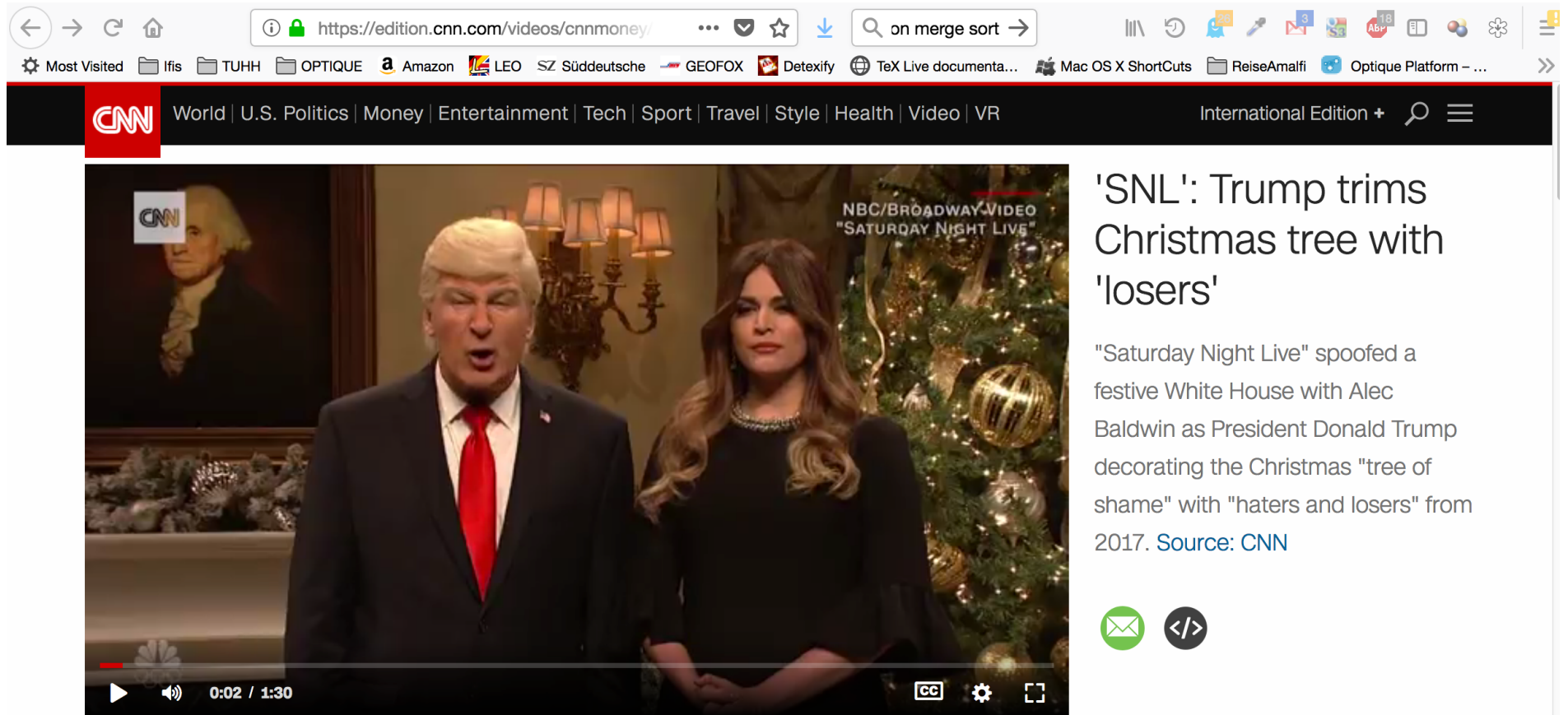
503 504 ... (page from Run 1)
170 908 994 ... (page from Run 2)
154 426 653
612 613 700 ... (page Run B-1)
087 (out buffer)

503 504 ... (page from Run 1)
908 994 ... (page from Run 2)
426 653
612 613 700 ... (page Run B-1)
087 154 170 (out buffer)

