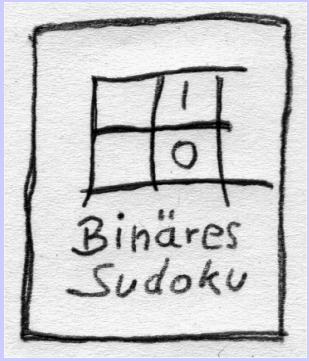
hochschule mannheim

Jörn Fischer
j.fischer@hs-mannheim.de

Willkommen zur Vorlesung

Informatik 2

Algorithmen und Datenstrukturen



Überblick

Inhalt

1 Eigenschaften von Algorithmen

– Algorithmenbegriff

– O-Notation

– Entwurfstechniken

2 Datenstrukturen

– Arrays

– verkettete Listen

– Bäume

– Stacks

– Queues

– Graphen

3 Sortierverfahren

– Bubblesort

– Mergesort

4 Suchen

– Lineare Suche

– Iterative Suche

5 Hashverfahren

– Hashfunktion

– Sondieren

1 - Eigenschaften von Algorithmen

1 – Eigenschaften von Algorithmen

Kapitel 1 – Eigenschaften von Algorithmen

- Algorithmenbegriff
- O-Notation
- Entwurfstechniken

1 – Eigenschaften von Algorithmen

Algorithmenbegriff (Definition):

- Ein **Algorithmus** ist ein Verfahren mit einer
  - präzisen (d.h. in einer genau festgelegten Sprache formulierten)
  - endlichen Beschreibung unter Verwendung
    - effektiver (d.h. tatsächlich ausführbarer)
    - elementarer Verarbeitungsschritte
- Ein Algorithmus benötigt nur endlich viele Ressourcen:
  - Rechenzeit
  - Speicher

Ein Algorithmus ist unabhängig von der Programmiersprache.

1 – Eigenschaften von Algorithmen

Algorithmenbegriff (Existenz):

- Wenn man sich ausschließlich mit der positiven Aussage über die Existenz eines Algorithmus zufrieden gibt, so reicht es Beispielimplementationen vorzulegen.
- Möchte man zeigen, dass "es keinen Algorithmus gibt, der Problem XY löst", so muss man sich tiefer in den formalisierten Algorithmenbegriff einarbeiten.
- Dies ist aber nicht im Fokus dieser Vorlesung und wird hier nicht näher behandelt.

1 – Eigenschaften von Algorithmen

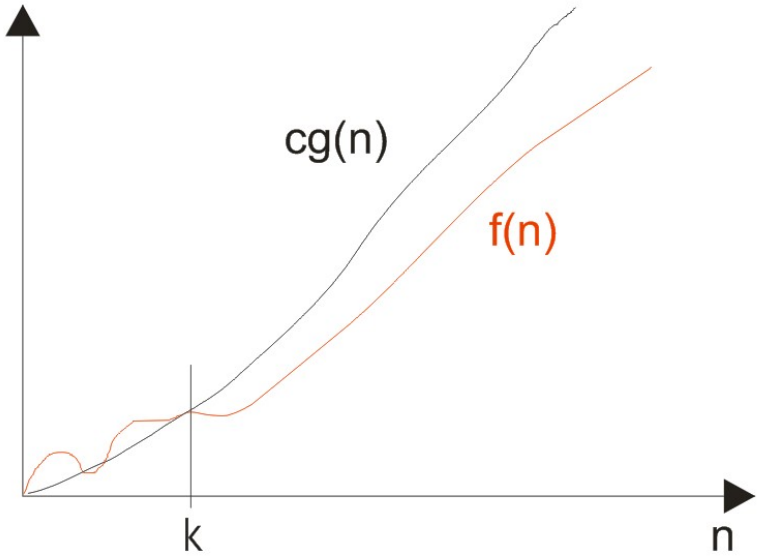
Algorithmenbegriff (Korrektheit):

- Um zu zeigen, dass ein Algorithmus korrekt ist muss bewiesen werden, dass er die gegebene Aufgabe in allen möglichen Fällen richtig löst.
- Das kann in den meisten Fällen nicht auf Basis von Beispielen geschehen, denn "man kann durch Testen zwar die Anwesenheit, nicht aber die Abwesenheit von Fehlern, also die Korrektheit eines Programmes, zeigen" (Zitat: E. Dijkstra).

