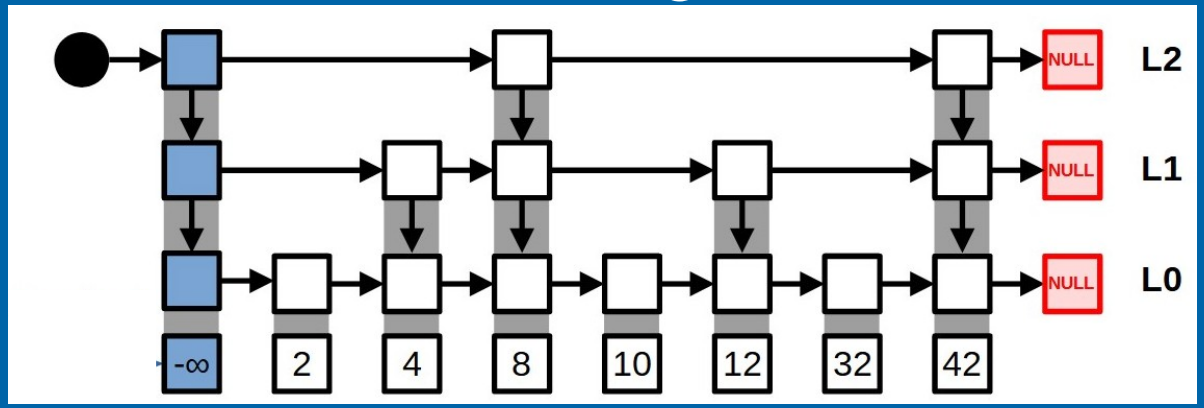
SKIP LIST

Listen mit Abkürzungen



Informationsquellen:

MIT OpenCourseWare: Srinivas Devadas: 7. Randomization: Skip Lists (YouTube)

Shusen Wang: Alg-2C: Skip List (YouTube)

Inhalt des Vortrags

- Assoziative Datenstrukturen
  - Schlüssel
  - Array, Linked List
- Skip List
  - Idee
  - Implementation
  - Code
  - Benchmark Ergebnisse
  - Anwendungen

Assoziative Datenstrukturen

- Einträge sind/enthalten Schlüssel
  - Einzigartig, Vergleichbar
    - Optimal: sortierbar

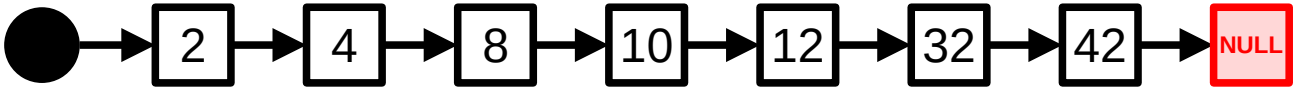
```
{
  "type":"Modul",
  "name":"Seminar: Exploring data structures in C",
  "isInteresting": true
}
```

Linked List

- Unsortiert:
  - Zugriff, Einfügen, Entfernen: O(n)

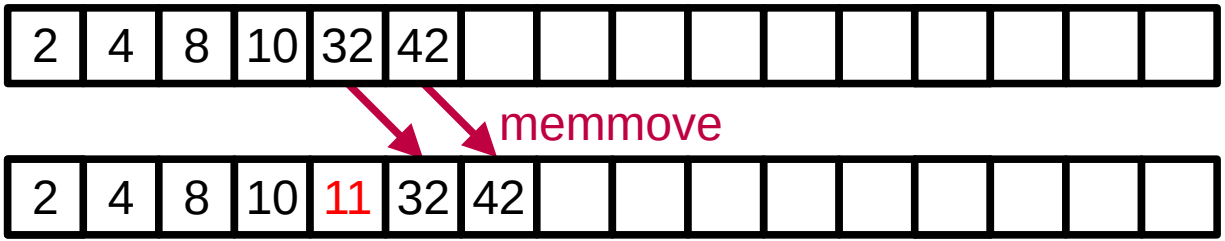
- Sortiert:
  - Zugriff, Einfügen, Entfernen: O(n), doppelt so schnelles erkennen von nicht existierenden Einträgen

am Anfang der Liste: O(1)



Liste in Array

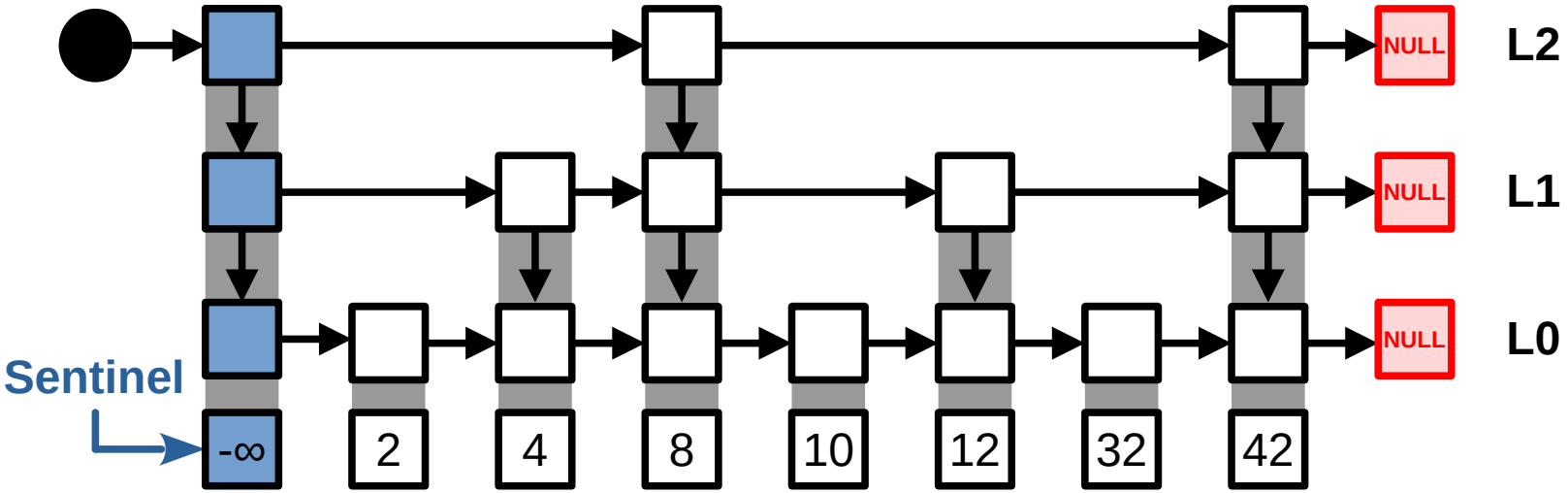
- Suche O(n) (von vorn nach hinten gehen), O(log(n)) binäre suche
- Einfügen: O(n)