Bisher nur I/O-Zeit betrachtet

- Verwende Auswahlbaum zur Kostenreduktion („Tree of Losers“)
- Reduktion der Vergleiche auf $\log_2 (B-1)$ bzw. $\log_2 (B/b-1)$



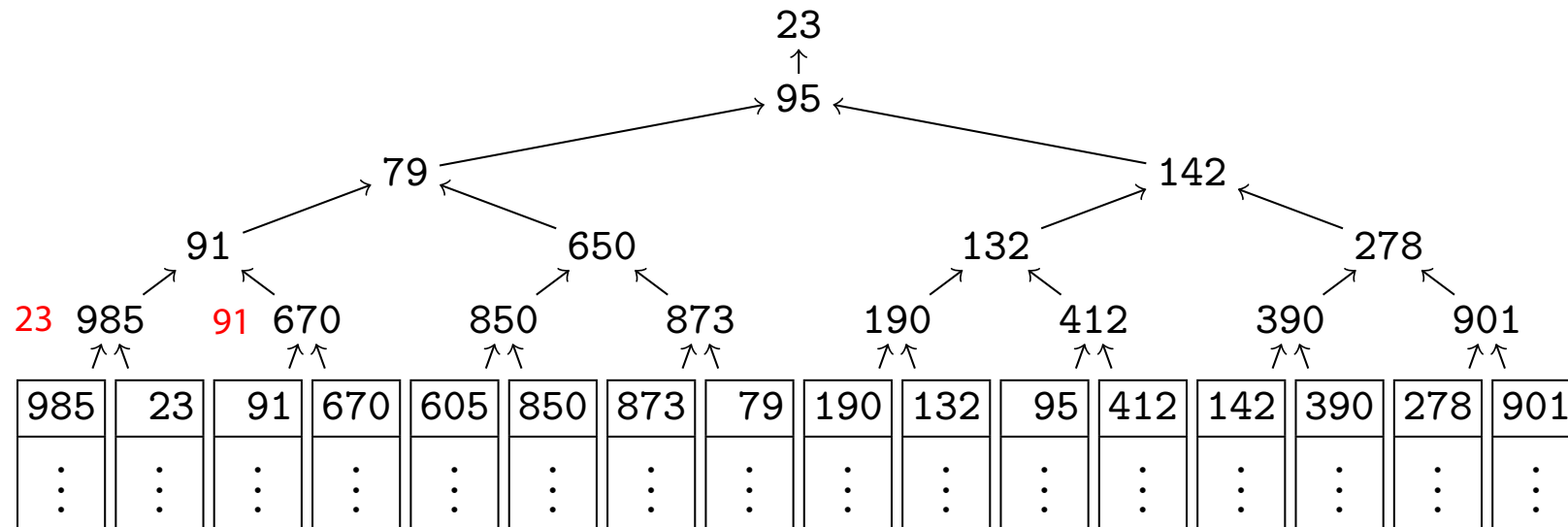


'SNL': Trump trims Christmas tree with 'losers'