1 – Eigenschaften von Algorithmen

O-Notation

- Die O-Notation ist ein Mass für die Zeitkomplexität eines Algorithmus
- f(n)=O(g(n)) bedeutet, dass es Konstanten k und c gibt, so dass



- f(n) wächst mit ansteigendem n
- maximal so schnell wie cg(n)
- (worst case)

1 – Eigenschaften von Algorithmen

Entwurfstechniken

- Um einen Algorithmus zu entwerfen kann man kaum strenge Regeln anwenden, trotzdem gibt es immer wieder verwendete Strategie, die helfen, einen Algorithmus zu finden, der das gegebene Problem löst
- Im zweiten Schritt kann man dann den Algorithmus in seinen Einzelteilen optimieren. Ist der Algorithmus effizient, d.h.hat er eine Zeit- und Speicherkomplexität, die gut skaliert, so steht einer Umsetzung auf heutige Computer nichts mehr im Wege.

1 – Eigenschaften von Algorithmen

Entwurfstechniken (Greedy Strategie)

- Bei manchen Problemen ist es möglich den kompletten Lösungsraum zu durchsuchen und so die Lösung zu finden:

  Beispiele:

  - Die kürzeste Rundreise bei 5 Städten

  - Die Teiler einer 5 stelligen Zahl

  - Das größte Element aus einer Liste von 1000000 Elementen

1 – Eigenschaften von Algorithmen

Entwurfstechniken (Divide And Conquer)

- „Divide And Conquer“ wird oft fälschlicherweise übersetzt mit „teile und herrsche“

  heißt aber eigentlich „teile und erobere“ oder „teile und überwinde“
- Viele Probleme lassen sich so in Teilprobleme zerlegen, die mit weniger Aufwand zu lösen sind
- Beispiele:

  - Iterative Suche in sortierte Zahlen (wie Suche im Telefonbuch oder Lexikon)

  - Sortieren mittels Quicksort oder Mergesort

1 – Eigenschaften von Algorithmen

Entwurfstechniken (Optimierungstechniken)

- Für NP-schwere Algorithmen, die mit bekannten Algorithmen nur in nicht polinomineller Zeit gelöst werden können gibt es Optimierungsstrategien
- Dazu gehören z.B. Evolutionäre Strategien wie Genetische Algorithmen, Neuronale Netze, Reinforcement Learning etc.
- Oftmals muss man sich mit weniger optimalen Ergebnissen zufrieden geben.