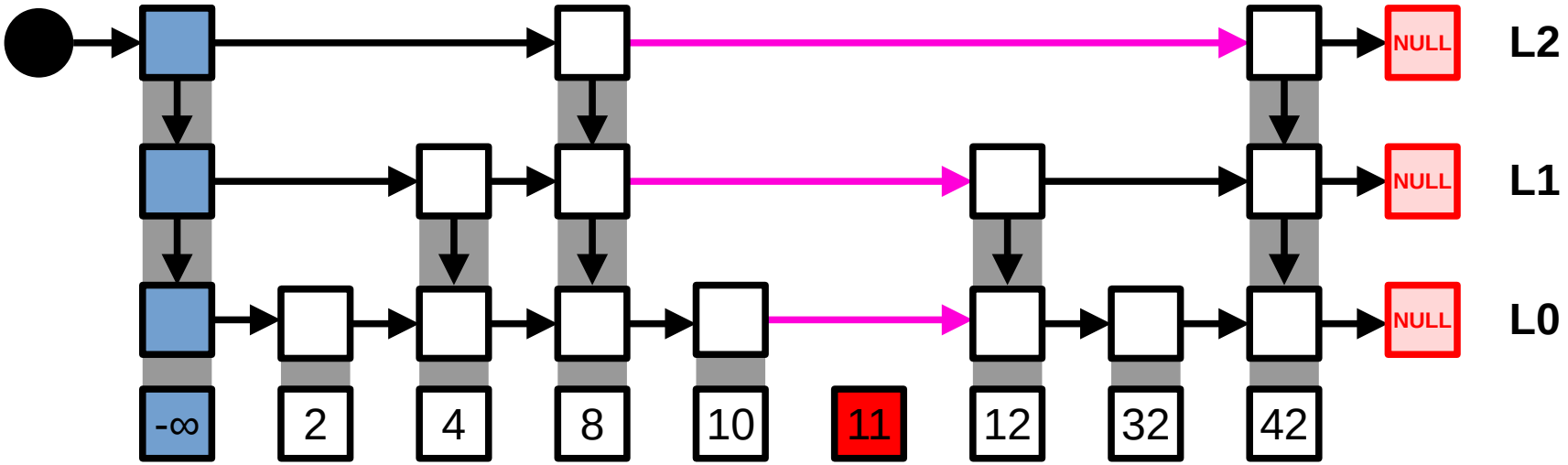
buffer voll: realloc

Skip List – Abstrakte Erklährung

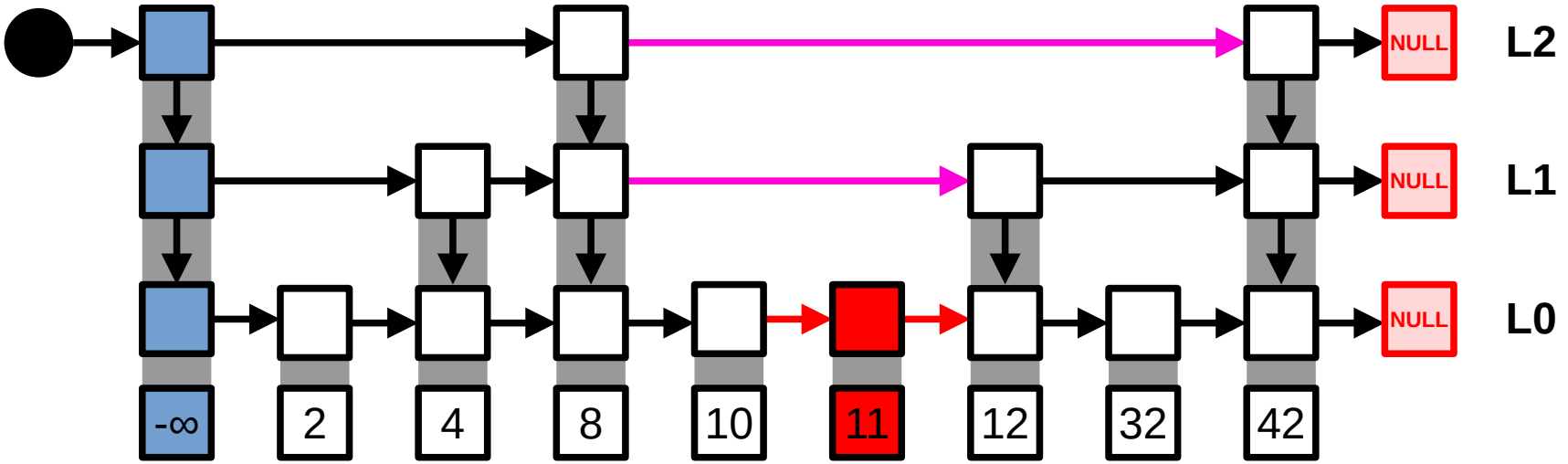
- Zufällig
- Suche O(log(n))



