TECHNISCHE UNIVERSITÄT DRESDEN

VORSTELLUNG EINES NEUEN ALGORITHMUSSES ZUR PARALLELEN SORTIERUNG

Frank Hoffmann

Dresden, 12.7.2011

Problem

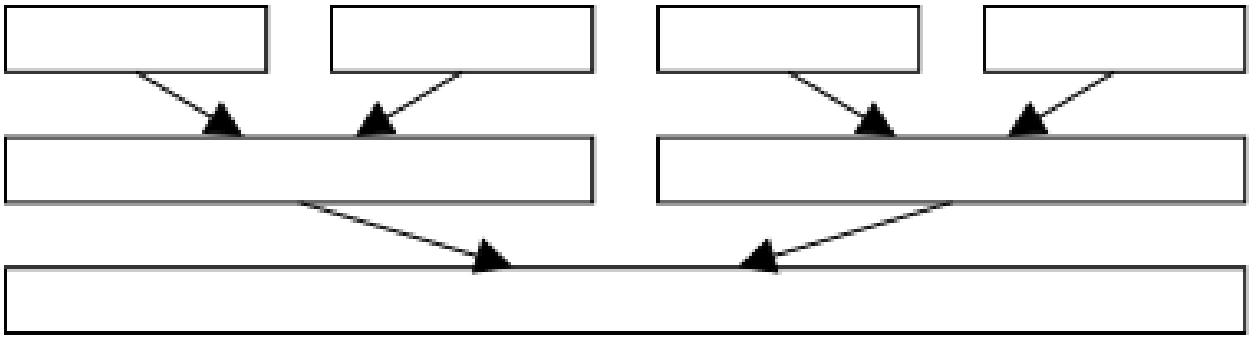
Funktionsweise

Auswertungen

Problem

Funktionsweise

Auswertungen

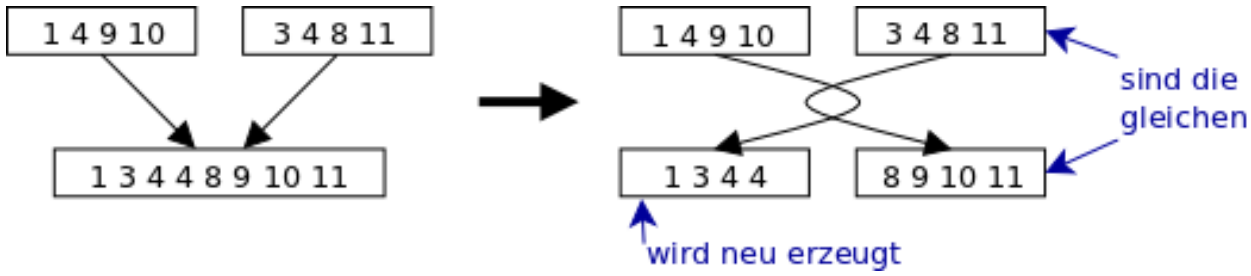


Ursprung der Idee ist der gewöhnliche Mergesort.
Problem bei der Parallelisierung ist das in den letzten Phasen immer weniger Prozessoren ausgelastet werden können.
Die Ursache ist das wir immer weniger Listen zur Verfügung haben.

Problem

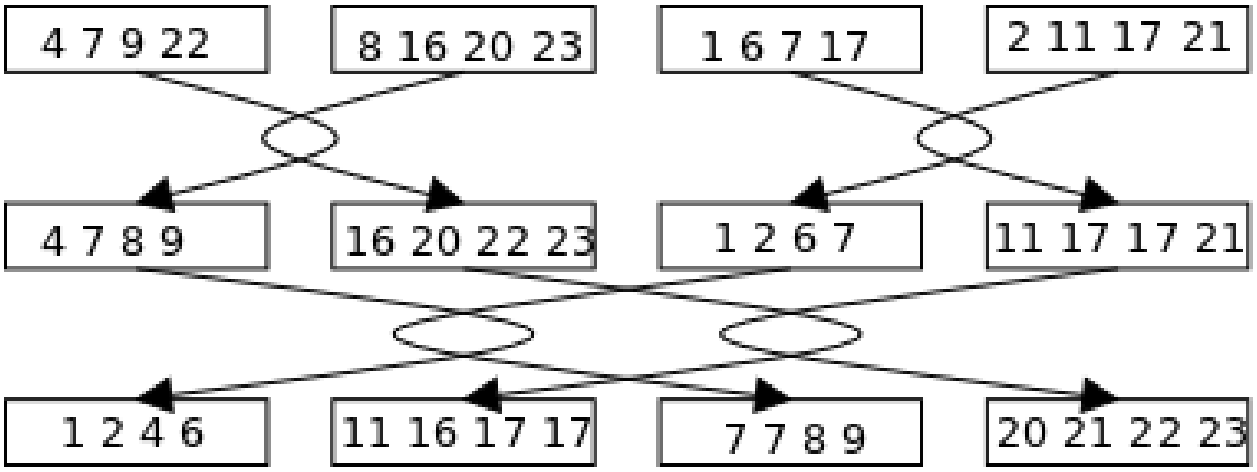
Funktionsweise

Auswertungen

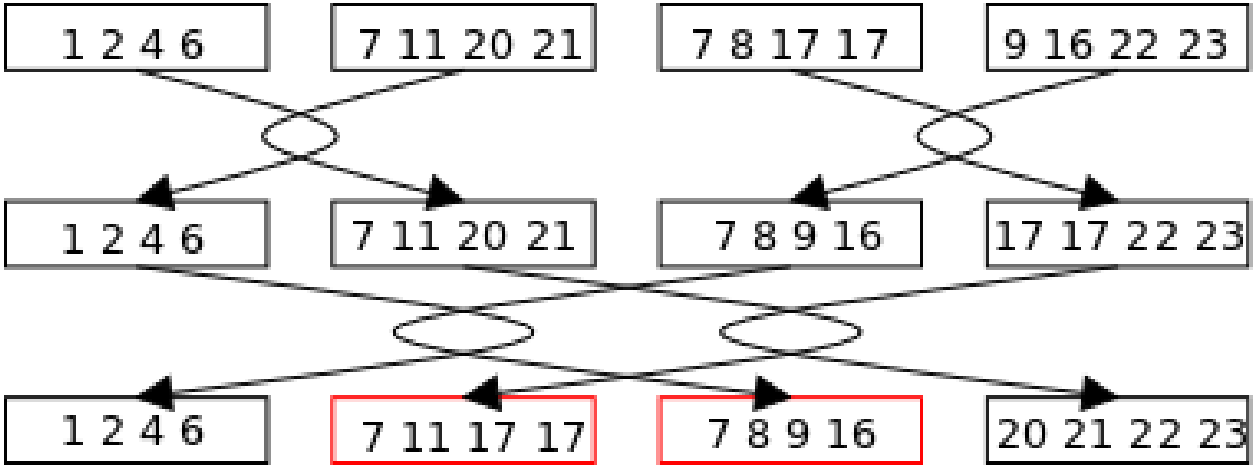


So, damit wäre die Ursache eliminiert.
Weitere Vorteile:

- die zweite Liste können wir wiederbenutzen.
- die letzten Elemente der zweiten Liste bleiben unter umständen unangetastet.

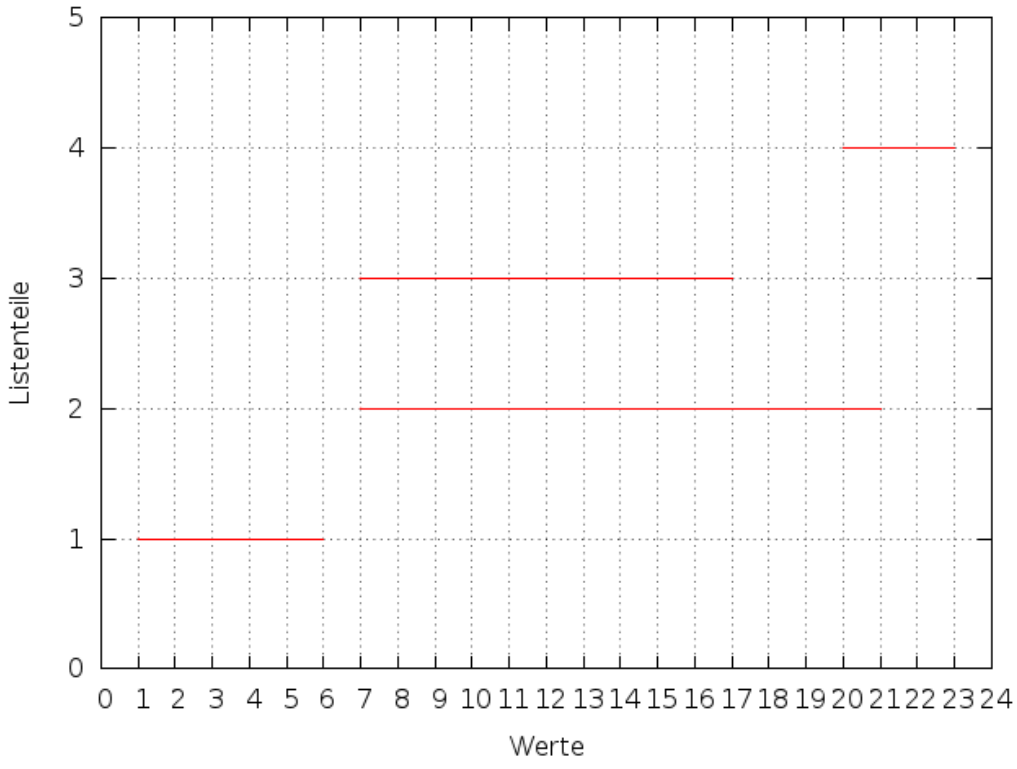


Diese Listen müssen jetzt nur noch in die richtige Reihenfolge gebracht werden und wir sind fertig.
Geht das aber für alle Listen so einfach?