2 - Datenstrukturen

2 – Datenstrukturen

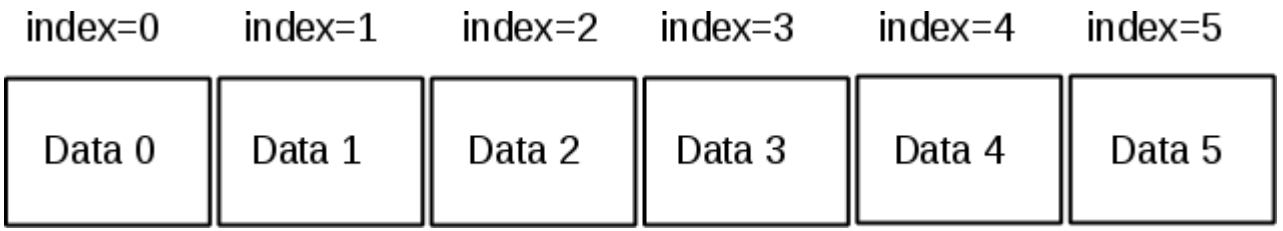
Kapitel 2 – Datenstrukturen

- Arrays
- Verkettete Listen
- Bäume
- Stacks
- Queues
- Graphen

2 - Datenstrukturen

Arrays

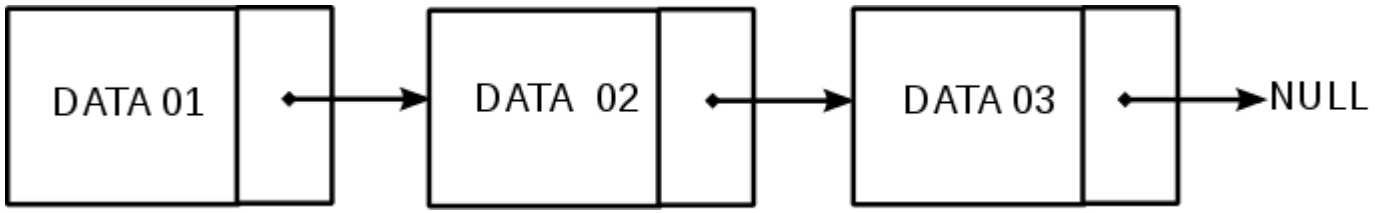
- Ein Array ist Speicher eines vorgegebenen Datentyps
- Jedes Speicherelement bekommt einen Index, über den man das Speicherelement abrufen oder beschreiben kann



2 - Datenstrukturen

Verkettete Listen

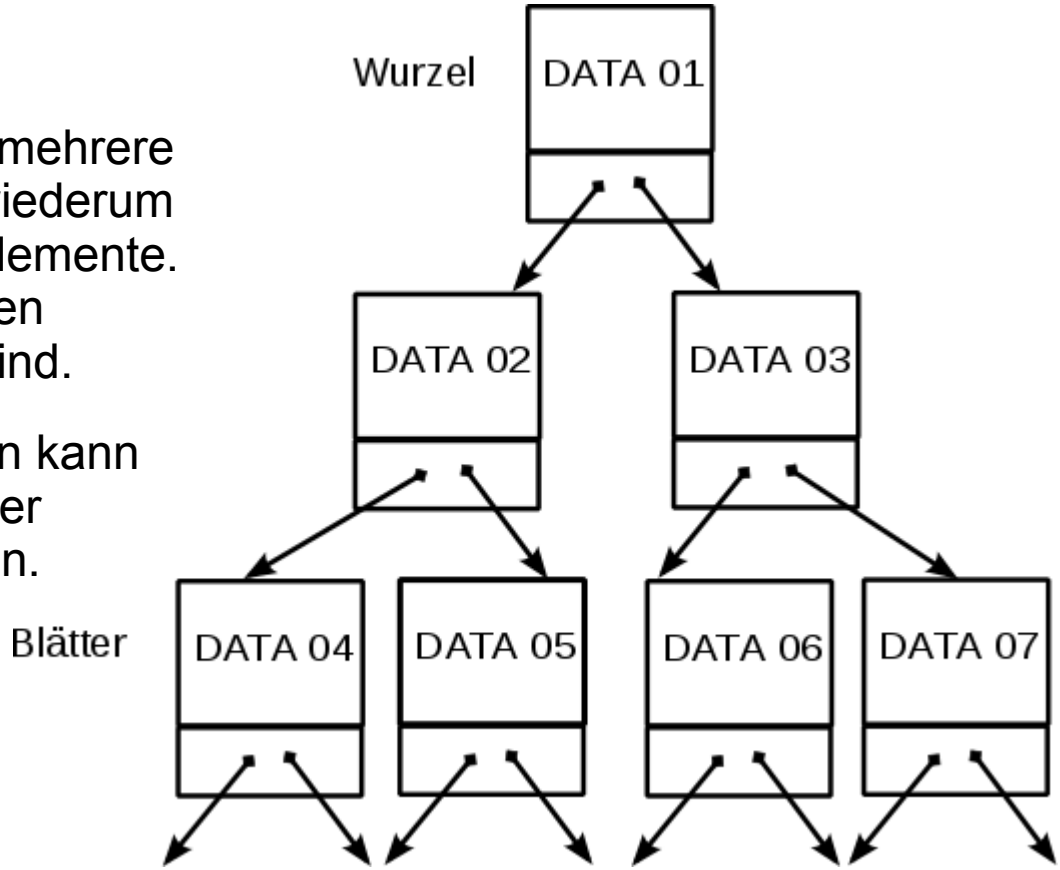
- Eine verkettete Liste ist eine Kette von Speicherelementen eines vorgegebenen Datentyps, die jeweils auf ihren Nachfolger (und Vorgänger) zeigen.
- Möchte man ein Element finden, so muss man die Kette vom Anfang aus, über die Zeiger der Elemente, bis zu dem zu suchenden Element entlang hangeln