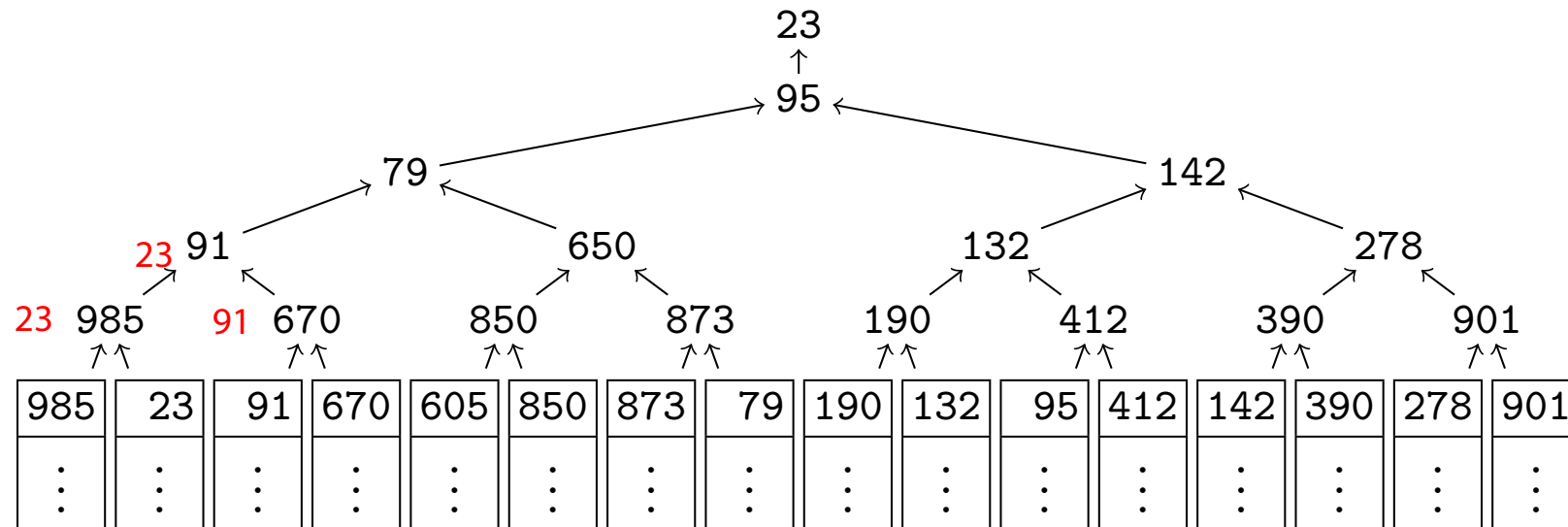


<https://edition.cnn.com/videos/cnnmoney/2017/12/17/snl-trump-christmas-tree-omarosa-ivanka-orig-cws.cnn>

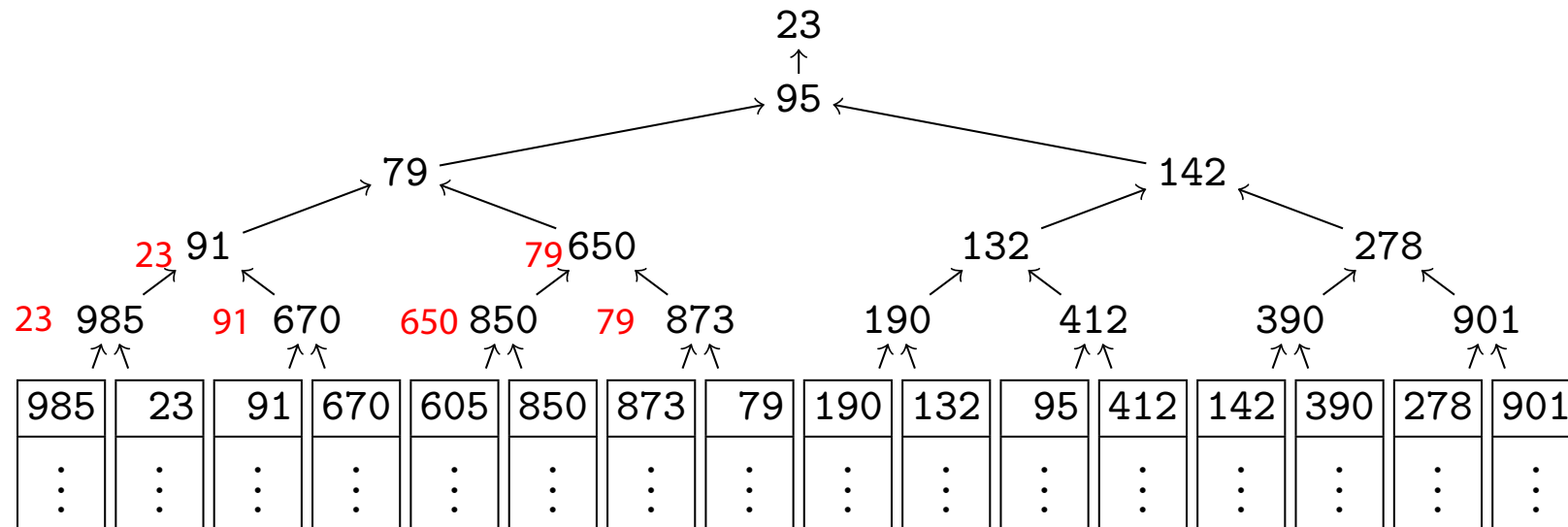
Tree of Losers (and hidden winners)