Nein, aber müssen wir wirklich öfter „mergen“ als bei mergesort?

Darstellung der Listen mittels ihrer Wertebereiche.



Wenn wir uns dieses Beispiel ansehen, merken wir das eigentlich nur 3 Listen umverteilen müssten.
Was in diesem Fall die Listenumverteilungen auf 2 reduziert.

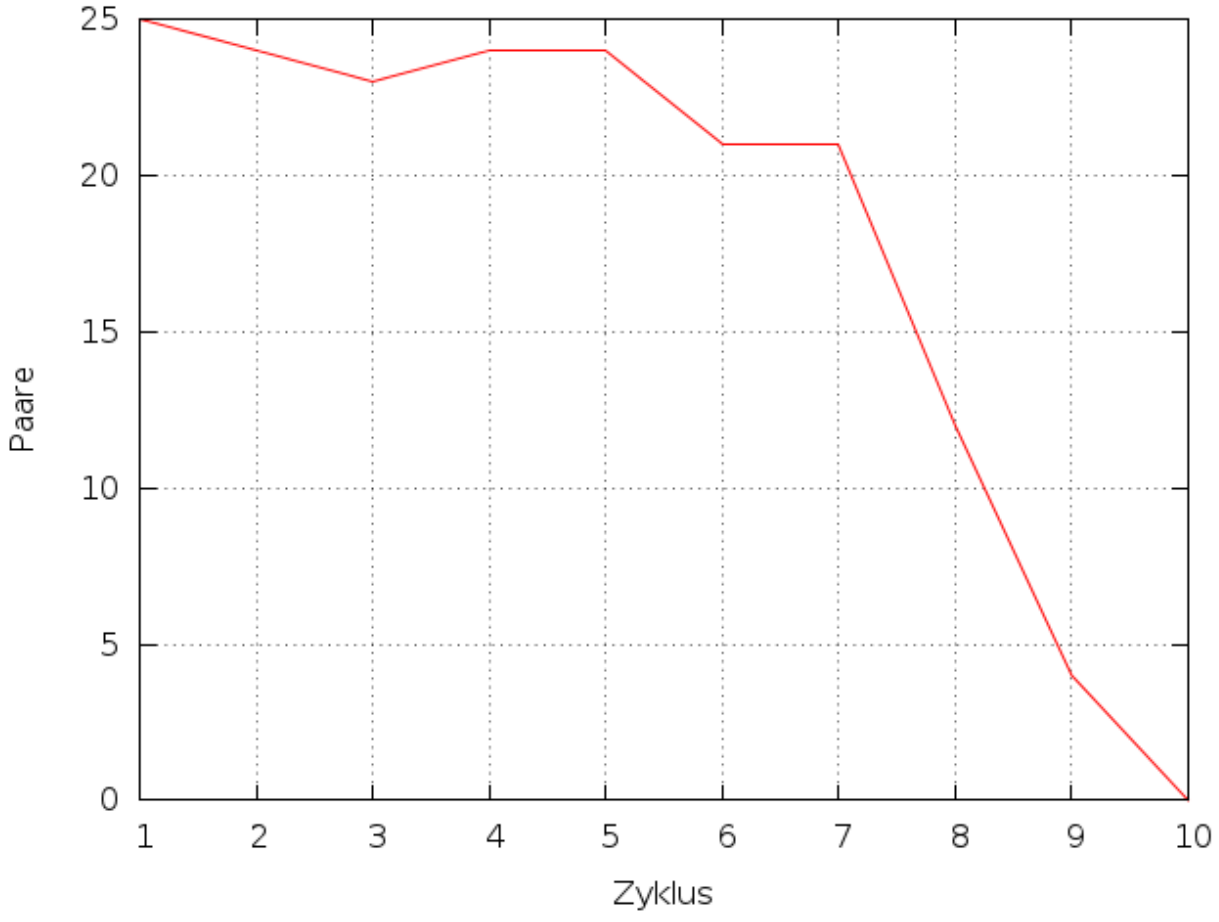
Wir brauchen eine Strategie welche Paare von Listen ermittelt bei denen eine Umverteilung am sinnvollsten ist.
Anforderungen:

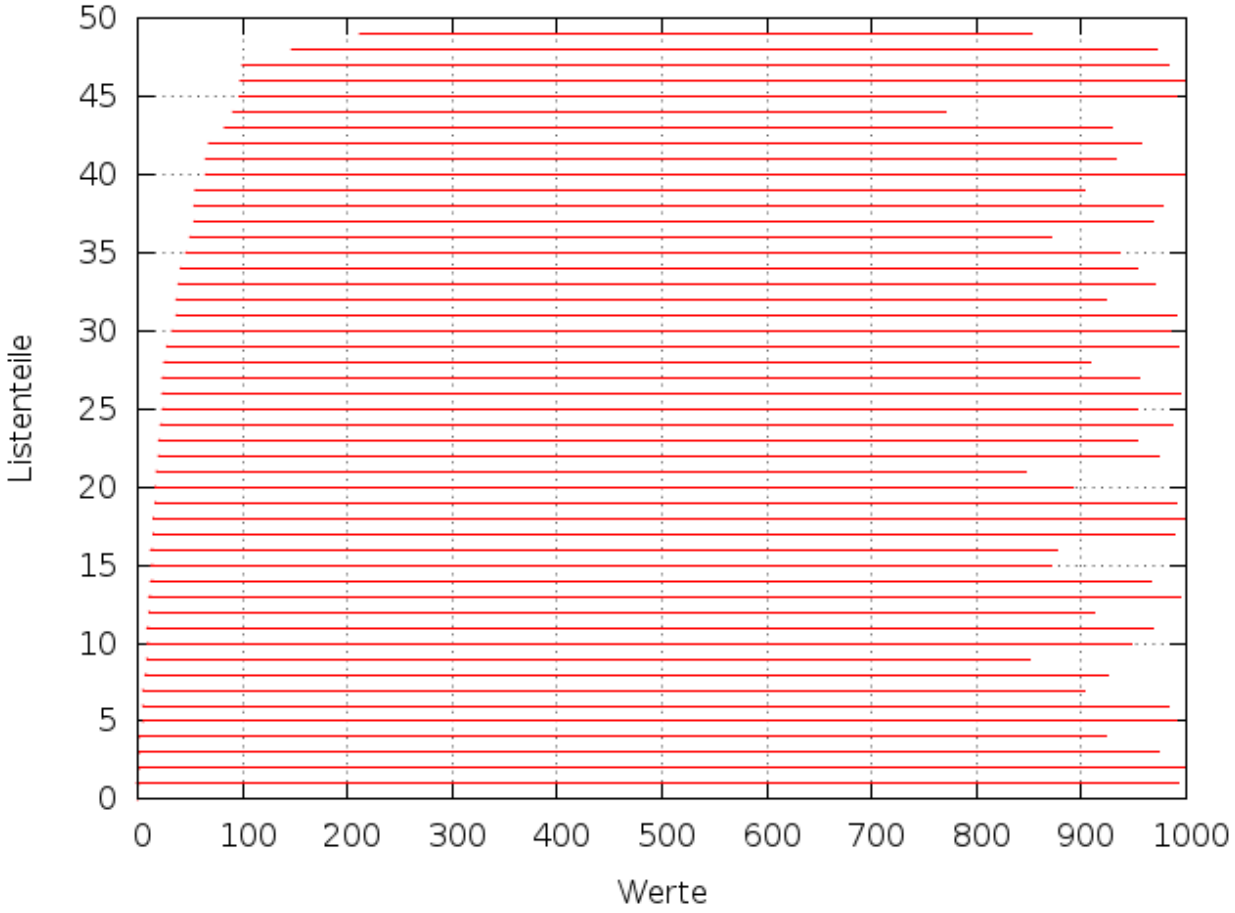
- möglichst schnell
- möglichst viele Paare finden damit wir viele Prozessoren auslasten können.