2 - Datenstrukturen

Bäume

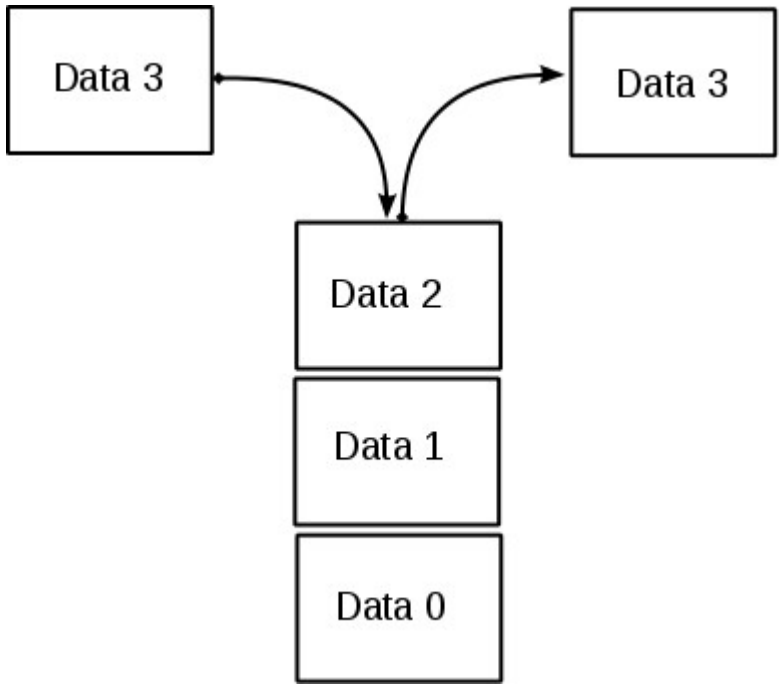
- Ein Baum hat eine Wurzel (ein Speicherelement), welches auf mehrere Nachfolgelemenete zeigt. Die wiederum zeigen auf mehrere Nachfolgeelemente. Das geht so weiter, bis die letzten Elemente, die Blätter, erreicht sind.
- Um einen Baum zu durchsuchen kann man ihn traversieren, d.h. in einer vordefinierten Weise durchlaufen.



2 - Datenstrukturen

Stacks

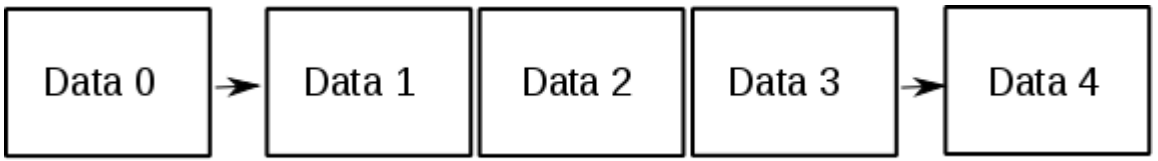
- Ein Stack ist ein Stapelspeicher. Jedes neue Speicherelement wird oben auf den Stapel gelegt.
- Wenn man ein Speicherelement auslesen möchte kann man i.allg. Nur das zuletzt aufgelegte Speicherelement wieder vom Stack nehmen.
- Ein Stack wird auch als LIFO-Speicher bezeichnet. Dabei steht LIFO für „Last In First Out“



2 - Datenstrukturen

Queues

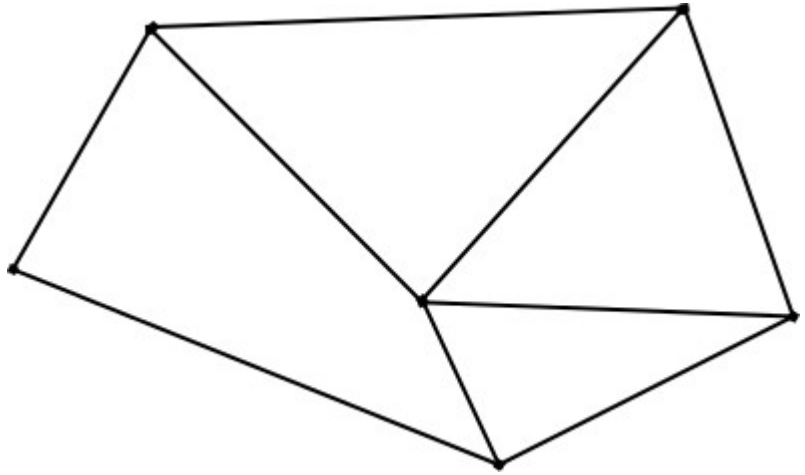
- Eine Queue ist eine Warteschlange. Das erste Element, welches in die Warteschlange geschrieben wird kommt auch als erstes wieder raus.
- Eine Queue wird auch als FIFO-Speicher bezeichnet. Dabei steht FIFO für „First In First Out“