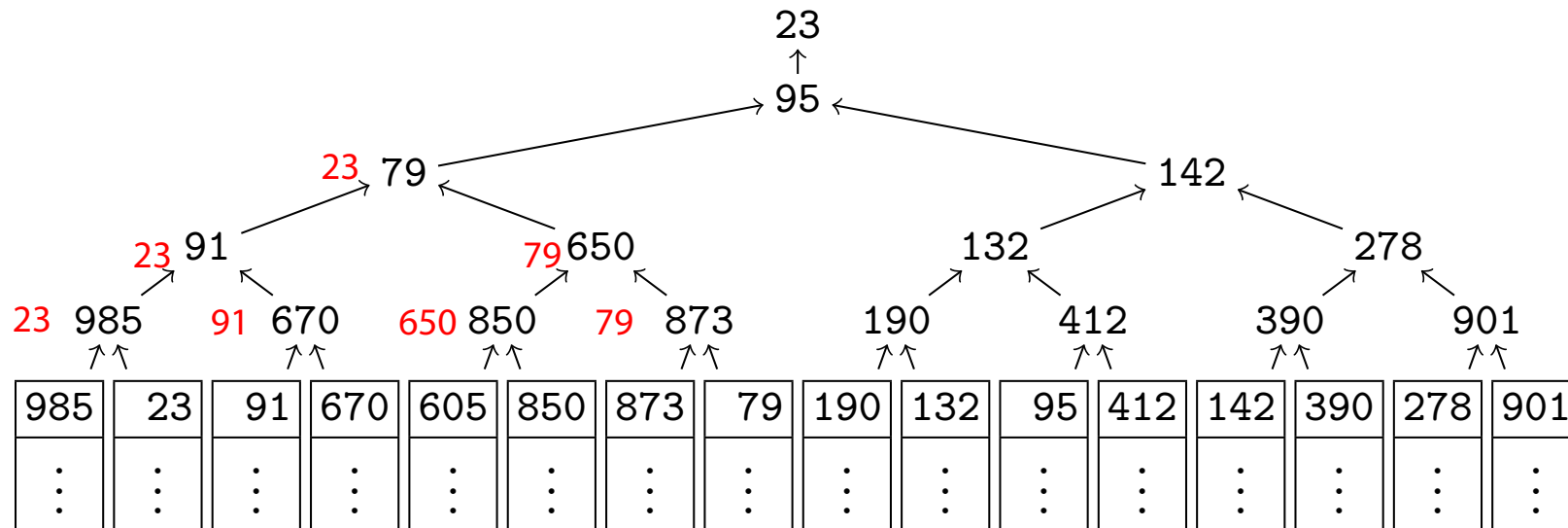
Tree of Losers (and **hidden winners**)