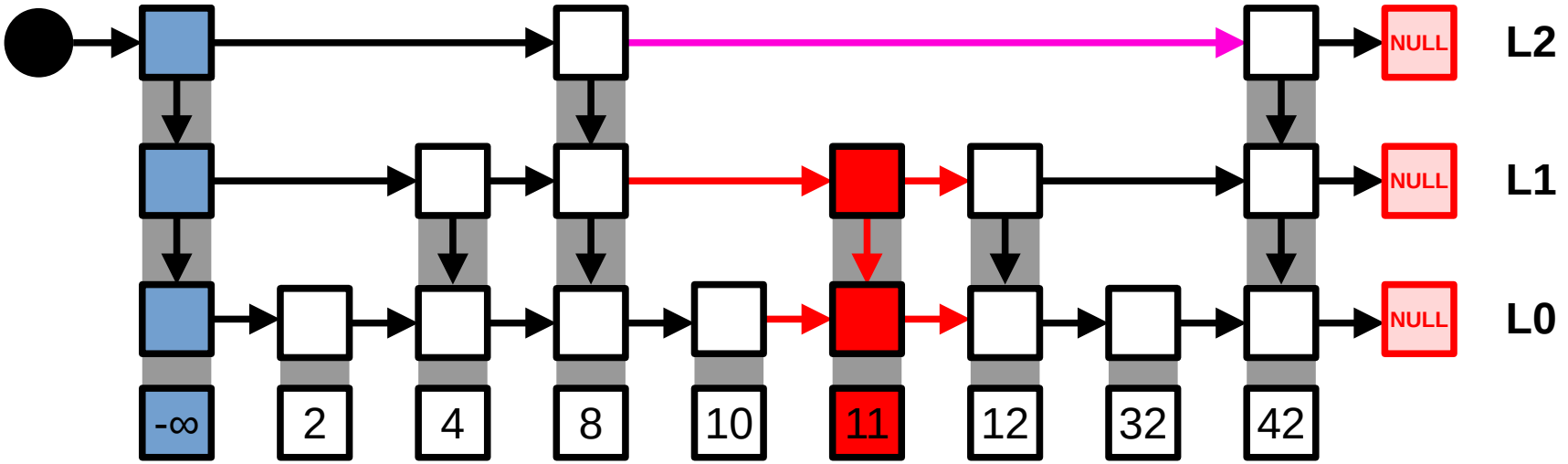
Skip List – Abstrakte Erklährung



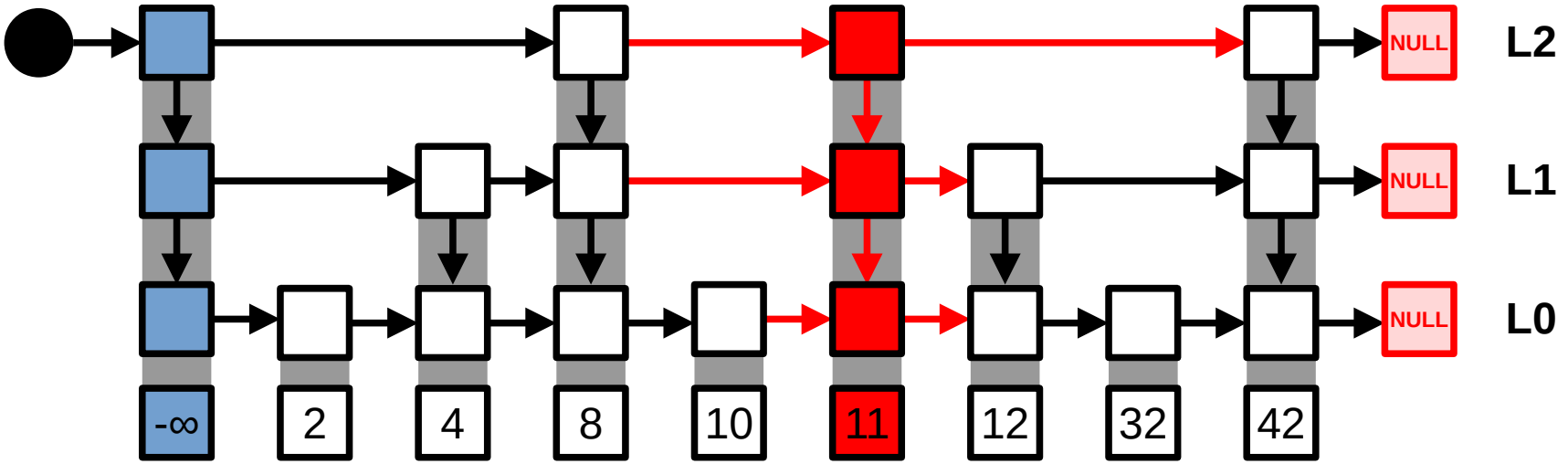
Skip List – Abstrakte Erklährung



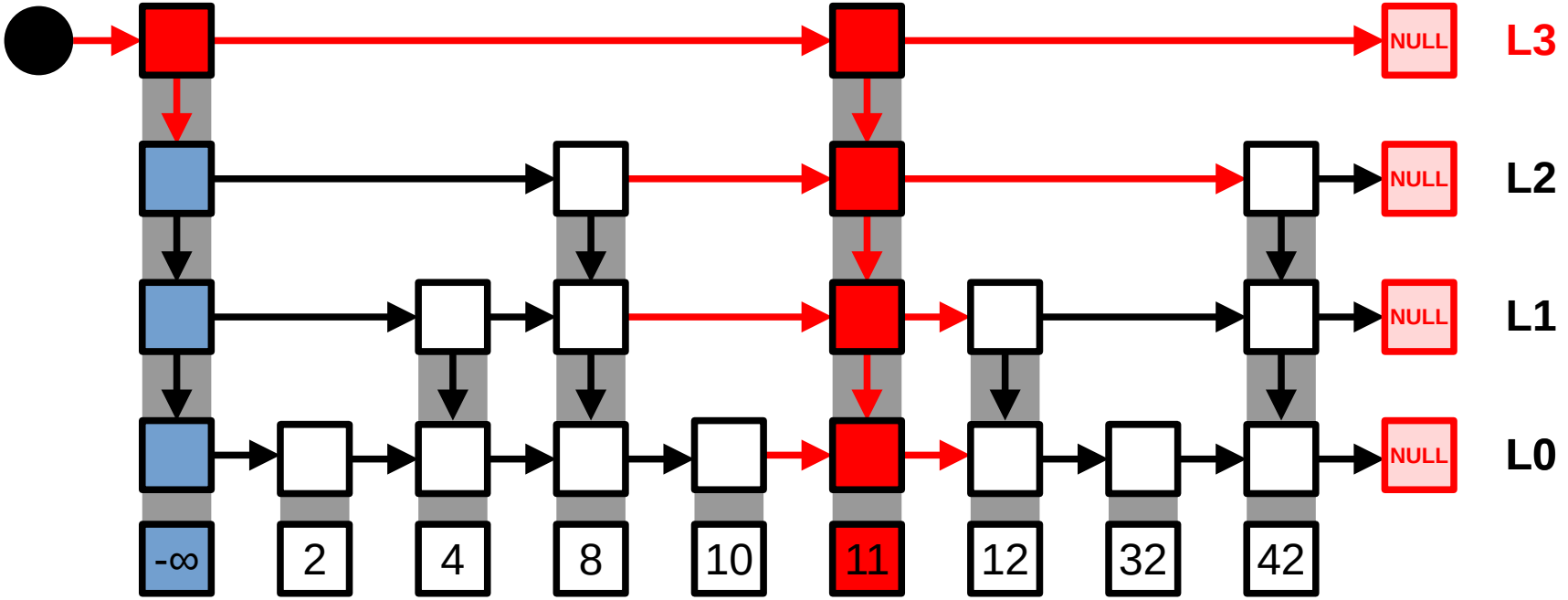
Skip List – Abstrakte Erklährung



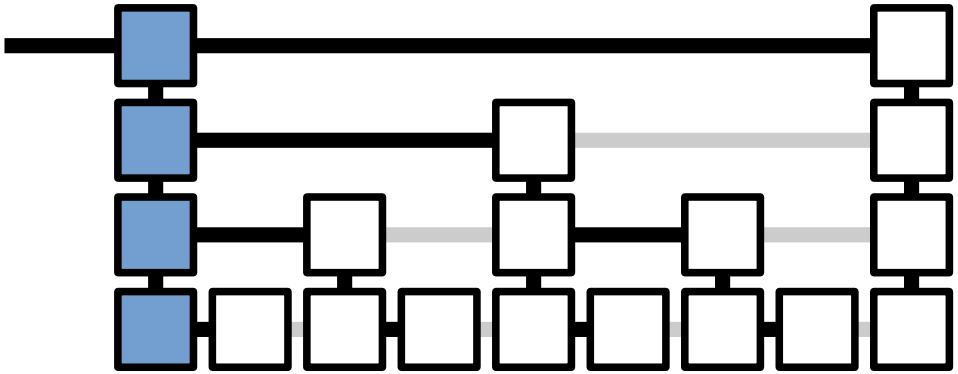
Skip List – Idee



Skip List – Abstrakte Erklährung



Warum O(log(n)) ? (intuitiv)