2 - Datenstrukturen

Graphen

- Ein Graph ist eine Datenstruktur, in dem jeder Knoten eine Verbindung zu jedem anderen haben kann.
- Die Graphentheorie ist eine eigene Disziplin der Mathematik und ist ein weites Feld für Optimierungsaufgaben



3 - Sortierverfahren

3 – Sortierverfahren

Kapitel 3 – Sortierverfahren

- Bubblesort
- Mergesort

3 - Sortierverfahren

Bubblesort

- Bubblesort ist einer der einfachsten Sortieralgorithmen
- Man vergleicht benachbarte Elemente in einem Array und vertauscht die Elemente, wenn das Element mit dem kleineren Index größer (kleiner) ist als das mit dem größeren index. Dafür muss man das Array mehrfach durchlaufen

3 - Sortierverfahren

Mergesort

- Mergesort ist ein effizienter Sortieralgorithmus mit einer Komplexität von O(nlog(n)).
- Es ist ein Vertreter der „Divide and Conquer“ Strategie, bei dem man das Problem in Teilprobleme zerlegt.

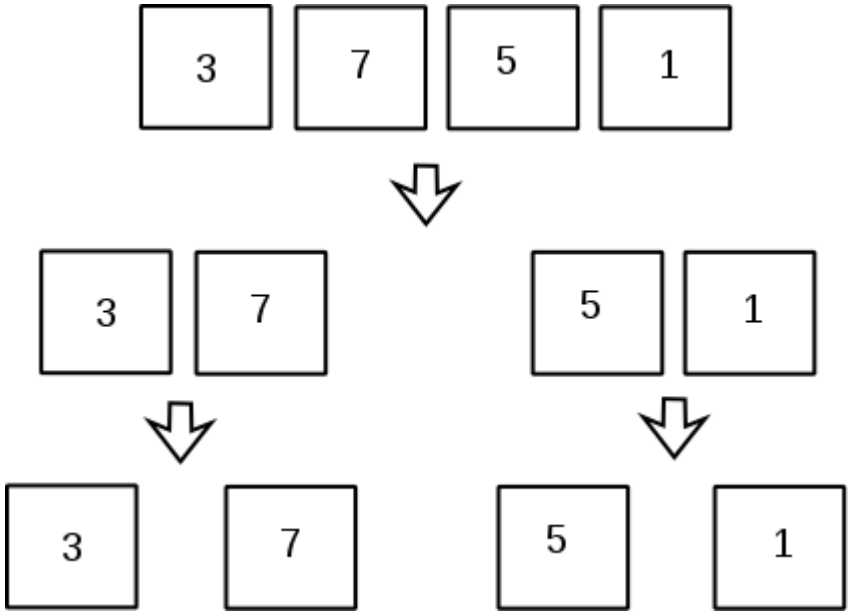
3 - Sortierverfahren

Mergesort

- Zunächst wird die Menge der zu sortierenden Zahlen in der Mitte zerteilt (split).
- Die resultierenden Mengen werden wiederum mit dem Mergesort sortiert
- Sind die beiden Mengen sortiert, so werden die Mengen wieder zu einer zusammengefügt (merge). Dabei zeigen zwei Indizes jeweils auf das erste Element der beiden Mengen. Das kleinere der Elemente, auf das die Indizes zeigen wird in die zusammengefügte Menge geschrieben und der Zeiger, auf den das kleinste Element zeigte um eine Position weiter gerückt.
- Das wird wiederholt, bis alle Elemente aus den beiden Mengen in die zusammengefügte Menge übertragen worden sind.

