Grundlagen für das Seminar *Analyse von Algorithmen*

Eric Nöth

Universität Siegen

WS 15/16

Gliederung

Grundlagen

Was bekannt sein sollte:

1. Landau-Notation:

$$\mathcal{O}(f(n)), \Theta(f(n)), \Omega(f(n)), o(f(n))$$

Vorteilhaft:

Grundlagen

Was bekannt sein sollte:

1. Landau-Notation:

$$\mathcal{O}(f(n)), \Theta(f(n)), \Omega(f(n)), o(f(n))$$

2. Komplexe Zahlen $\mathbb{C}$

Vorteilhaft:

Grundlagen

Was bekannt sein sollte:

1. Landau-Notation:

$$\mathcal{O}(f(n)), \Theta(f(n)), \Omega(f(n)), o(f(n))$$

2. Komplexe Zahlen $\mathbb{C}$

Vorteilhaft:

1. Grundlagen Analysis

Literatur


Robert Sedgewick and Philippe Flajolet.
An introduction to the analysis of algorithms.
Addison-Wesley-Longman, 1996.


N.G. De Bruijn, S.O. Rice, and Donald Ervin Knuth.
Average height of planted plane trees.
Technical Report STAN-CS-71-218, Stanford University (Stanford,CA US), 1971.


Donald E. Knuth, J.H. Morris, and Vaughan R. Pratt.
Fast pattern matching in strings.
SIAM Journal of Computing, 6(2):323–350, 1977.


P. Flajolet and R. Sedgewick.
Analytic Combinatorics.
Cambridge University Press, 2009.

Motivation

Wozu Algorithmen analysieren?

1. Klassifizierung der Probleme und Algorithmen nach Schwierigkeit

Motivation

Wozu Algorithmen analysieren?

1. Klassifizierung der Probleme und Algorithmen nach Schwierigkeit
2. Vorhersage der Performanz, Vergleich von Algorithmen, Feineinstellung von Parametern

Motivation

Wozu Algorithmen analysieren?

1. Klassifizierung der Probleme und Algorithmen nach Schwierigkeit
2. Vorhersage der Performanz, Vergleich von Algorithmen, Feineinstellung von Parametern
3. Besseres Verständnis der Algorithmen, Finden möglicher Verbesserungen

Motivation

Wozu Algorithmen analysieren?

1. Klassifizierung der Probleme und Algorithmen nach Schwierigkeit
2. Vorhersage der Performanz, Vergleich von Algorithmen, Feineinstellung von Parametern
3. Besseres Verständnis der Algorithmen, Finden möglicher Verbesserungen
4. Mathematisches Interesse

Alan Turing

It is convenient to have a measure of the amount of work involved in a computing process, even though it be a very crude one. We may count up the number of times that various elementary operations are applied in the whole process and give them various weights. We might, for instance, count the number of additions, subtractions multiplications, divisions, recordings of numbers, and extractions of figures from tables.

Alan Turing, in
Rounding-off Errors in Matrix Processes (1947)

Der Knuthsche Ansatz, TAOCP ('60er)

Donald E. Knuth
The Art of Computer Programming.
Addison-Wesley-Longman, 1997.

1. Schreibe eine gute Implementierung

Der Knuthsche Ansatz, TAOCP ('60er)

Donald E. Knuth
The Art of Computer Programming.
Addison-Wesley-Longman, 1997.

1. Schreibe eine gute Implementierung
2. Identifiziere unbekannte Größen, die die Basisoperationen repräsentieren

Der Knuthsche Ansatz, TAOCP ('60er)

Donald E. Knuth
The Art of Computer Programming.
Addison-Wesley-Longman, 1997.

1. Schreibe eine gute Implementierung
2. Identifiziere unbekannte Größen, die die Basisoperationen repräsentieren
3. Determiniere die Kosten für jede Basisoperation

Der Knuthsche Ansatz, TAOCP ('60er)

Donald E. Knuth
The Art of Computer Programming.
Addison-Wesley-Longman, 1997.