L3: n/8 Knoten

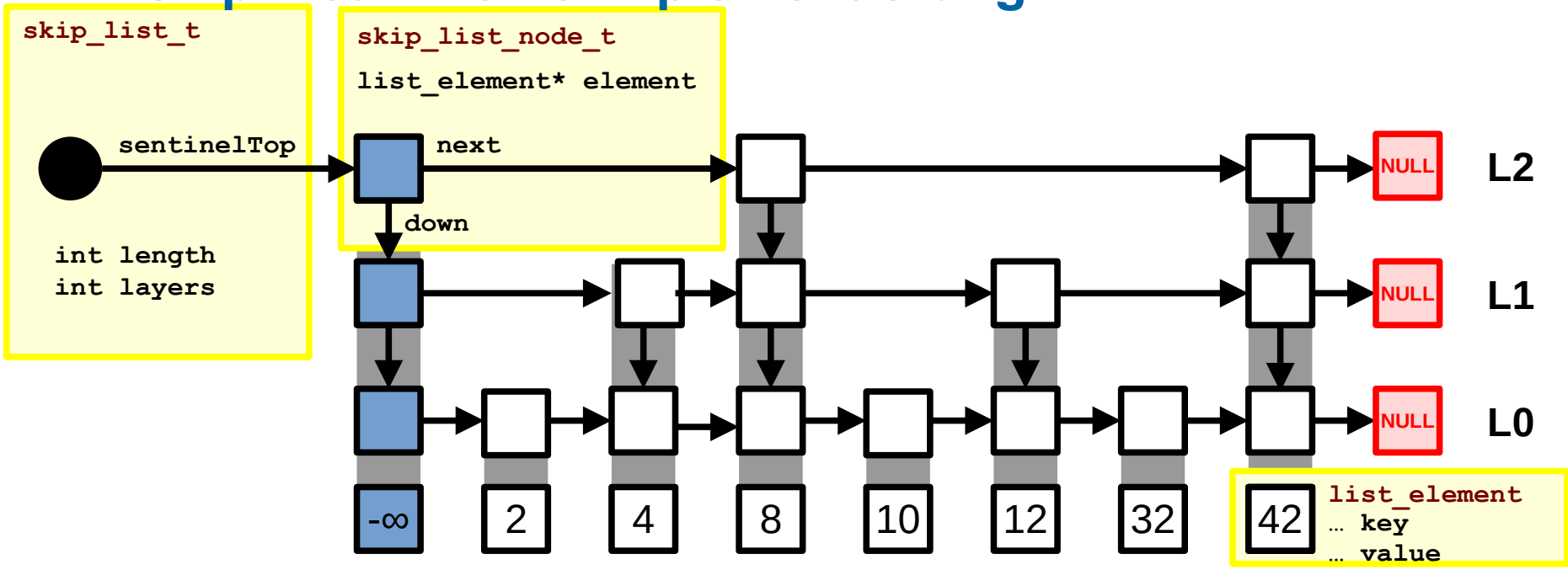
L2: n/4 Knoten

L1: n/2 Knoten

L0: n Knoten

n verdoppeln → +1 Schicht

Skip List – Meine Implementierung

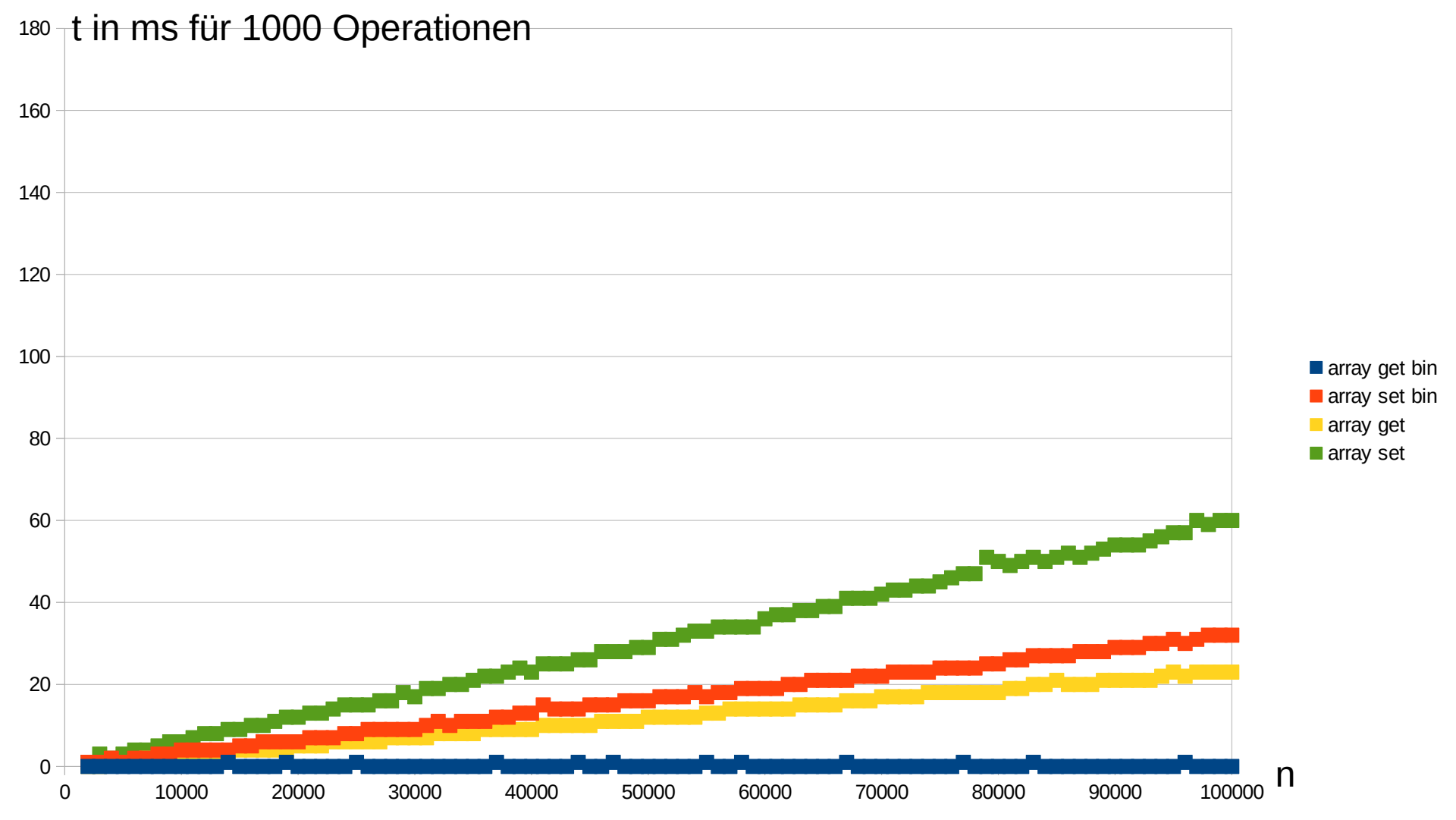


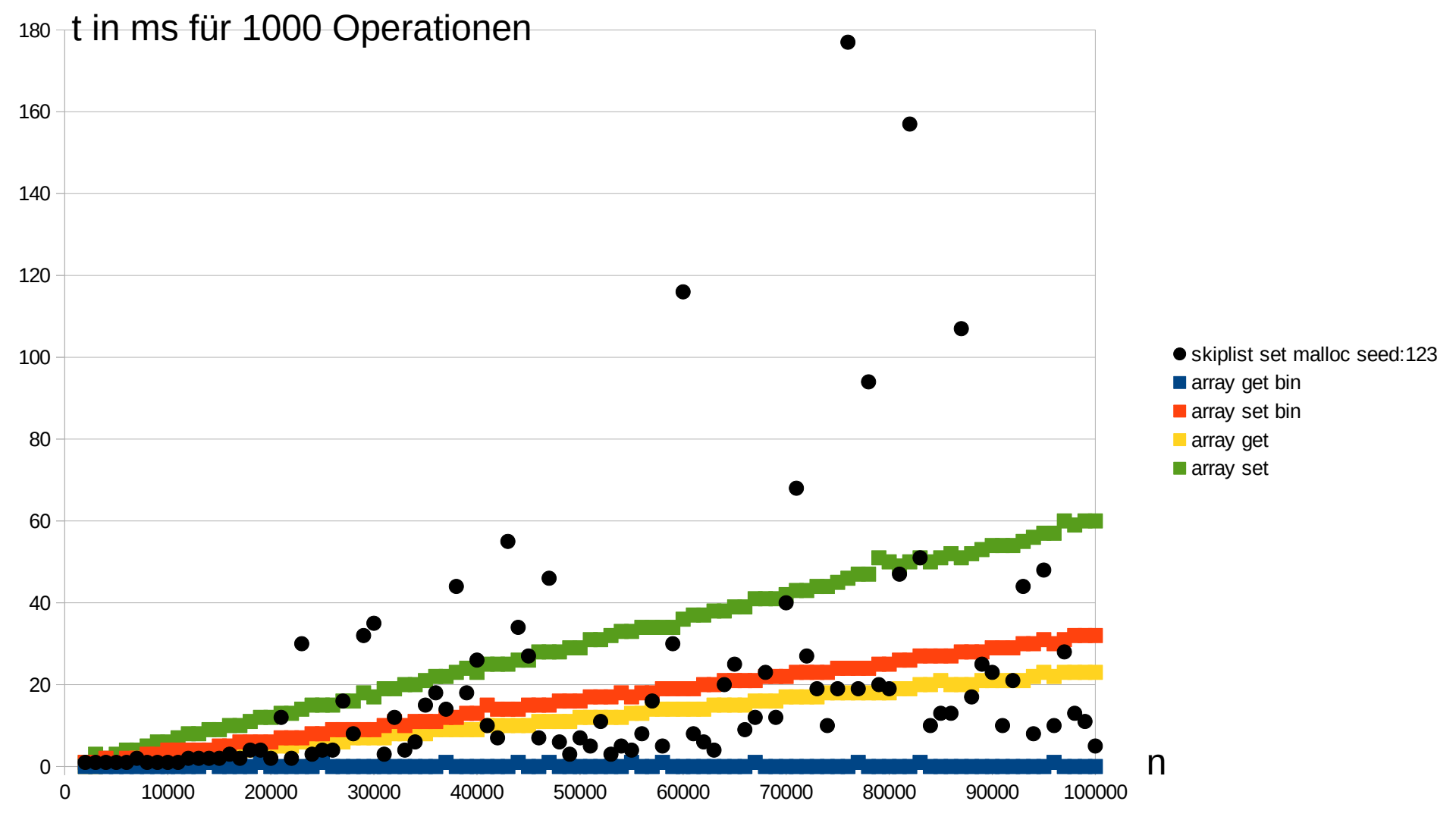
```c
#ifndef LIST_ELEMENT_H
#define LIST_ELEMENT_H


#include <limits.h>

#define LIST_KEY_TYPE int

#define LIST_VALUE_TYPE int
#define LIST_VALUE_NOT_FOUND INT_MIN

// returns negative number if targetKey is after checkKey, returns positive number if checkKey is after targetKey,
if equal
#define LIST_KEY_COMPARE(targetKey, checkKey) ((targetKey) - (checkKey))

#define LIST_KEY_SENTINEL INT_MIN

typedef struct list_element {
    const LIST_KEY_TYPE key;
    LIST_VALUE_TYPE value;
} list_element_t;



#endif
```