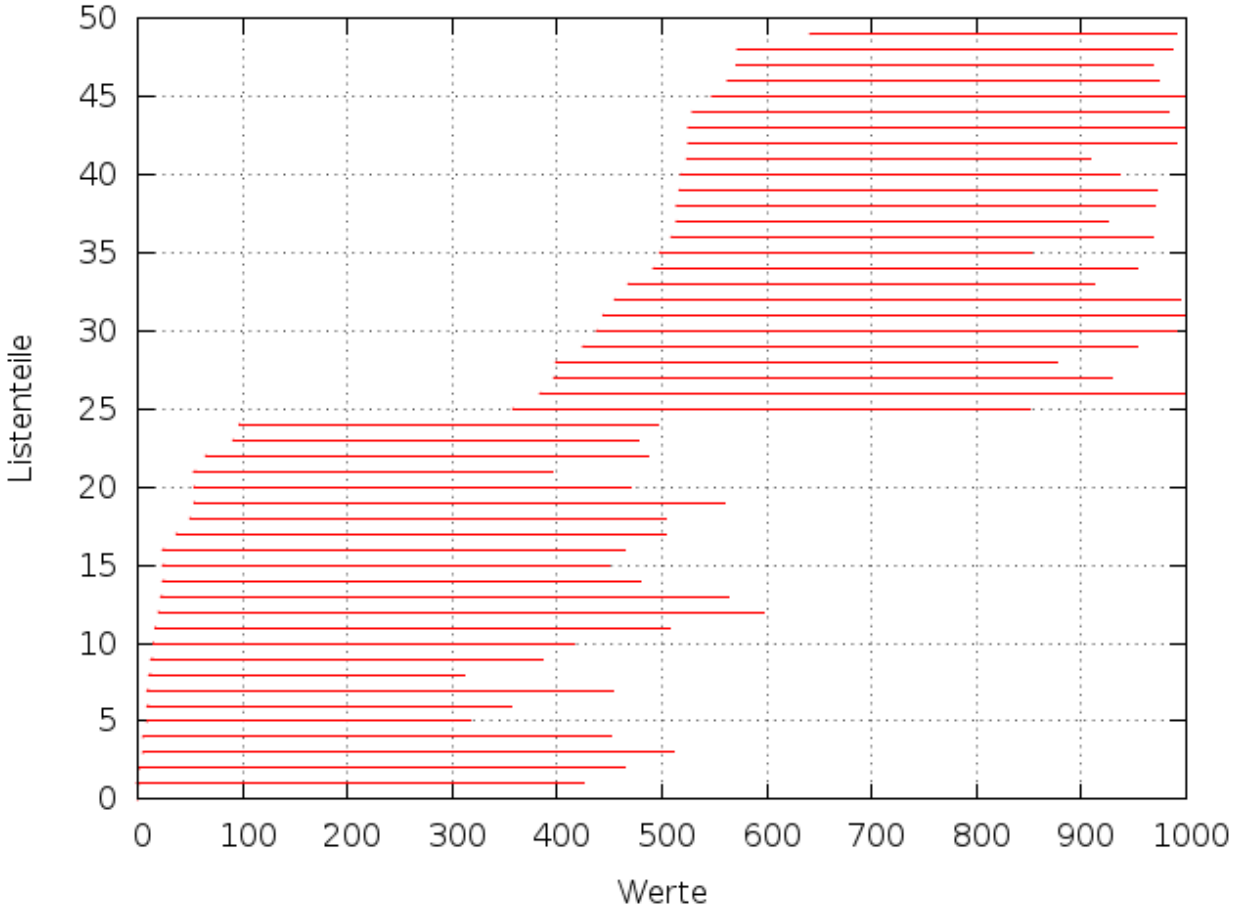
Strategie:
Das Ziel ist es Teillisten mit möglichst großen Überlappungen zu paaren.

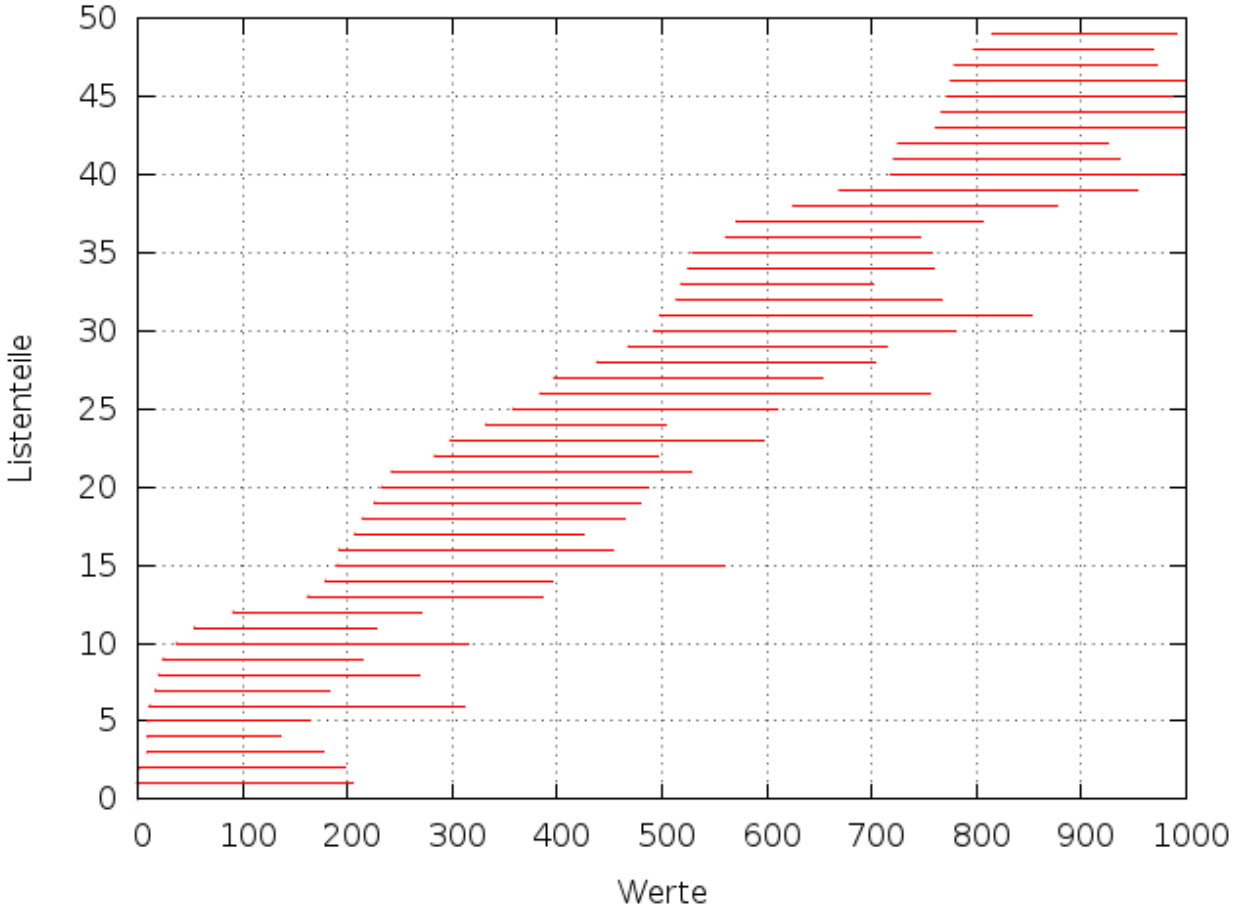
- nimm die Liste mit dem größten Wertebereich (Liste a)
- suche in den verfügbaren Listen nach einer die den Wertebereich von a maximal überlappt (Liste b)
- wenn gefunden: markiere a und b als nicht verfügbar und bilde ein Paar
- wenn nicht gefunden: markiere a als nicht verfügbar
- wiederhole diese Schritte bis keine Liste mehr verfügbar ist.

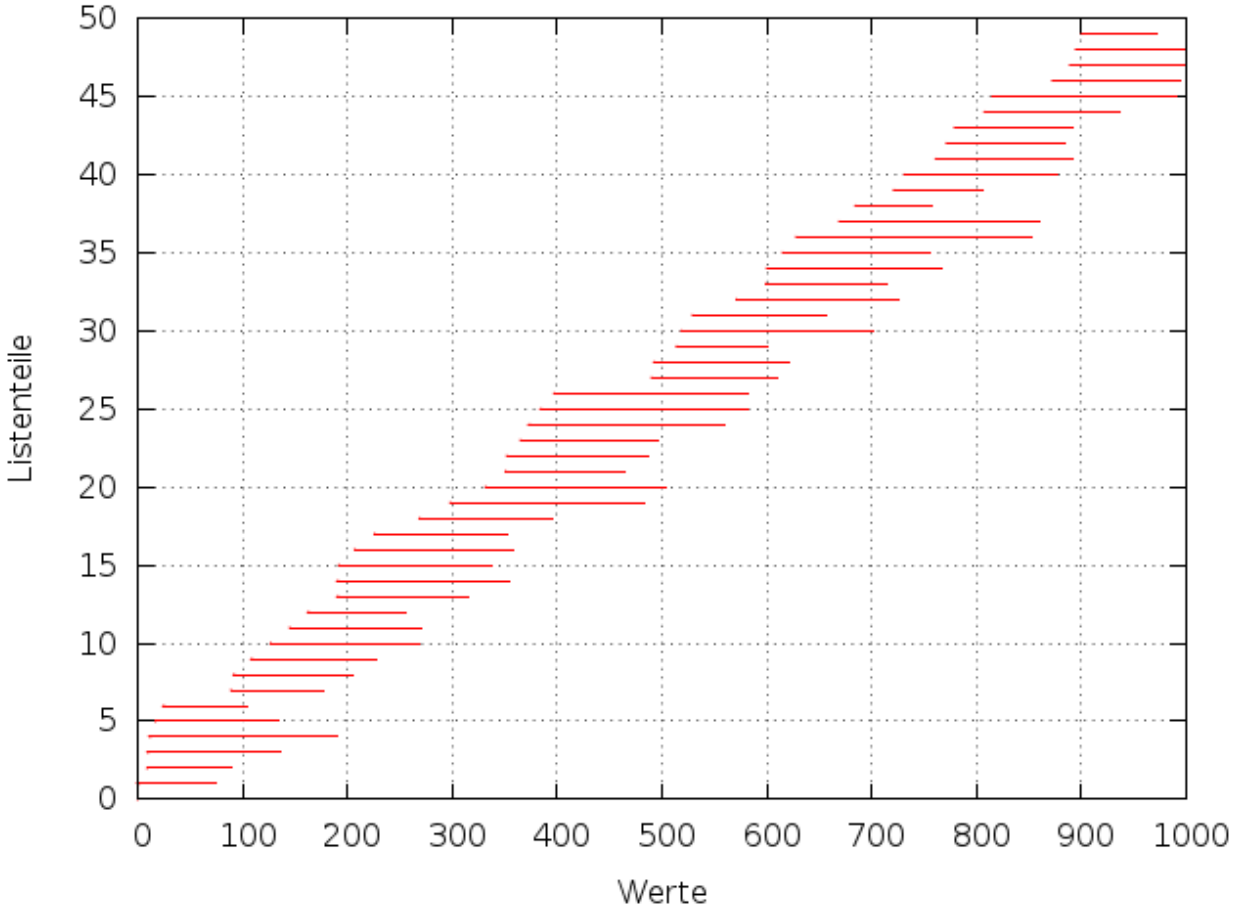
Diese Strategie auf eine Liste mit 1000 Elementen angewendet, welche in 50 Teile geteilt wurde.