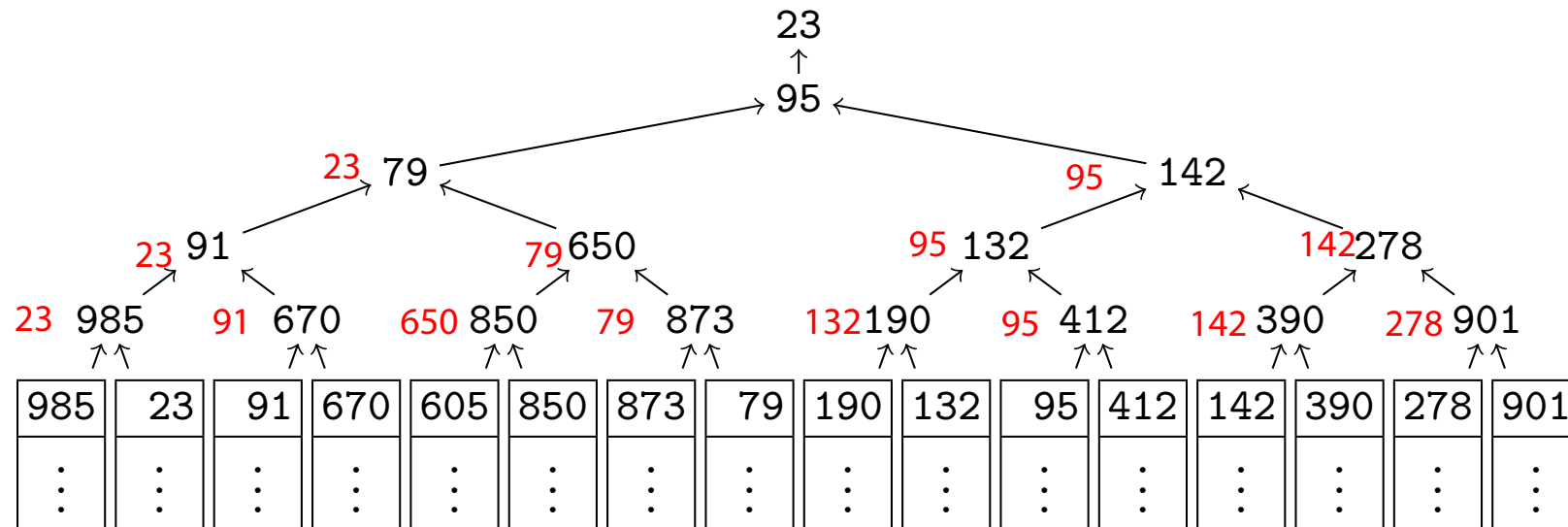
Tree of Losers (and **hidden winners**)



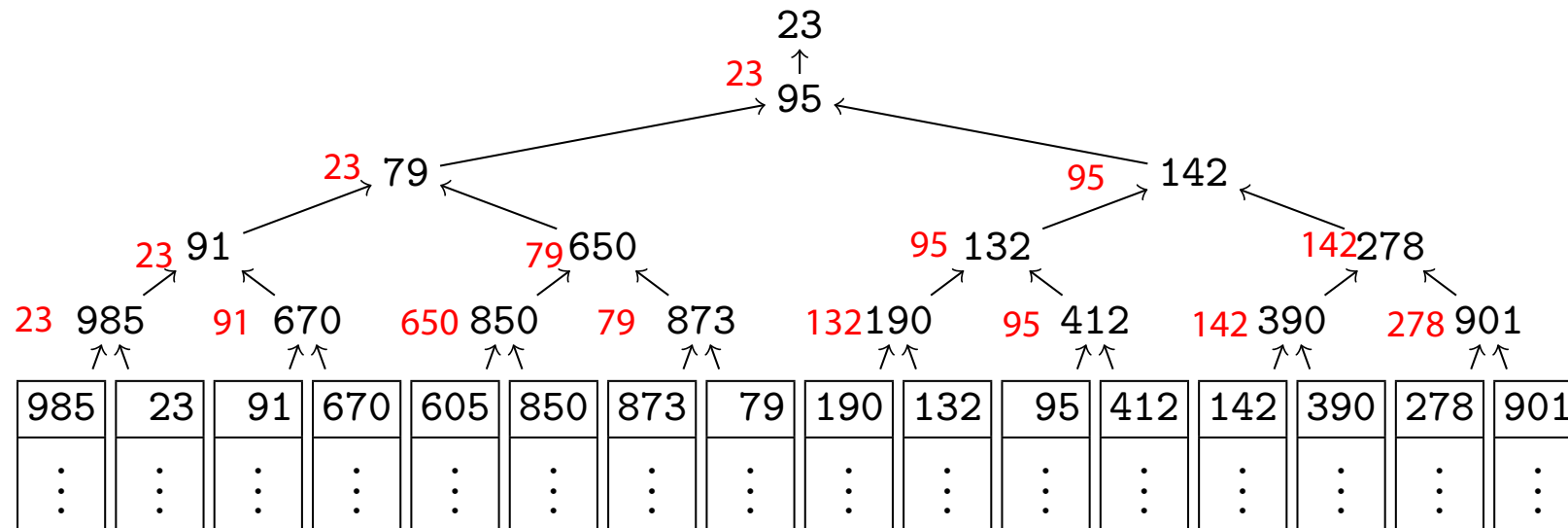
Tree of Losers (and **hidden winners**)



