Aufgabe 59:

Für Werte t = 1, 3, 5, 13 sind die Funktion f(t) eine Permutation der Menge {0, 1, ... , 1023}. Denn in unserem Programm ist “value” von “key” gleich 1, was gemeint ist, dass das Maximum von der List temp gleich 1 ist. Und diese Situation kommt nur vor, wenn es in jeder Zeller von der Liste genau ein Element gibt. Also alle 1024 unterschiedlichen Elemente werden dann in 1024 Zellen gespeichert.

Aufgabe 61:

a)**zuklein():** bleibt unverändert, weil zuklein() im schlimmsten Fall logd(n) aufgerufen wird. Also vom Blatt zur Wurzel. Mit Änderung von d ändert sich nur die Höhe des Baums, wenn es n Knoten hat. Aber die Menge der Komplexität bleibt O(logd(n))
einfügen(), verkleinereSchlüssel() haben auch gleiche Laufzeit, da in den beiden Funktion zuklein() einmal aufgerufen wird.

Laufzeit: O(logd(n))

zugroß(): in dieser Funktion müssen wir d-Mal vergleichen, damit wir das kleinste Kind finden. Und um die Eigenschaft von Halde nicht zu verletzen, müssen wir im schlimmsten Fall log(n)-Mal diese Funktion aufrufen.
entfernemin() hat gleiche Laufzeit wie zugroß(), weil in ihr zugroß() einmal aufgerufen wird und alles anderes konstante Laufzeit hat.

Laufzeit: O(d*logd(n))

b)
Im Algorithmus Dijkstra wird

1. entfernemin() n-mal. Die Laufzeit von entfernemin() ist gleich O(d*logd(n))
2. einfügen() n -mal. Die Laufzeit von einfügen() ist gleich O(logd(n))
3. verkleinereSchlüssel() m-mal. Die Laufzeit von verkleinereSchlüssel() ist gleich O(logd(n))

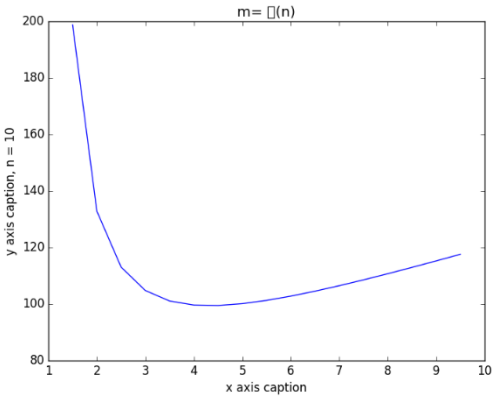
aufgerufen.

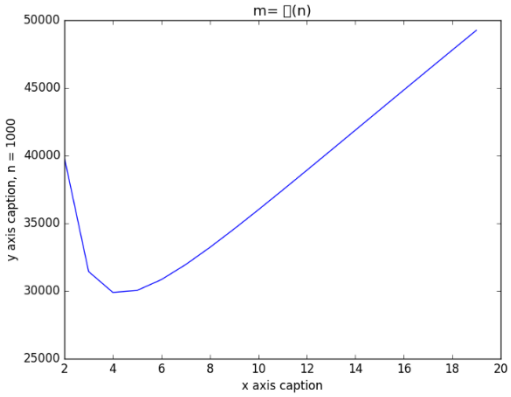
Dann entspricht die gesamte Laufzeit von Dijkstra： **O(n*d*logd(n)+(m+n)logd(n))**

Bestimmung der besten asymptotischen Laufzeit:

1. Als eine naive Methode können wir einfach das Polynom von der Laufzeit einmal ableiten, und nach der Extremstelle suchen, damit wir das Minimum finden können. Aber leider ist es zu schwer, die Nullstelle von der ersten Ableitung zu lösen.
2. Wir können uns die Funktion für verschiedene n und m ausplotten lassen. Besonders betrachten wir die Extremfälle m = Θ(n) und m = Θ(n^2)

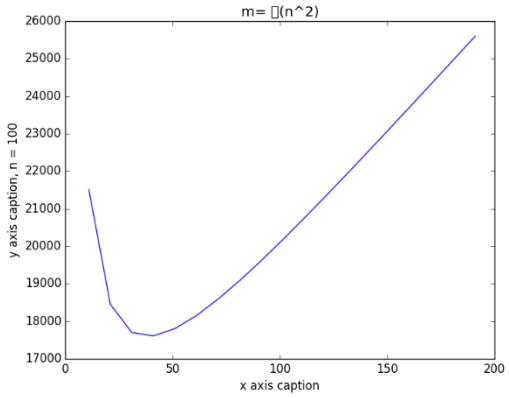
m = Θ(n):

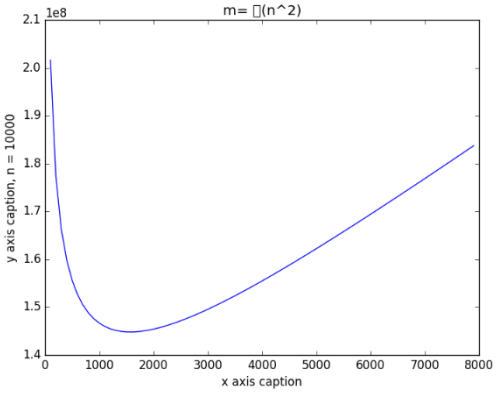




Wir können leicht beobachten von den beiden Bildern, dass die Größe n in diesem Fall nicht zum Ergebnis relevant. Das Minimum liegt immer zwischen 4 und 5. Dann soll man d = 4 wählen, wenn man weiß, dass der zu untersuchende Graph wenig Kante enthält.

m = Θ(n^2):





Von den beiden Bildern können wir leicht darüber informieren, dass der optimale Grad d der Halde mit aufsteigender Knotenanzahl n und dazu exponentiell steigender Kantenzahl m immer größer wird. Also, wenn der Graph vollständiger ist, sollte Grad d der Halde größer gewählt werden.

Die Laufzeit in den Extremfällen:

1. d = n → eine unsortierte Liste

dijkstra(): O(n*n*logn(n)+(m+n)logn(n)) = O(n^2 + m + n)

m = Θ(n) → O(n^2 + 2*n)

m = Θ(n^2) → O(2n^2 + n)

2. d = 1 → eine sortierte Liste

dijkstra(): O(n^2 + n + n*m)

m = Θ(n) → O(2n^2 + n)

m = Θ(n^2) → O(n^3 + n^2 + n)