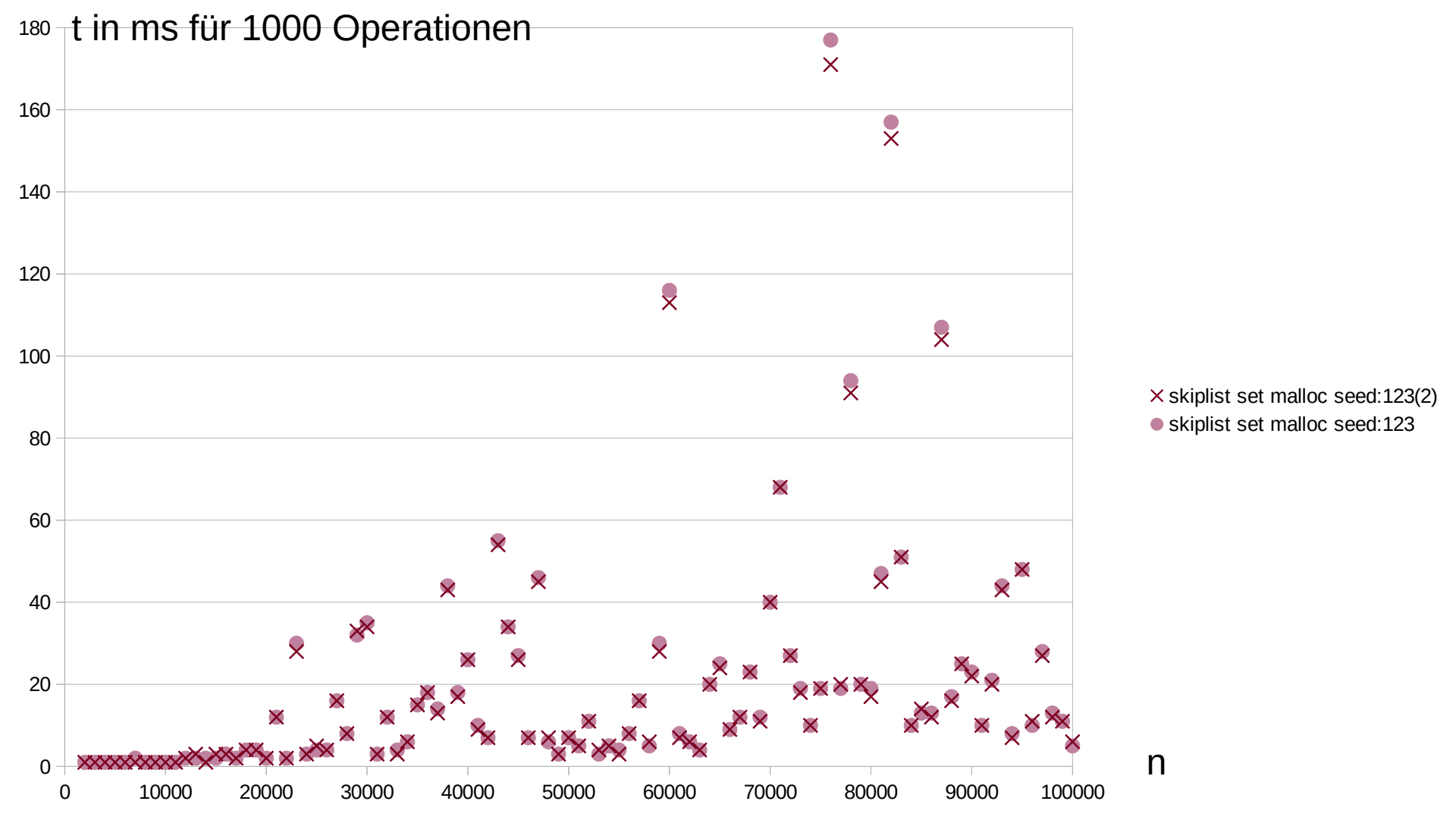
Benchmark

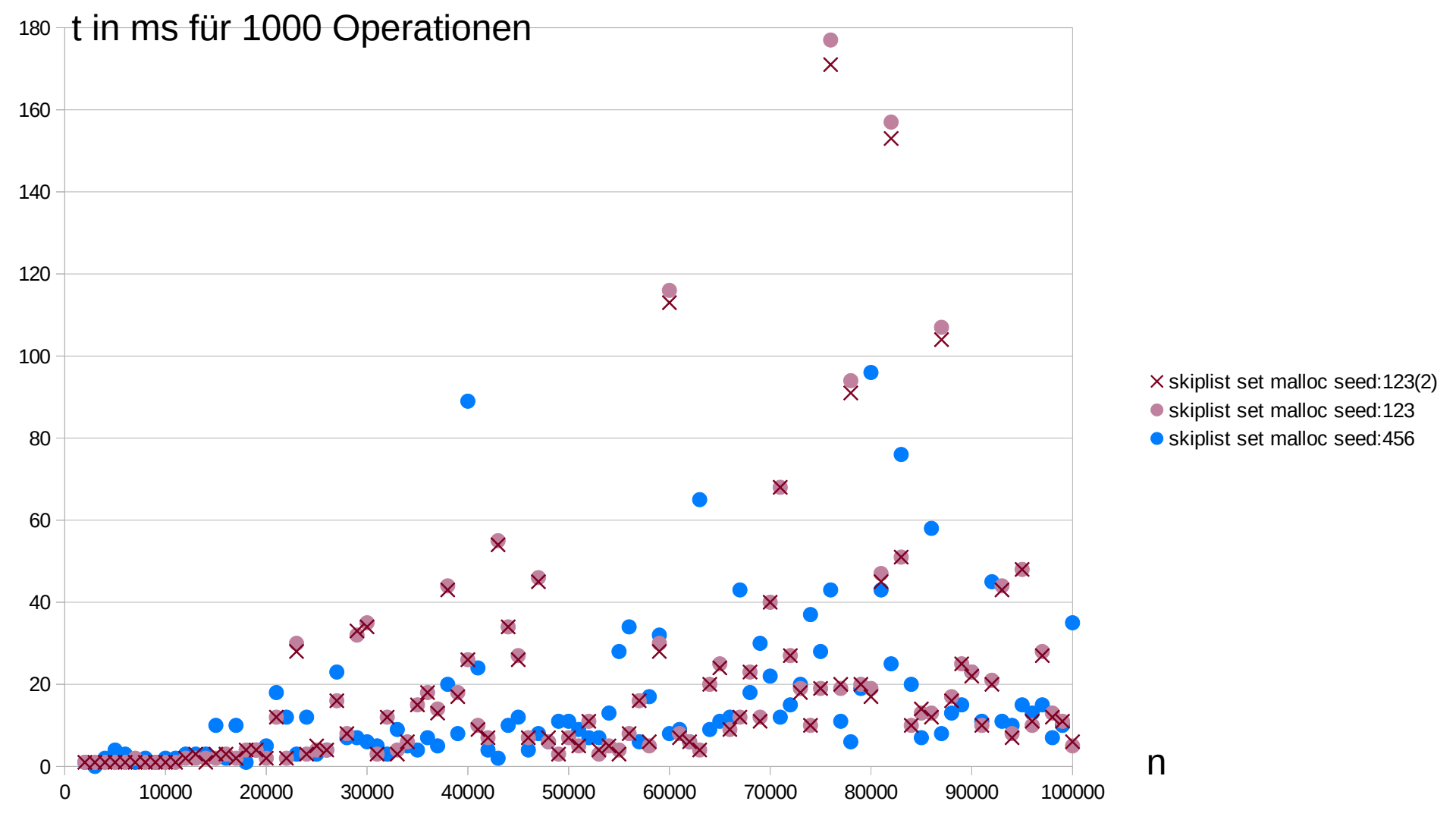
- Schlüssel in Datei
- Vorbereitung: Bauen der Liste mit n Elementen:
  - Schlüssel zwischen 0 und 33554430, einzigartig, gerade
- Test: set,get 1000 mal, Zeit messen
  - Schlüssel zwischen 1 und 33554431, einzigartig, ungerade