3 - Sortierverfahren

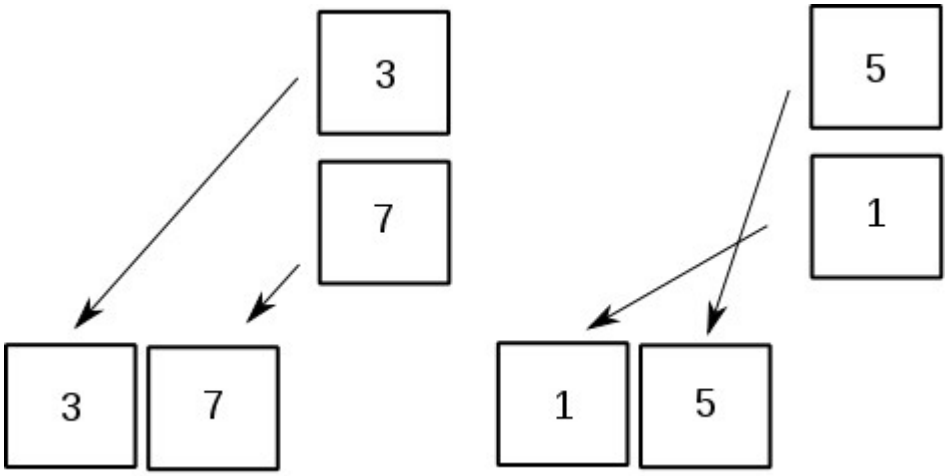
Mergesort (Split)



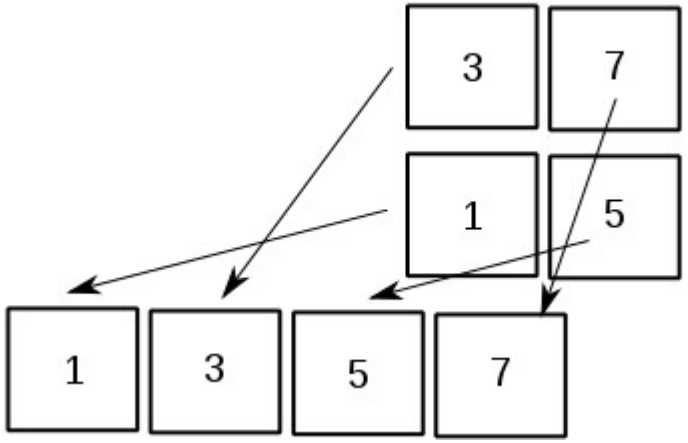
3 - Sortierverfahren

Mergesort (merge)

Merge Schritt 1



Merge Schritt 2



4 - Suchverfahren

4 – Suchverfahren

Kapitel 4 – Suchverfahren

- Lineare Suche
- Iterative Suche

4 - Suchverfahren

Lineare Suche

- Die Lineare Suche ist die einfachste Form der Suche.
- Jedes Element wird dabei nach vorgegebener Reihenfolge verglichen
- Der Aufwand für die lineare Suche ist O(n)

4 – Suchverfahren

Iterative Suche

- Die iterative Suche ist deutlich effizienter als die lineare Suche
- Sie funktioniert jedoch nur auf sortierten Mengen
- Man teilt die Menge in zwei (in etwa) gleich große Teile
- Dann prüft man, in welchem Teil das gesuchte Objekt ist
- Diese Menge teilt man wieder in zwei Teile, bis jedes Teil nur aus einem Element entsteht.
- Mit der Überprüfung in welchem Teil sich das zu suchende Element befindet hat man das gesuchte Element gefunden
- Die iterative Suche hat einen Aufwand von O(log(n))