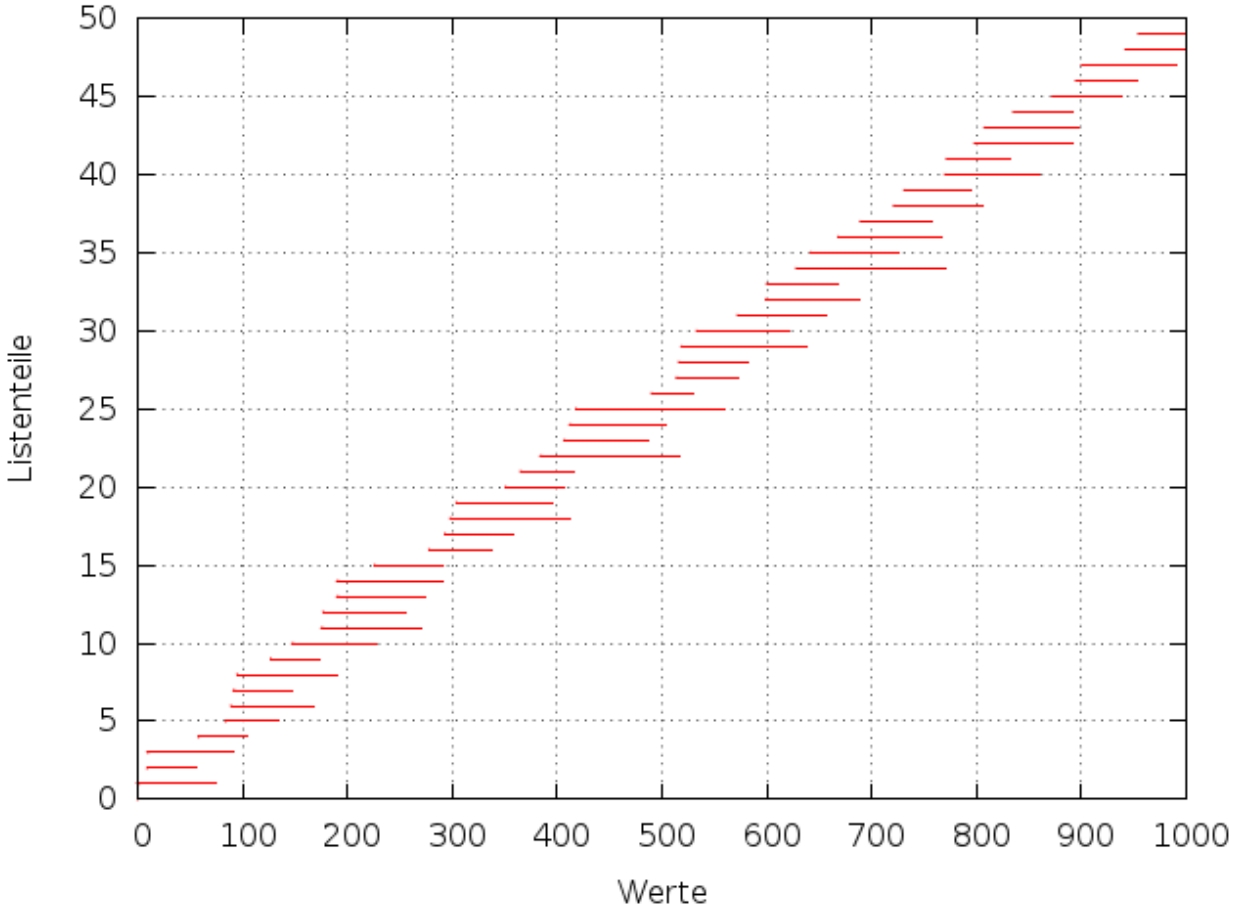


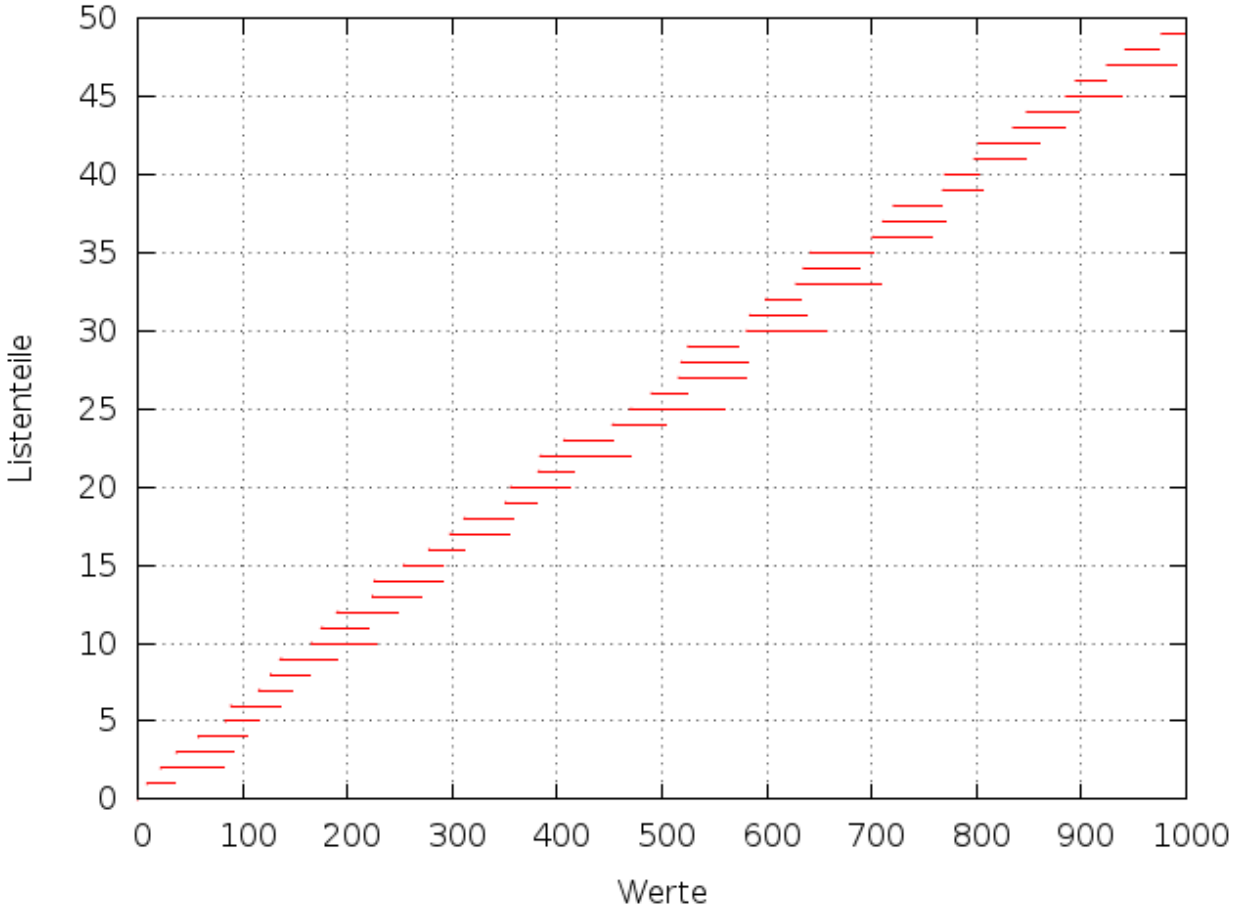

Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000

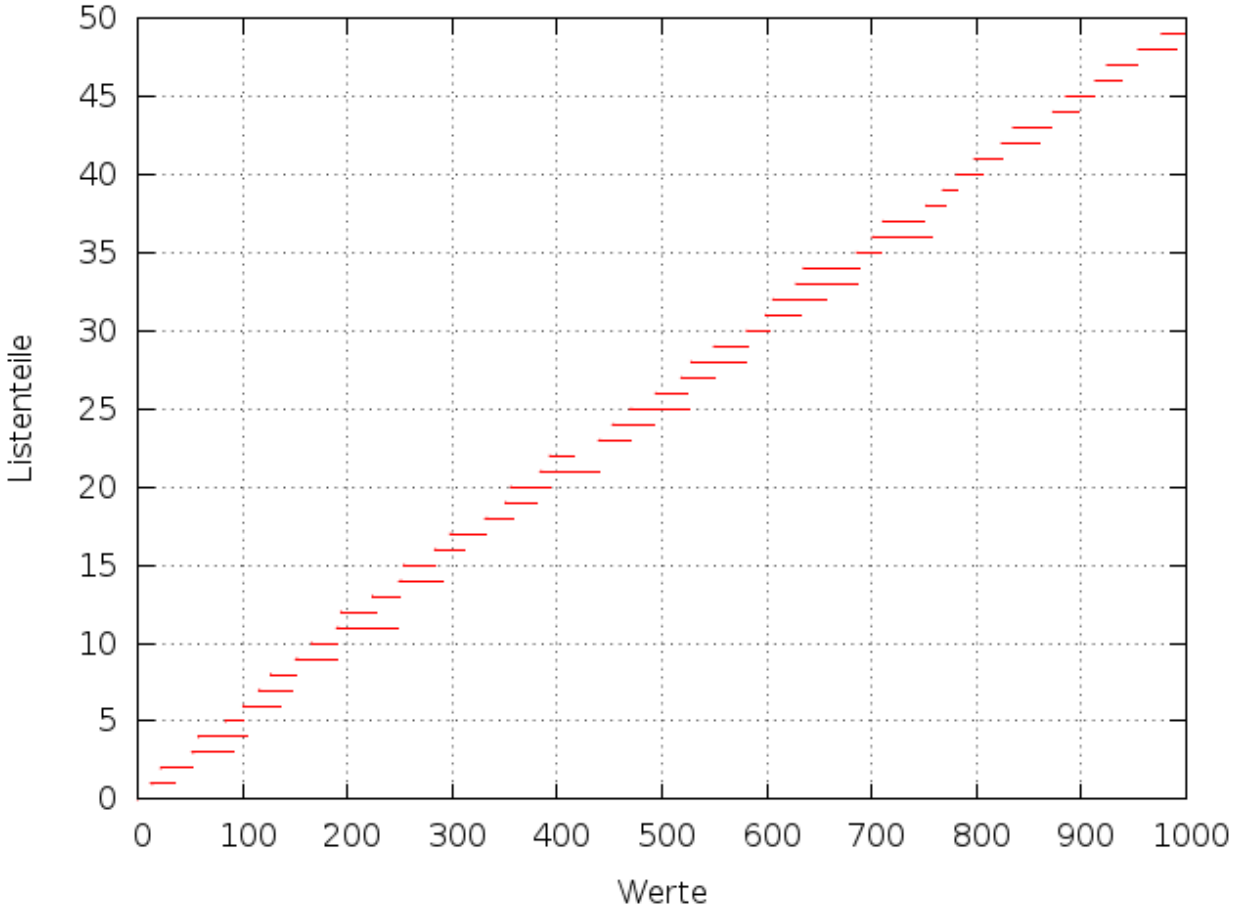

Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000

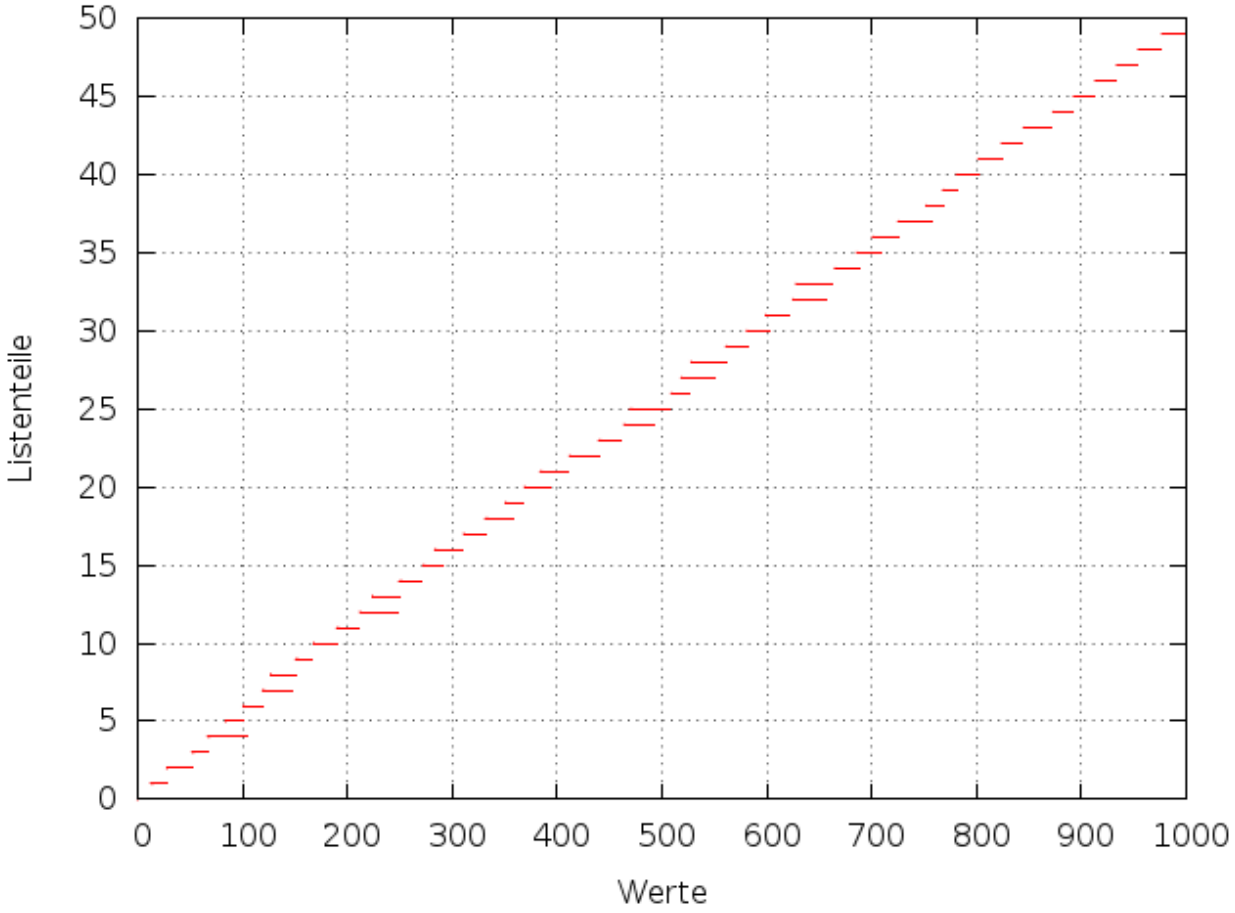

Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000