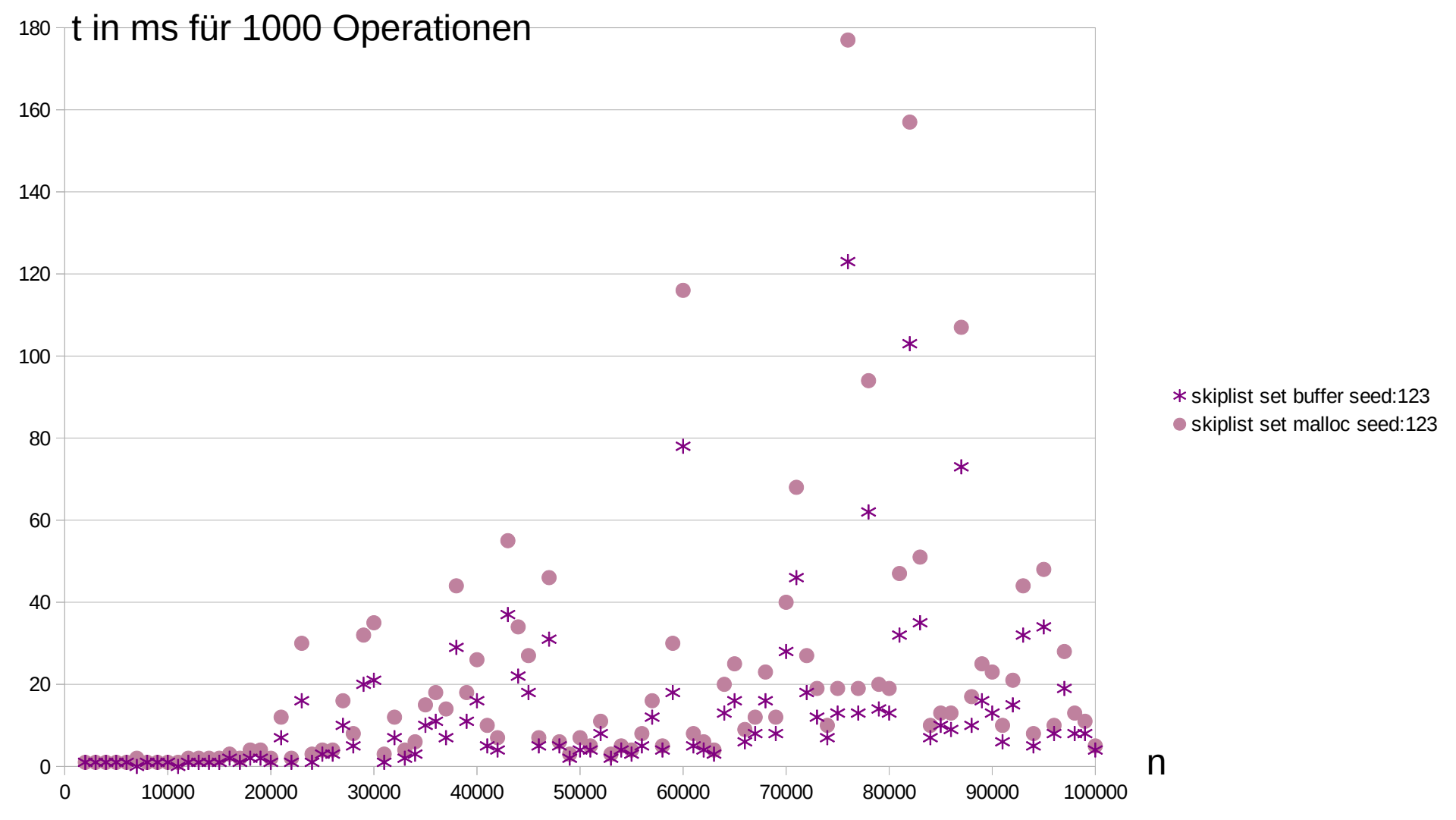
    → nicht in Liste vorhanden

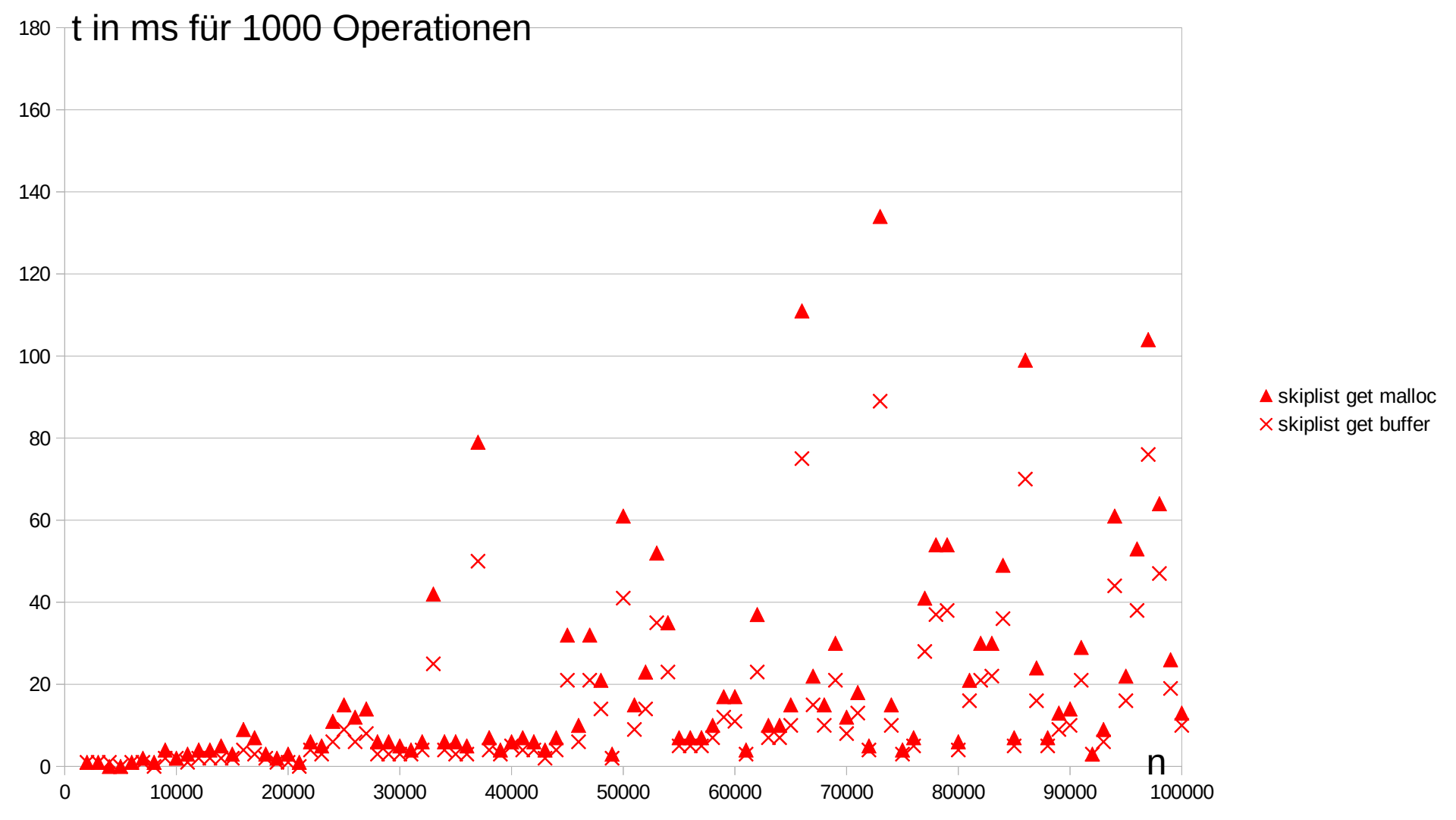

t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
array get bin
array set bin