5 - Hashverfahren

5 - Hashverfahren

Hashfunktion

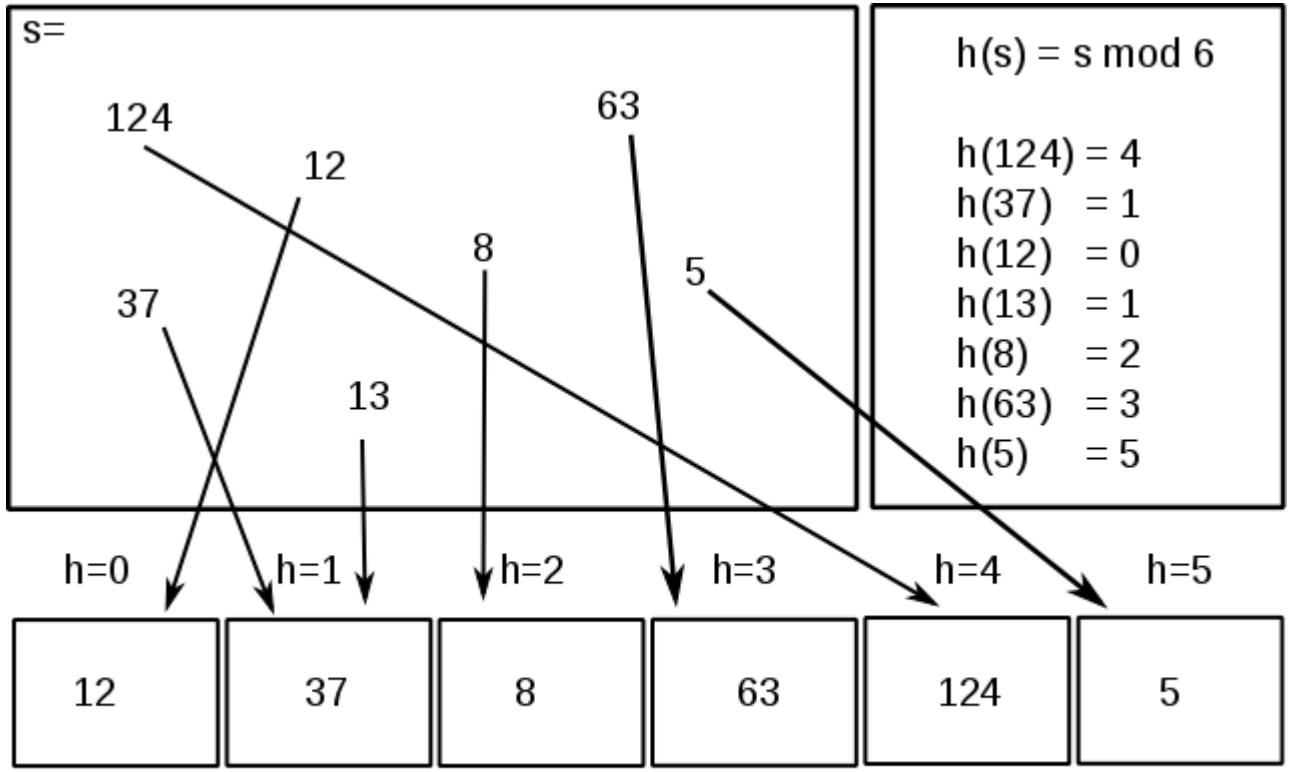
- Möchte man eine Menge von Daten in einer Hashtabelle speichern, so kann man n Speicherplätze reservieren und mittels einer Hashfunktion eine Adresse für diese Daten berechnen (ähnlich dem Index eines Arrays)
- Um möglichst unterschiedliche Adressen zu generieren wird die Adresse aus dem Datum s mittels der Modulofunktion hergeleitet.

$$h(s) = s \bmod n$$

- Dabei kann es zu Kollisionen kommen wenn h(s1) = h(s2).

5 - Hashverfahren

Beispiel:



Hashkollision bei h(37) = h(13)

5 - Hashverfahren

Sondieren (linear)

- Es kann vorkommen, dass zwei zu speichernde Datenwerte denselben Hashwert haben. Man spricht von einer Kollision.
- Um den zweiten Datenwert trotzdem speichern zu können, kann man z.B. linear sondieren, d.H. einen neuen Hashwert berechnen, bis man auf einen freien Speicherplatz trifft.

$$h^i(s) = (h(s) + i) \bmod n$$

5 - Hashverfahren

Sondieren (quadratisch)

- Dass sich beim linearen sondieren Datenwerte an bestimmten Stellen im Speicher häufen wird oftmals quadratisches sondieren genutzt:

$$h^i(s) = \left(\left(h(s) + (-1)^{(i+1)}\right)(i/2)^2\right) \bmod n$$