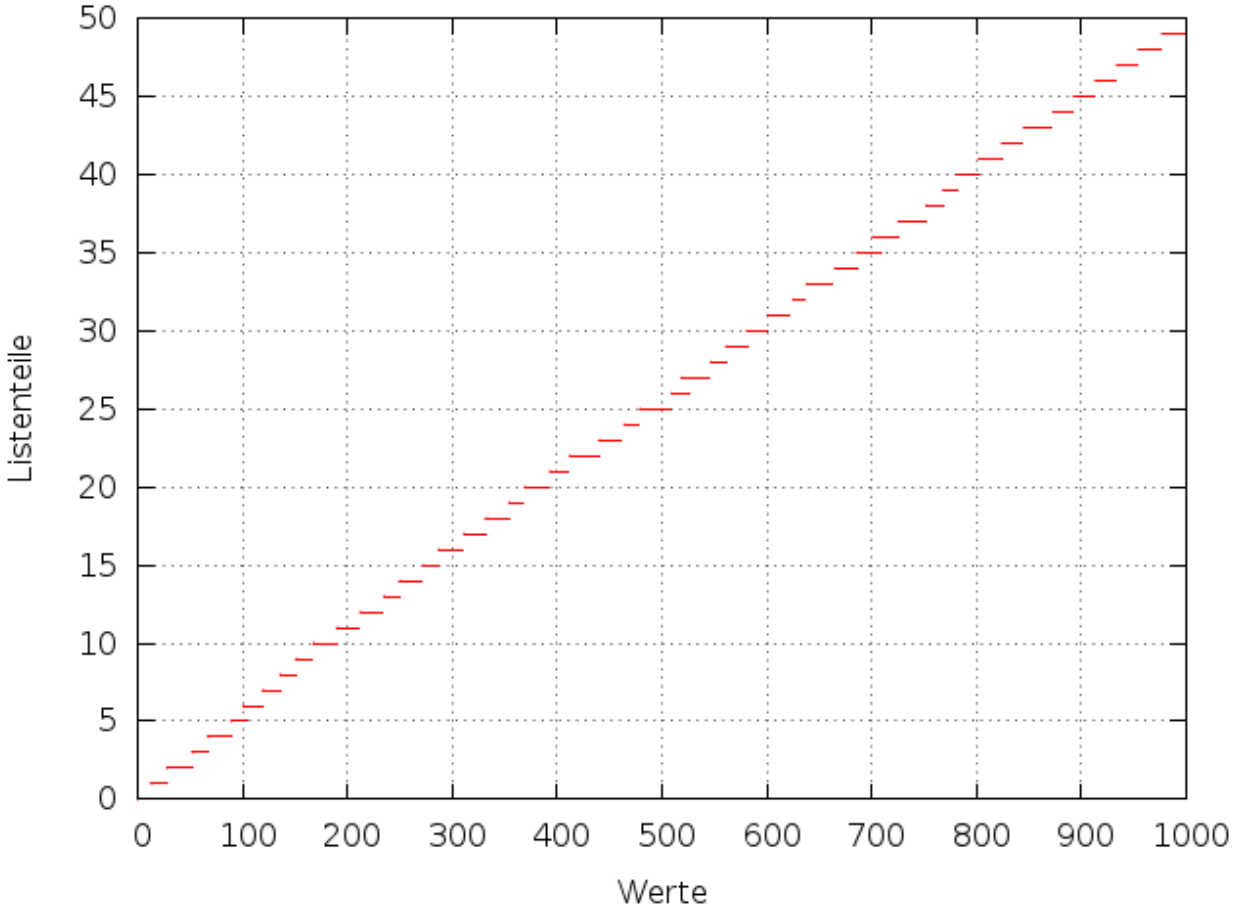

Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000

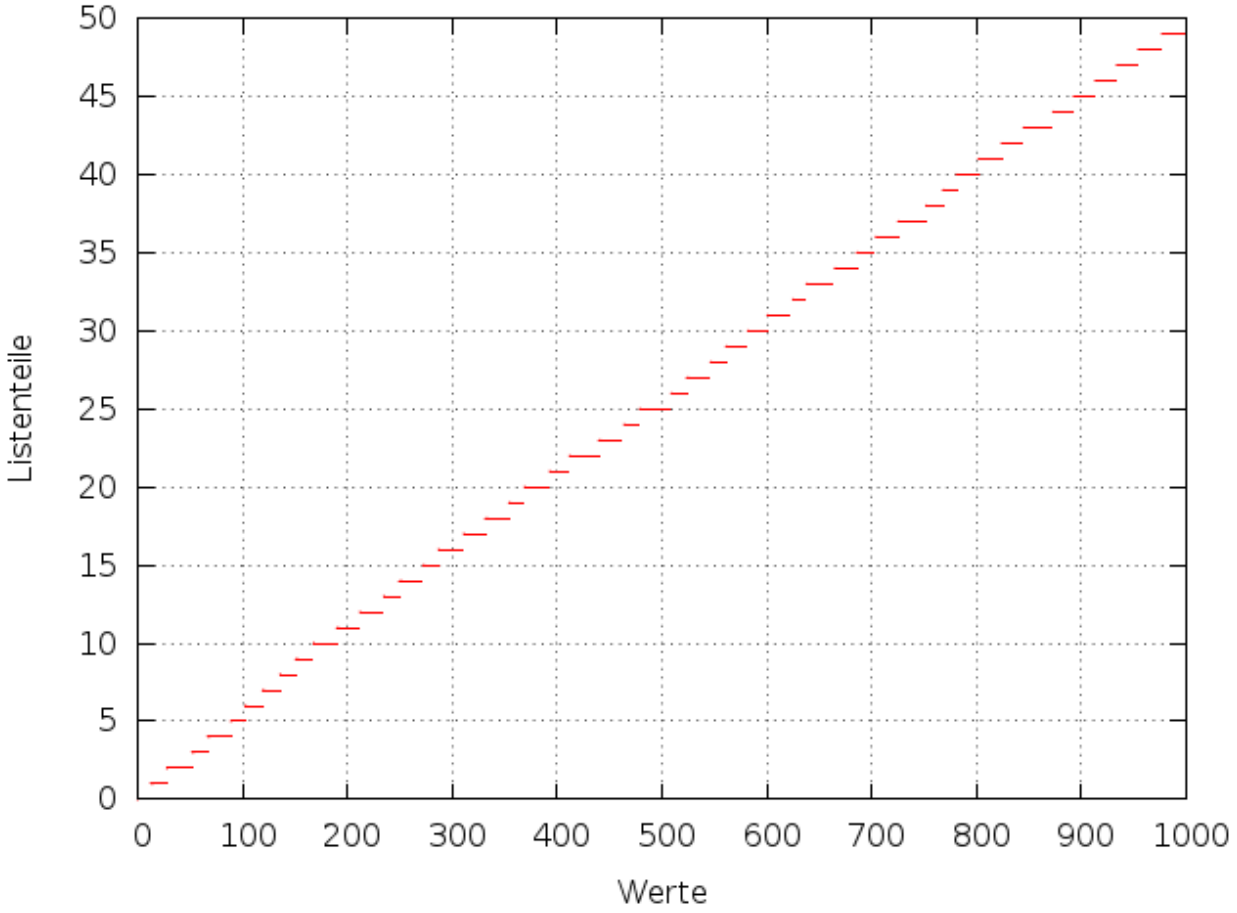

Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000


Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000

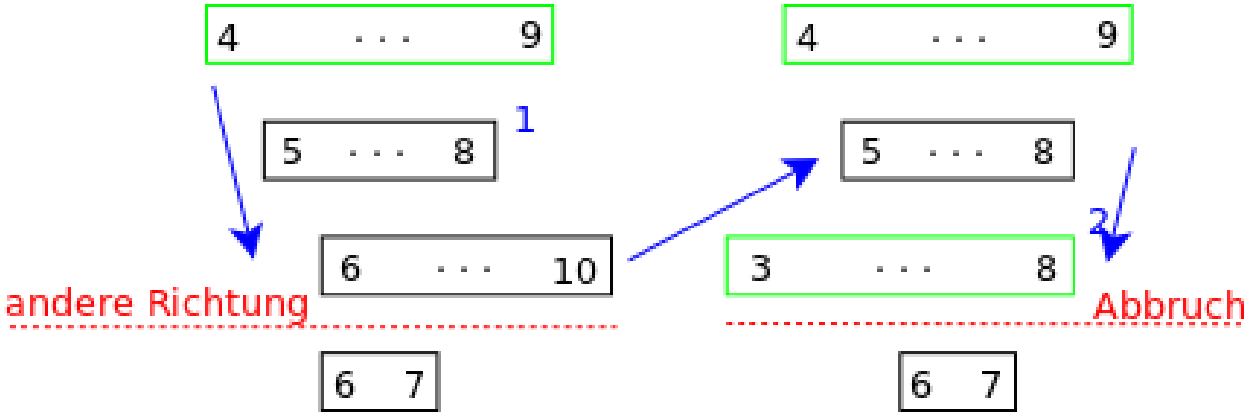

Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000


Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000


Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000


Listenteile
Werte
0
5
10
15
20
25
30
35
40
45
50
0
100
200
300
400
500
600
700
800
900
1000

suchen von Anfang/Ende der Liste.



Vorteile:

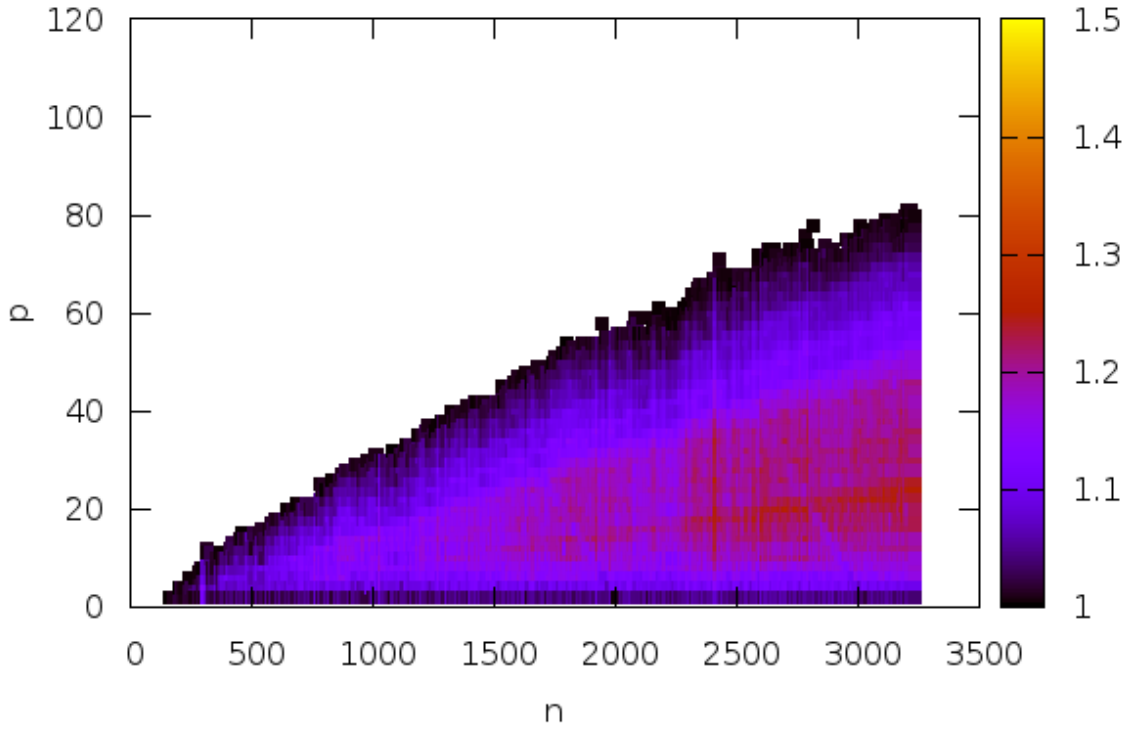
- man kann die Suche frühzeitig abbrechen.
- $T(p) = 3p \log p + p\, s(p)$