array get
array set

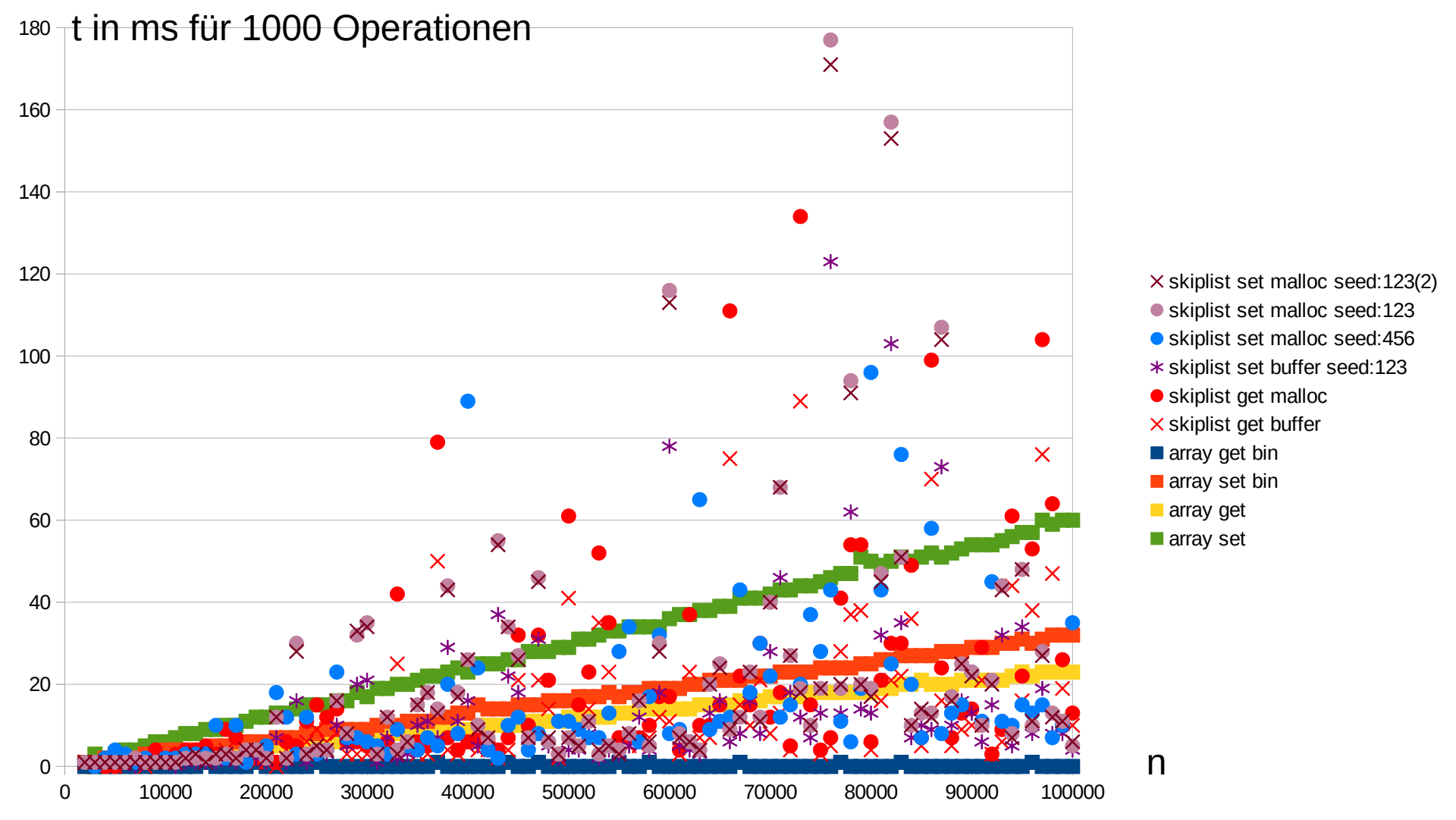

t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
skiplist set malloc seed:123
array get bin
array set bin
array get
array set