Tree of Losers (and **hidden winners**)

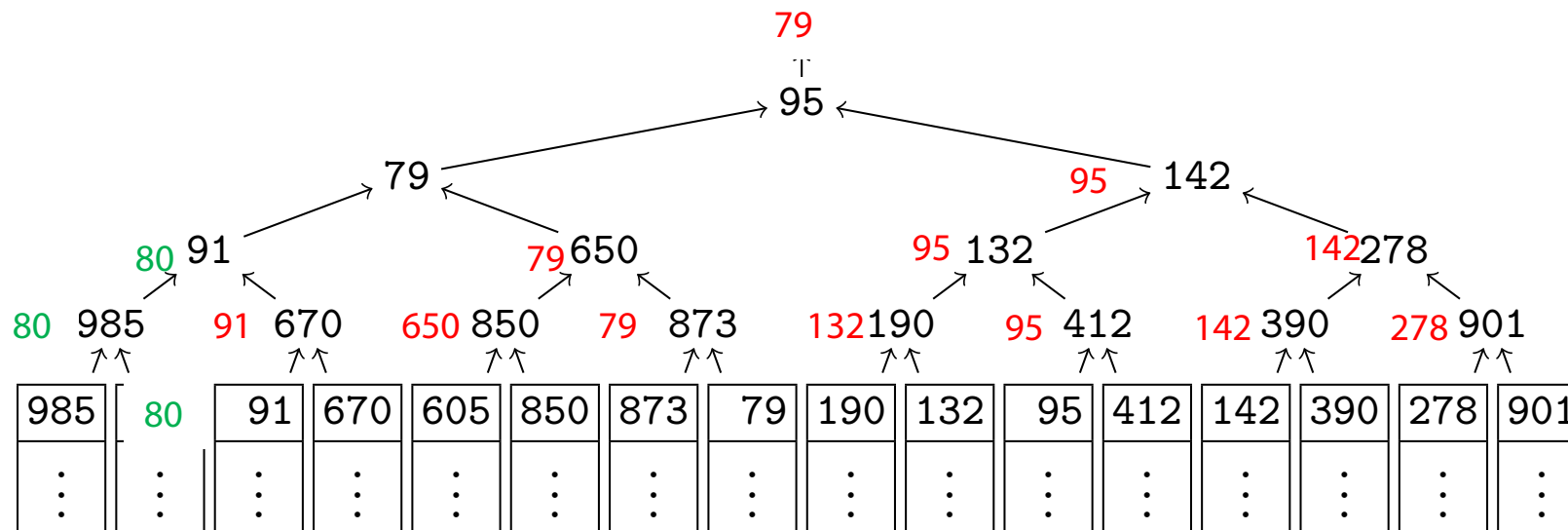


Tree of Losers and **hidden winners (not stored)**



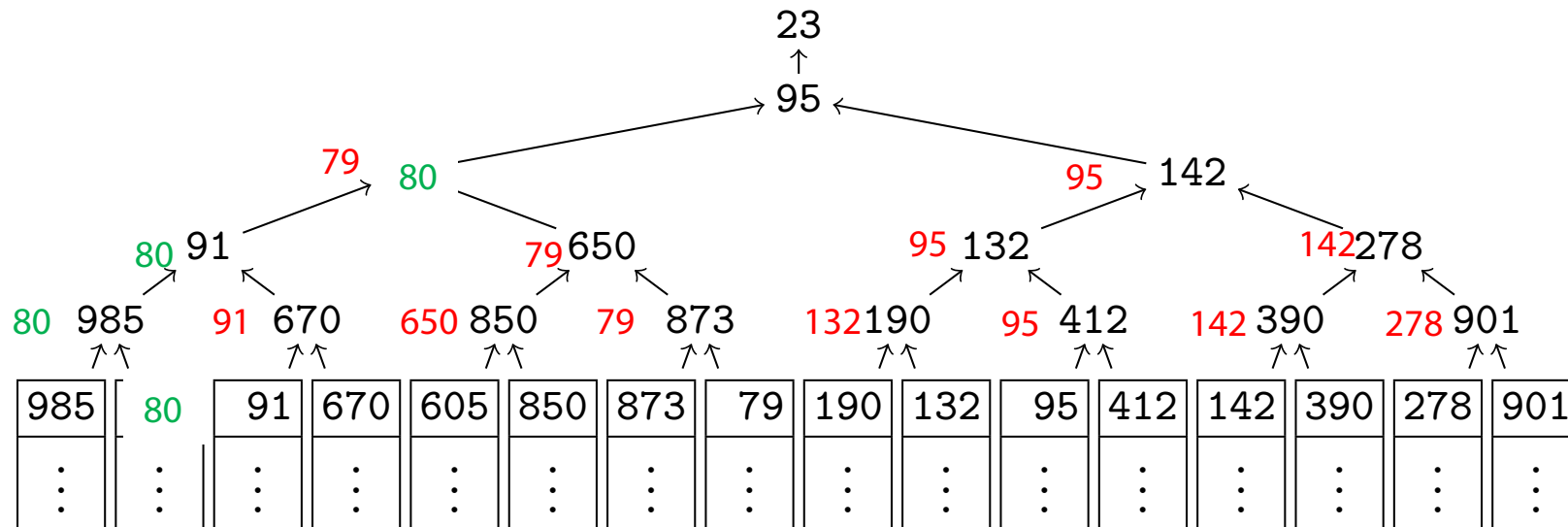
Nächstes Element (80) in Run vom Winner

Muss nicht die versteckten Gewinner betrachten (2 Referenzschritte: Self->parent-> sibling)
Sondern nur Vaterknoten (1 Referenzschritt: self-> parent)