Problem

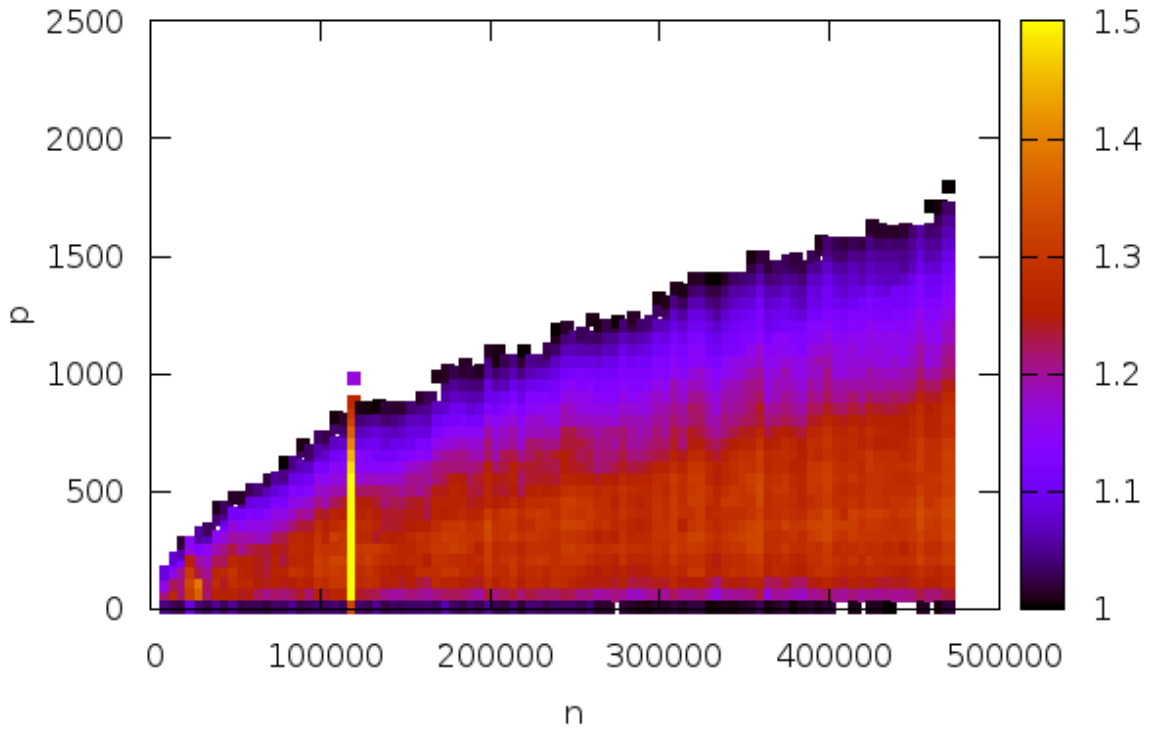
Funktionsweise

Auswertungen

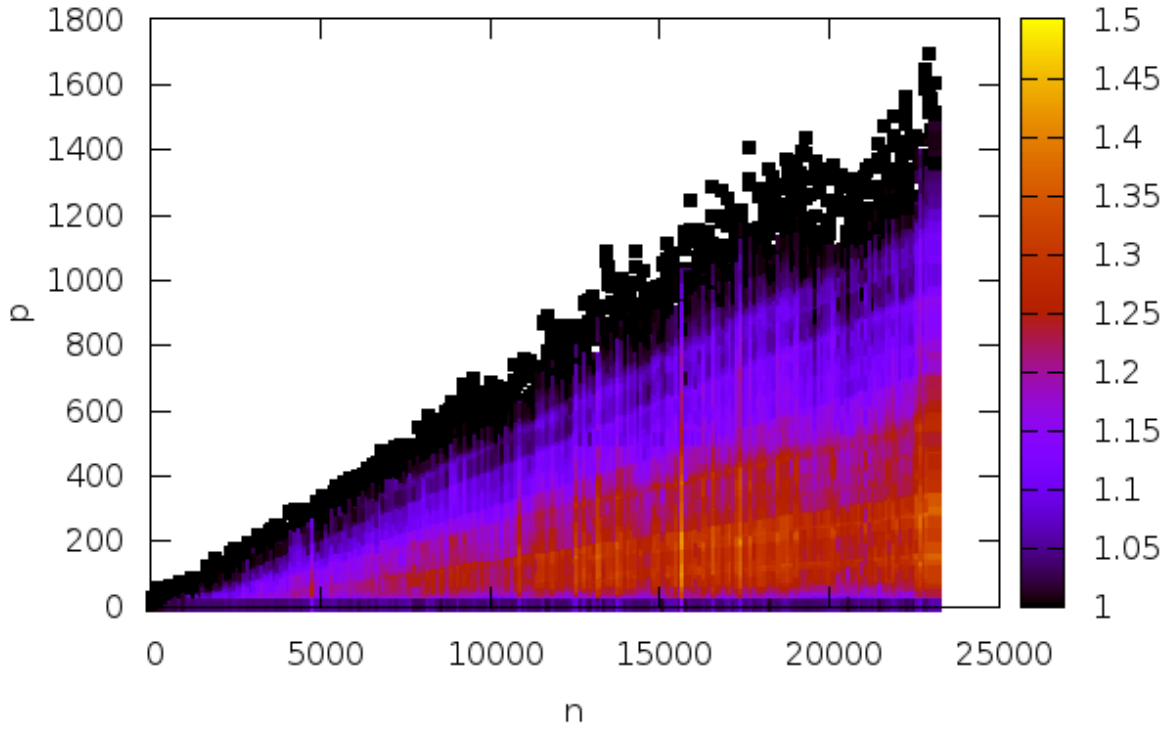
core2: T_C/T_D


120
100
80
60
40
20
0
p
0
500
1000
1500
2000
2500
3000
3500
n
1.5
1.4
1.3
1.2
1.1
1

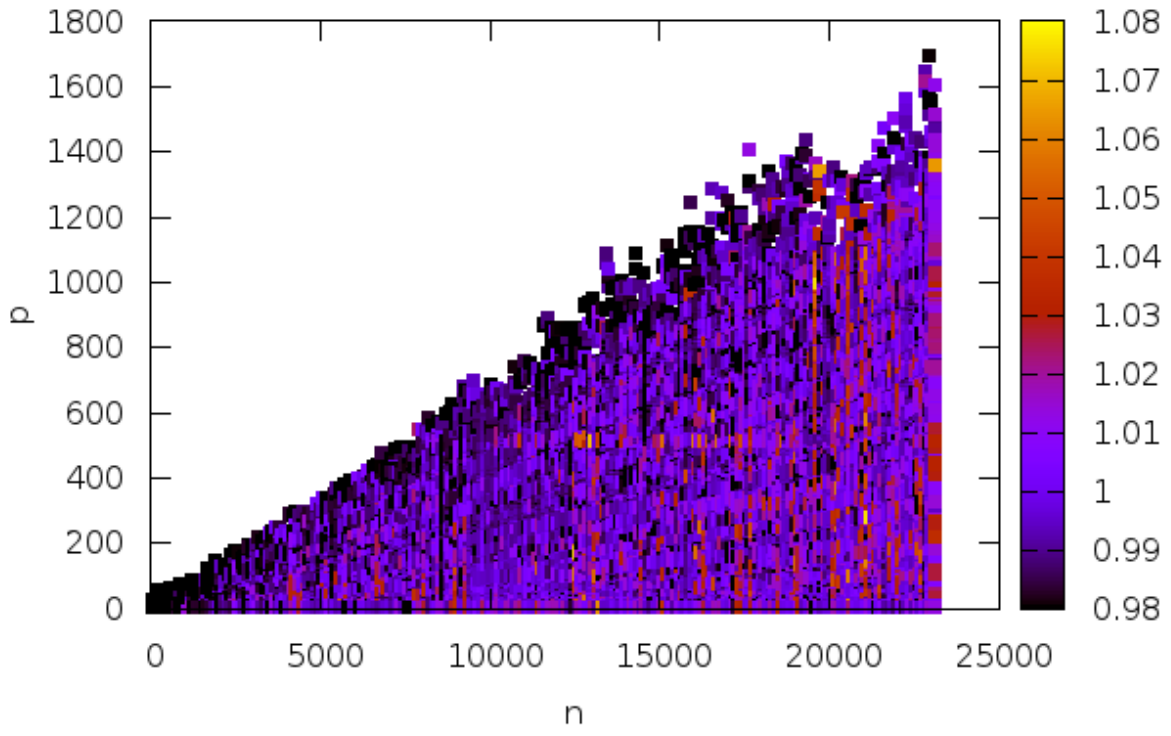
core2: T_C/T_D


2500
2000
1500
1000
500
0
p
0
100000
200000
300000
400000
500000
n
1.5
1.4
1.3
1.2
1.1
1

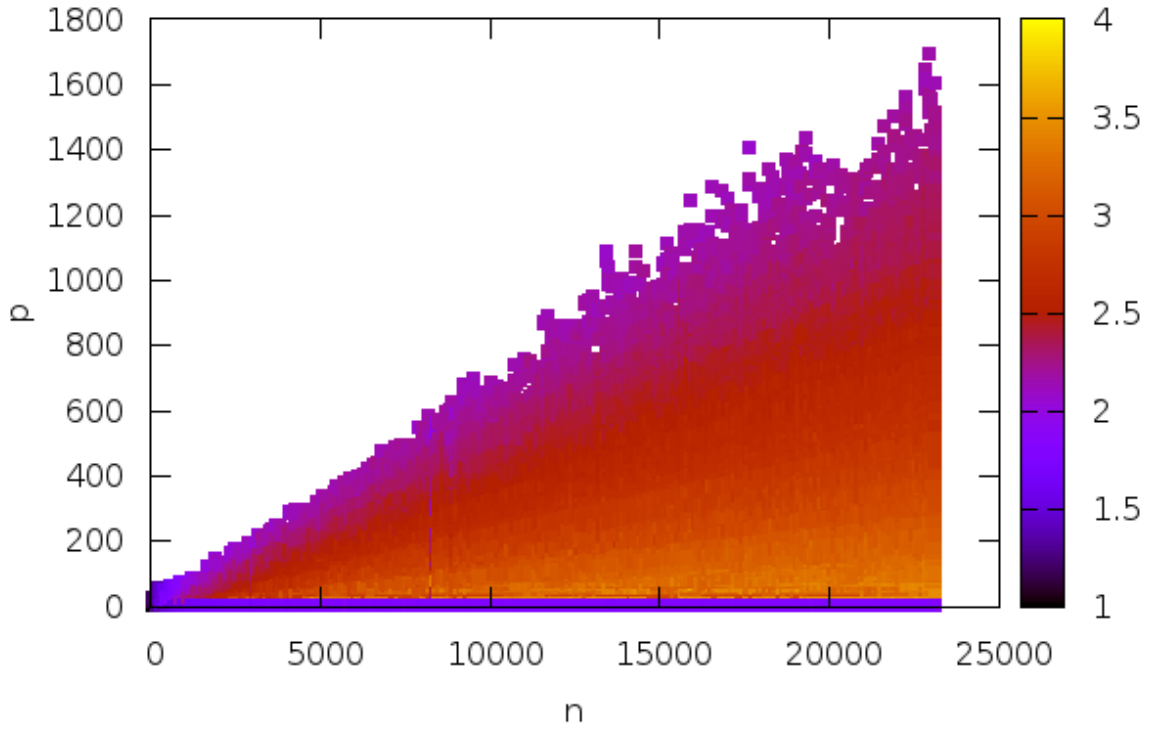
i7: T_C/T_D


1800
1600
1400
1200
1000
800
600
400
200
0
p
0
5000
10000
15000
20000
25000
n
1.5
1.45
1.4
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1