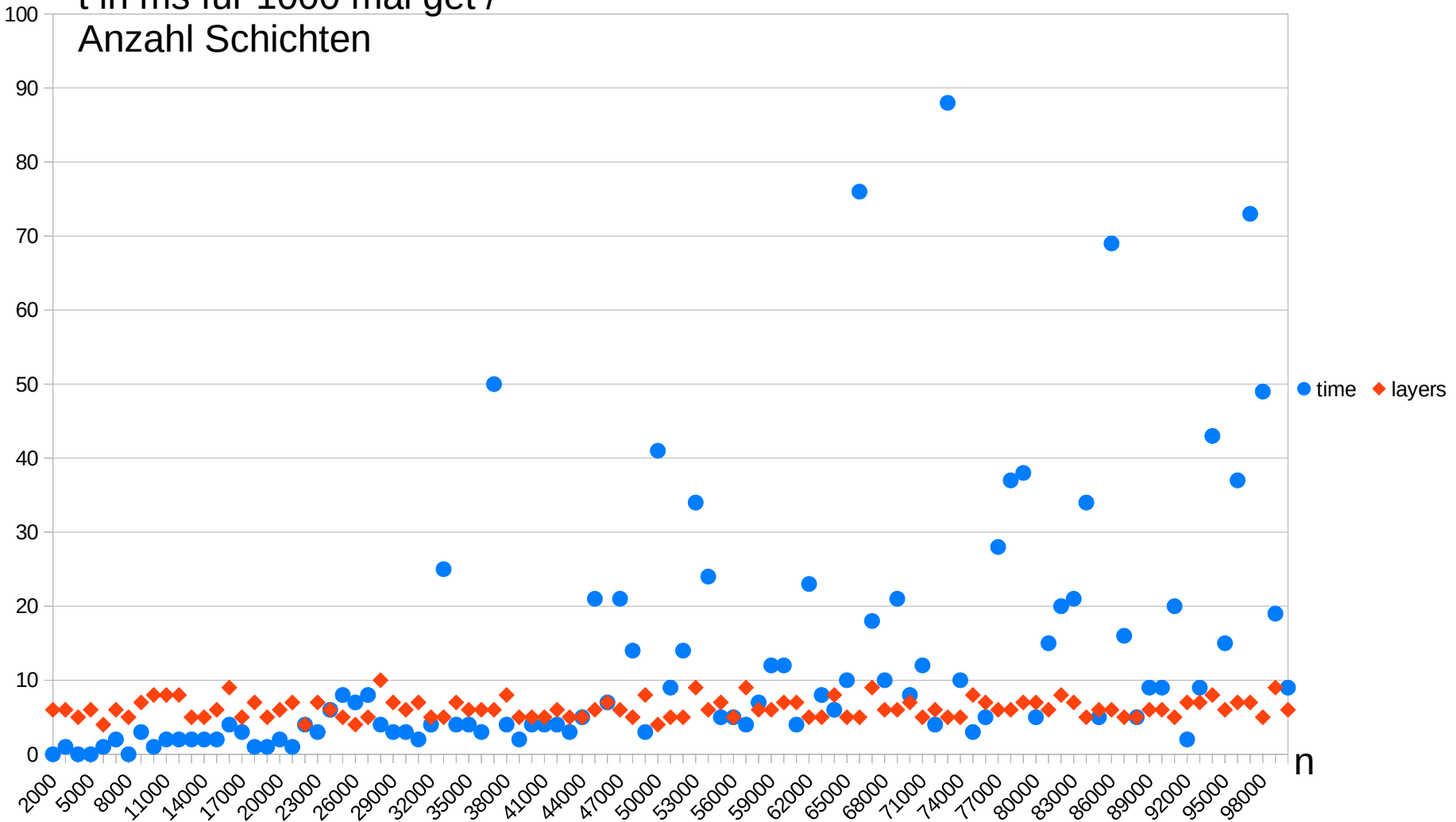

t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
skiplist set malloc seed:123(2)
skiplist set malloc seed:123

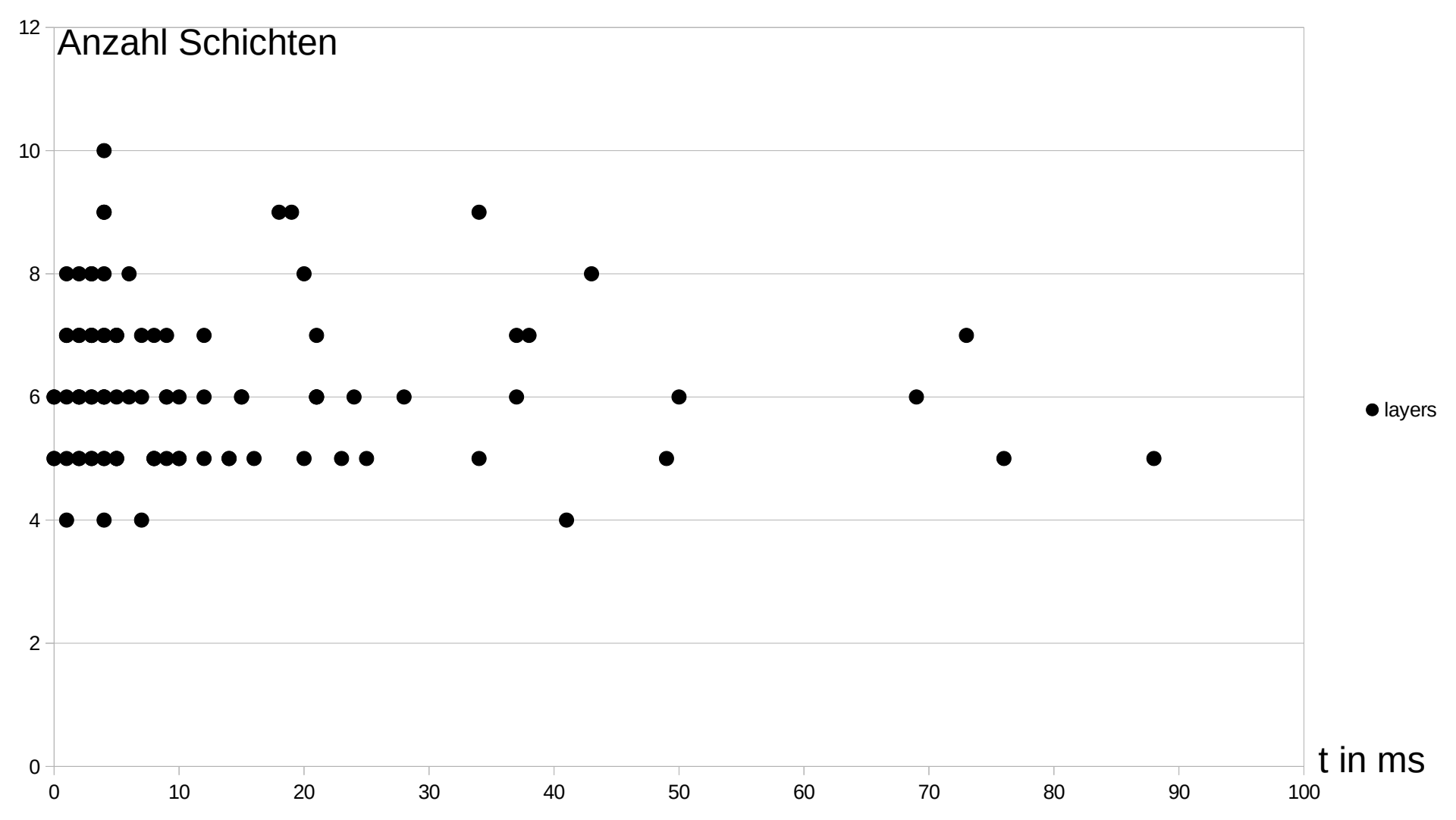

t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
skiplist set malloc seed:123(2)
skiplist set malloc seed:123
skiplist set malloc seed:456


t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
skiplist set buffer seed:123
skiplist set malloc seed:123


t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
skiplist get malloc
skiplist get buffer


t in ms für 1000 Operationen
180
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
n
skiplist set malloc seed:123(2)
skiplist set malloc seed:123
skiplist set malloc seed:456
skiplist set buffer seed:123
skiplist get malloc
skiplist get buffer
array get bin
array set bin
array get
array set


t in ms für 1000 mal get / Anzahl Schichten
100
90
80
70
60
50
40
30
20
10
0
n
2000
5000
8000
11000
14000
17000
20000
23000
26000
29000
32000
35000
38000
41000
44000
47000
50000
53000
56000
59000
62000
65000
68000
71000
74000
77000
80000
83000
86000
89000
92000
95000
98000
time
layers


Anzahl Schichten
12
10
8
6
4
2
0
0
10
20
30
40
50
60
70
80
90
100
t in ms
layers

Anwendungen

- Wo man Assoziative Datenstrukturen braucht
- Datenbank

Gibt es einen besseren Weg die Skip List auf zu bauen? (vielleicht mit weniger Zufall?)

tu-freiberg.de

TU Bergakademie Freiberg bergakademie_freiberg TUBergakademie TUBergakademie

TU BERGAKADEMIE FREIBERG
Universitätskommunikation
Prüferstr. 2
09599 Freiberg
Tel. +49(0)3731 39-2711, -3461
kommunikation@tu-freiberg.de