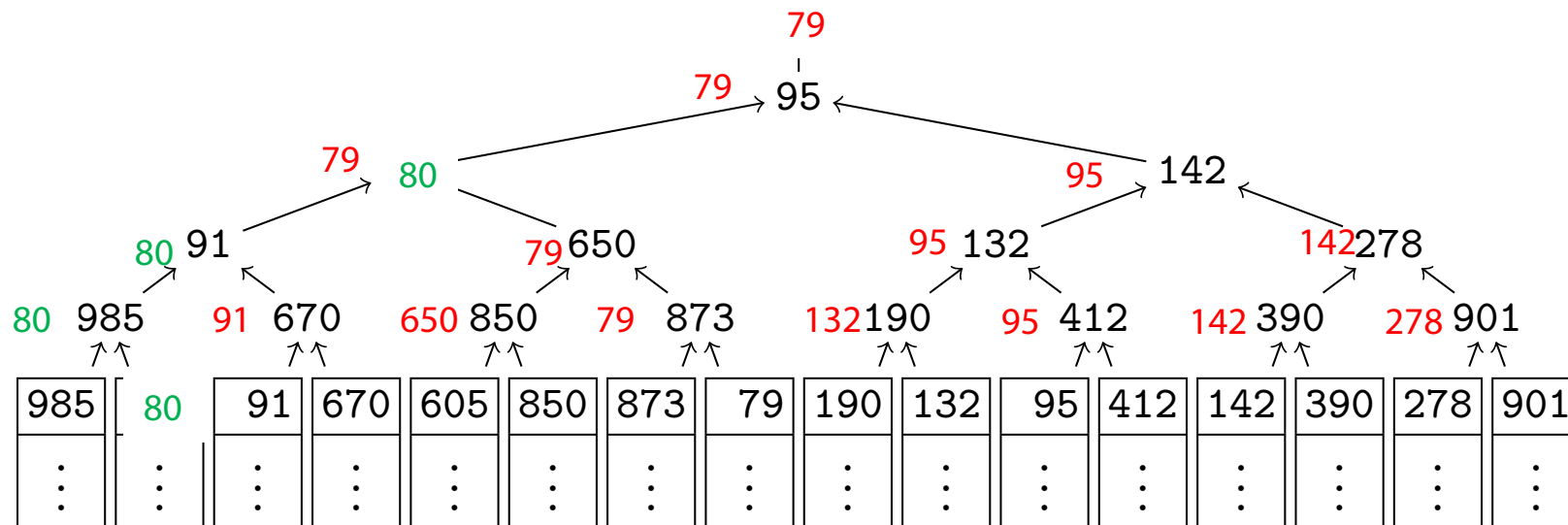
Nächstes Element in Run vom Winner

Muss nicht die versteckten Gewinner betrachten (2 Referenzschritte: Self->parent-> sibling)
Sondern nur Vaterknoten (1 Referenzschritt: self-> parent)



Nächstes Element in Run vom Winner

Muss nicht die versteckten Gewinner betrachten (2 Referenzschritte: Self->parent-> sibling)
Sondern nur Vaterknoten (1 Referenzschritt: self-> parent)



Warum nicht ein „tree of winners“?

- Tree of losers lässt sich leicht updaten: Pfad von neuem Element zur Wurzel ohne auf Geschwisterknoten zu schauen.

Externes Sortieren: Diskussion

- Misch-Schritte können auch **parallel** ausgeführt werden
- Bei ausreichend Speicher **reichen zwei Durchgänge (initial + 1 Mischdurchgang)** auch für große Dateien
- Mögliche **Optimierungen**:
 - **Doppelpufferung**: Verschränkung des Seitenladevorgangs und der Verarbeitung, um Latenzzeiten der Festplattenspeicher zu kaschieren
 - **Seitenersetzung während des Sortierens**: Erneutes Laden neuer Seiten wenn initiale Läufe rausgeschrieben werden (dadurch Erhöhen der initialen Lauflänge -> replacement sort)

Replacement Sort: Grundidee

- Puffer mit B Seiten
- Davon einer Input-Buffer (inB)
- Und einer Output-Buffer (outB)
- Rest wird aktuelle Menge (AM) genannt.

Replacement Sort: Algorithmus

1. Leeren Lauf zum Schreiben öffnen
2. $\text{inB} = \text{nächsteSeite}(\text{InputDatei})$
3. Solange Platz in AM, beweg Eintrag aus inB nach AM
(Wenn inB leer, dann nachladen)
4. Finde kleinstes k größer als Maximum k_{out} in outB
 - und schieb k ans Ende von OutB
 - Falls OutB voll, häng es ans Ende vom aktuellen Lauf
5. Falls kein solches k existiert, häng OutB ans Ende vom aktuellen Lauf und schließe es. Öffne neuen Run.
6. Falls Input vollständig gelesen, halte. Ansonsten gehe zu 3.