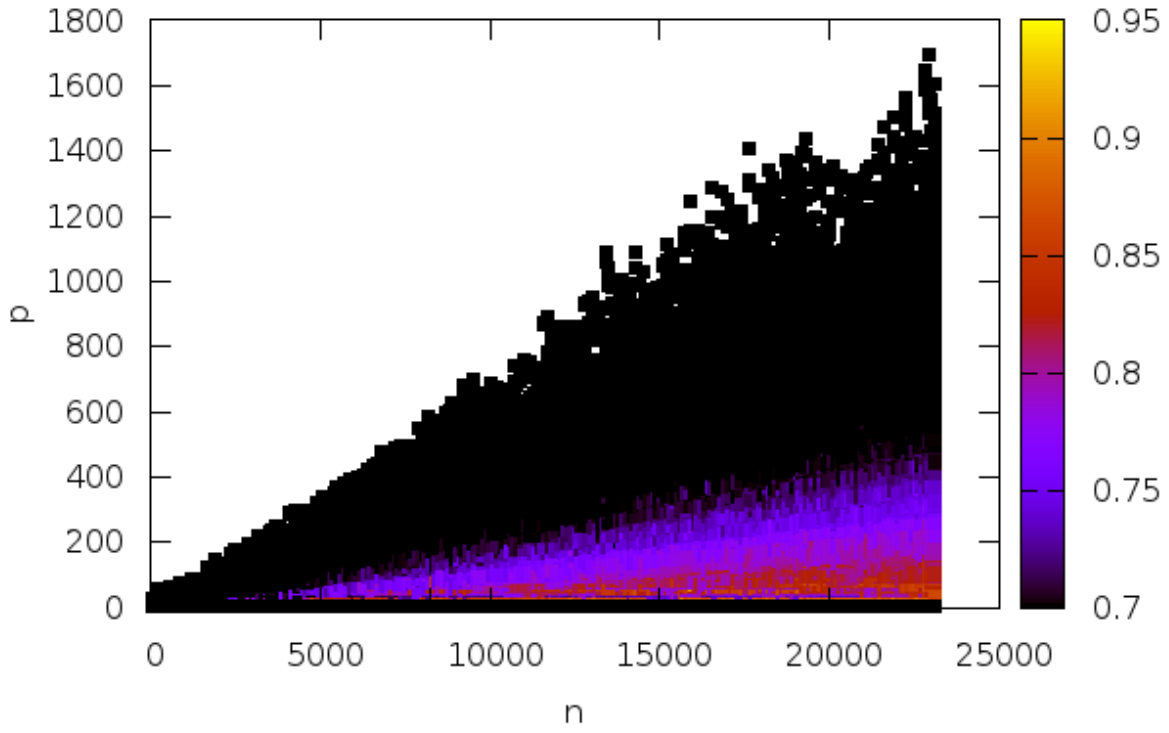
i7: T_D/T_{D1}


1800
1600
1400
1200
1000
800
600
400
200
0
p
0
5000
10000
15000
20000
25000
n
1.08
1.07
1.06
1.05
1.04
1.03
1.02
1.01
1
0.99
0.98

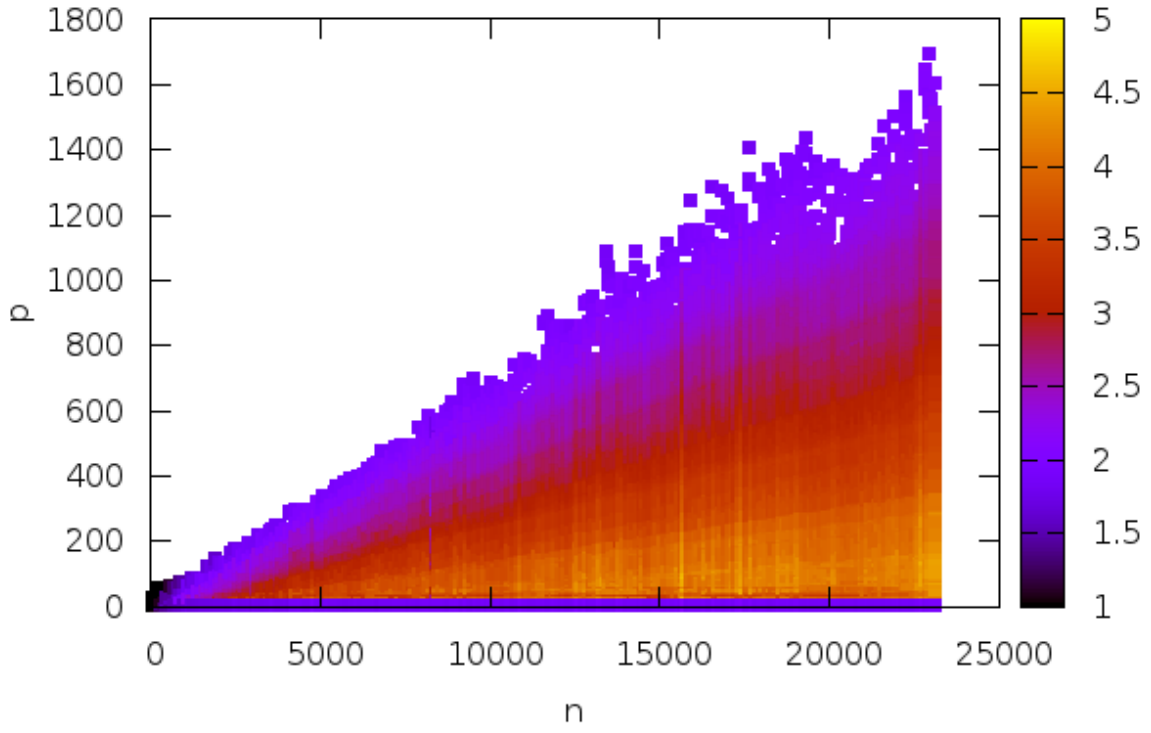
i7: T_{D1}/T_{D4}


1800
1600
1400
1200
1000
800
600
400
200
0
p
0
5000
10000
15000
20000
25000
n
4
3.5
3
2.5
2
1.5
1

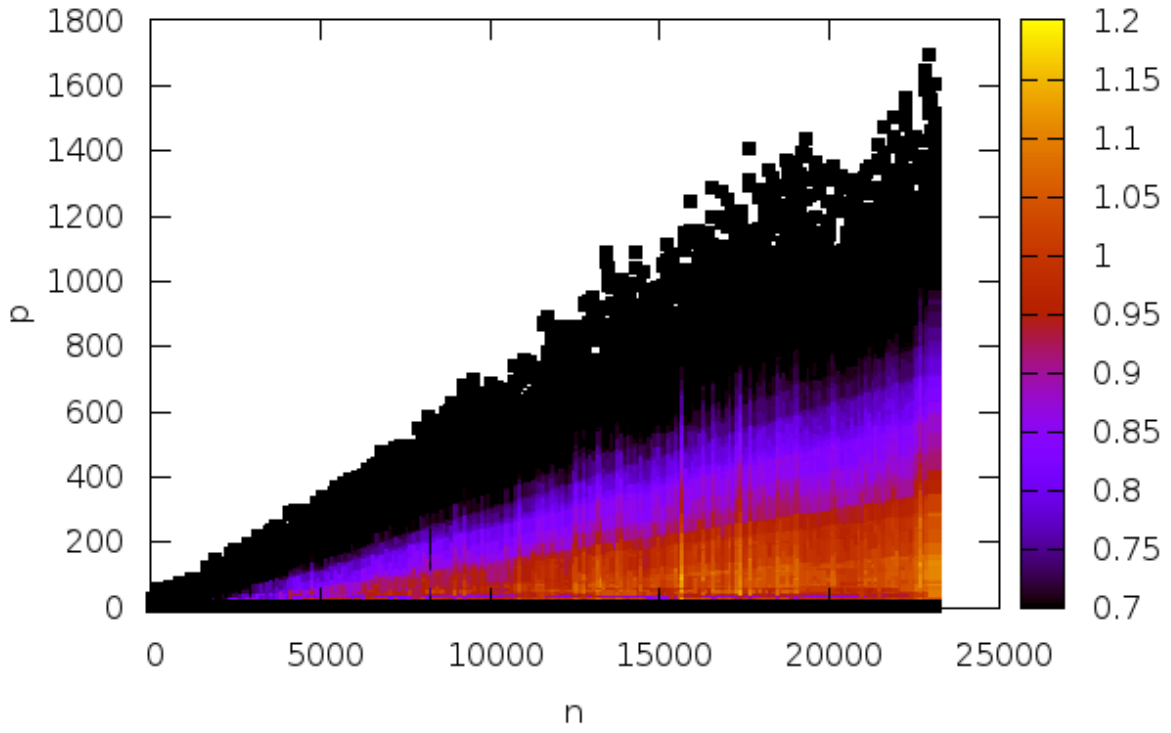
i7: $T_{D1}/(T_{D4}4)$


1800
1600
1400
1200
1000
800
600
400
200
0
p
0
5000
10000
15000
20000
25000
n
0.95
0.9
0.85
0.8
0.75
0.7

i7: T_C/T_{D4}


1800
1600
1400
1200
1000
800
600
400
200
0
p
0
5000
10000
15000
20000
25000
n
5
4.5
4
3.5
3
2.5
2
1.5
1

i7: $T_C/(T_{D4}4)$


1800
1600
1400
1200
1000
800
600
400
200
0
p
0
5000
10000
15000
20000
25000
n
1.2
1.15
1.1
1.05
1
0.95
0.9
0.85
0.8
0.75
0.7

$$T(n,p) = n\log(\tfrac{n}{p}) + an\log p + bp\log^2 p$$
$$0 < a < 1$$
$$b > 1$$

$\frac{n \log n}{T(n,p)}, a = 0.7, b = 5$