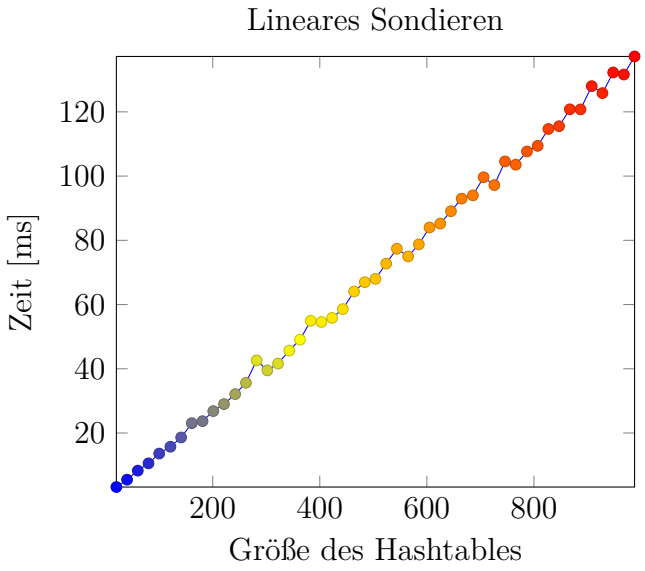
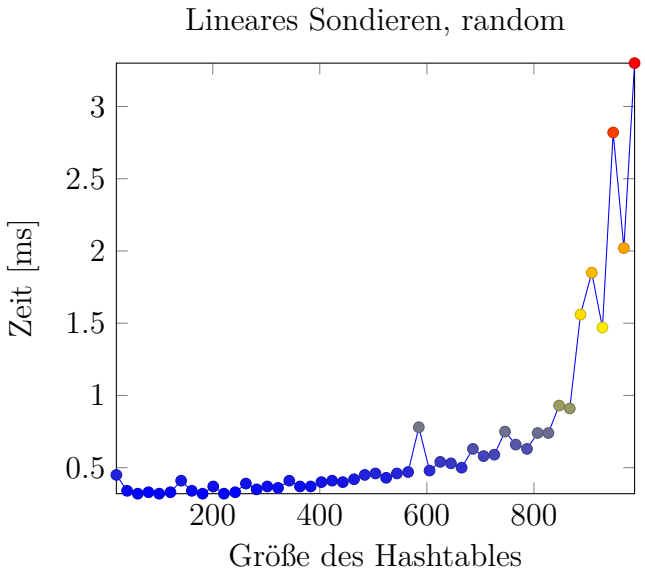
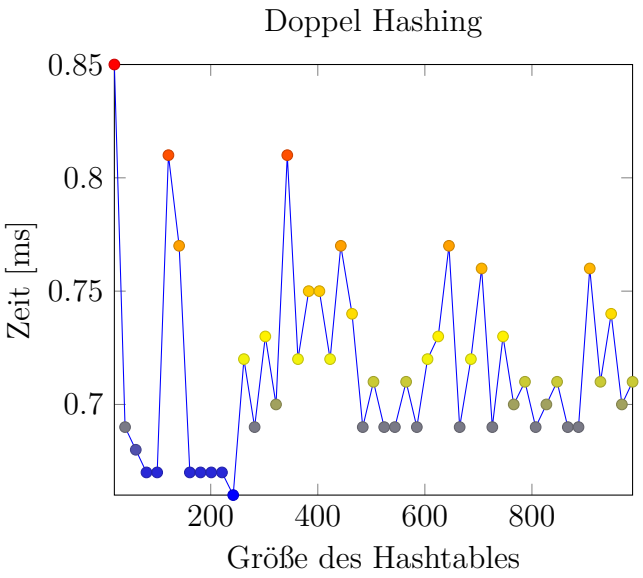
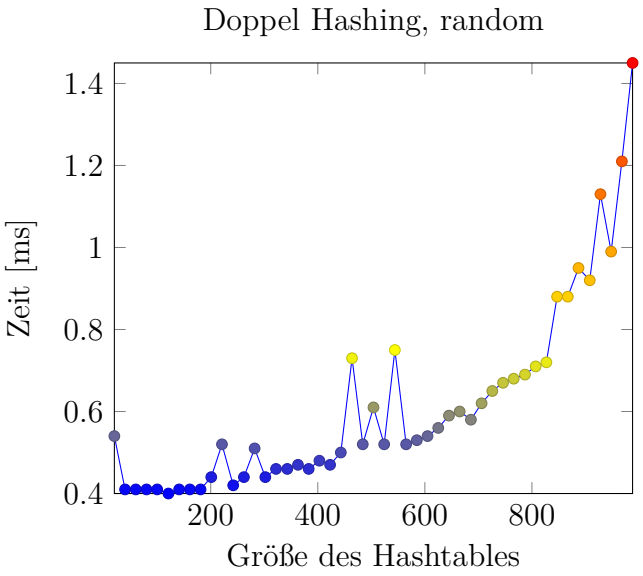
Plots


Lineares Sondieren
Zeit [ms]
120
100
80
60
40
20
200
400
600
800
Größe des Hashtables


Lineares Sondieren, random
Zeit [ms]
3
2.5
2
1.5
1
0.5
200
400
600
800
Größe des Hashtables


Doppel Hashing
0.85
0.8
0.75
0.7
Zeit [ms]
200
400
600
800
Größe des Hashtables


Doppel Hashing, random
1.4
1.2
1
0.8
0.6
0.4
Zeit [ms]
200
400
600
800
Größe des Hashtables

Aufgabe 2

a)

Die Erste for Schleife geht von $i = 1$ bis $n - 1$ durch das Array A.
Die Zweite for Schleife geht von $j = 0$ bis $i - 1$ ebenfalls durch das Array A.
Die Dritte for Schleife geht von $k = 0$ bis $n - 1$ ebenfalls durch das Array A.
Dabei wird jeweils die Absolute Differenz von dem Element A[i] und dem Element A[j] mit dem Element A[z] verglichen und falls die gleich sind wird ein True zurückgegeben, und falls alle Schleifen durchlaufen und die If-Bedingung kein einziges mal zutrifft wird ein False zurückgegeben.

Die asymptotische Laufzeit beträgt: $\mathcal{O}((n-1)(n-1)n) = \mathcal{O}(n^3)$

b)

```
def algorithm(A):
    for i in range(len(A)):                    # n
        for j in range(i):                     # n
            hashtable.append(abs(A[i]-A[j]))   # 1

    for i in A:                                # n
        if A in hashtable:                     # 1
            return True

    return False
```

Wie man an dem Pseudocode sieht, können wir mit Hashtables die Hintereinanderreihung von for Schleifen auf 2 statt 3 begrenzen, dadurch haben wir ein asymptotische Laufzeit von: $\mathcal{O}(n * n + n) = \mathcal{O}(n^2)$

c)

```python
def algorithm2(A):
    sorted(A)

    for i in range(len(A)): # n
        for j in range(i): # n
            if binarysearch(A, abs(A[i]-A[j])): # log n
                return True
    return False

def binarysearch(A, k):
    # search for k in A
    # if k in A return True, otherwise False
```

Wie man an dem Pseudocode sieht, sortieren wir zuerst das Array A.
Wie auch bei b beschränken wir uns hier auf zwei for Schleifen, das ist möglich weil wir eine sortierte Liste und den Binary Search verwenden, welches nur eine Laufzeit von $\mathcal{O}(\log n)$ hat.
Die asymptotische Laufzeit insgesamt beträgt also: $\mathcal{O}(n * n * \log n) = \mathcal{O}(n^2 \log n)$