1. Schreibe eine gute Implementierung
2. Identifiziere unbekannte Größen, die die Basisoperationen repräsentieren
3. Determiniere die Kosten für jede Basisoperation
4. Entwickle ein realistisches Inputmodell

Der Knuthsche Ansatz, TAOCP ('60er)

Donald E. Knuth
The Art of Computer Programming.
Addison-Wesley-Longman, 1997.

1. Schreibe eine gute Implementierung
2. Identifiziere unbekannte Größen, die die Basisoperationen repräsentieren
3. Determiniere die Kosten für jede Basisoperation
4. Entwickle ein realistisches Inputmodell
5. Analysiere die Häufigkeit der Ausführung der unbekannten Größen

Der Knuthsche Ansatz, TAOCP ('60er)

Donald E. Knuth
The Art of Computer Programming.
Addison-Wesley-Longman, 1997.

1. Schreibe eine gute Implementierung
2. Identifiziere unbekannte Größen, die die Basisoperationen repräsentieren
3. Determiniere die Kosten für jede Basisoperation
4. Entwickle ein realistisches Inputmodell
5. Analysiere die Häufigkeit der Ausführung der unbekannten Größen
6. Berechne die Gesamtlaufzeit:

$$\sum_{q} \text{häufigkeit}(q) \times \text{kosten}(q)$$

Der Knuthsche Ansatz, TAOCP ('60er)

Vor- und Nachteile:

1. Vorteile:

Der Knuthsche Ansatz, TAOCP ('60er)

Vor- und Nachteile:

1. Vorteile:
    1. Wissenschaftliche Grundlage AofA

Der Knuthsche Ansatz, TAOCP ('60er)

Vor- und Nachteile:

1. Vorteile:
   1. Wissenschaftliche Grundlage AofA
   2. Kann Performanz vorhersagen und Algorithmen vergleichen

Der Knuthsche Ansatz, TAOCP ('60er)

Vor- und Nachteile:

1. Vorteile:
   1. Wissenschaftliche Grundlage AofA
   2. Kann Performanz vorhersagen und Algorithmen vergleichen
2. Nachteile:

Der Knuthsche Ansatz, TAOCP ('60er)

Vor- und Nachteile:

1. Vorteile:
   1. Wissenschaftliche Grundlage AofA
   2. Kann Performanz vorhersagen und Algorithmen vergleichen
2. Nachteile:
   1. Modell könnte unrealistisch sein

Der Knuthsche Ansatz, TAOCP ('60er)

Vor- und Nachteile:

1. Vorteile:
   1. Wissenschaftliche Grundlage AofA
   2. Kann Performanz vorhersagen und Algorithmen vergleichen
2. Nachteile:
   1. Modell könnte unrealistisch sein
   2. Zu viele Details (bspw. Maschinenabhängigkeit!)

AHU, CLRS

Aho, Hobcroft, Ullmannn
The Design and Analysis of Algorithms.

Cormen, Leiserson, Rivest, Stein
Algorithms.

Ansatz:

1. Konzentration auf den *Worst-Case*, somit eine Laufzeitgarantie.

AHU, CLRS

Aho, Hobcroft, Ullmannn
The Design and Analysis of Algorithms.

Cormen, Leiserson, Rivest, Stein
Algorithms.

Ansatz:

1. Konzentration auf den *Worst-Case*, somit eine Laufzeitgarantie.
2. Landau-Notation.

AHU, CLRS

1. Vorteile:

AHU, CLRS

1. Vorteile:
    1. Bedeutend einfacher

AHU, CLRS

1. Vorteile:
   1. Bedeutend einfacher
   2. Garantierte Laufzeit

AHU, CLRS

1. Vorteile:
   1. Bedeutend einfacher
   2. Garantierte Laufzeit
2. Nachteil:
   Kann generell nicht genutzt werden, um Algorithmen zu vergleichen:

AHU, CLRS

1. Vorteile:
   1. Bedeutend einfacher
   2. Garantierte Laufzeit
2. Nachteil:
   Kann generell nicht genutzt werden, um Algorithmen zu vergleichen:
   - Quicksort: Worst-case $\mathcal{O}(n^2)$

AHU, CLRS

1. Vorteile:
   1. Bedeutend einfacher
   2. Garantierte Laufzeit
2. Nachteil:
   Kann generell nicht genutzt werden, um Algorithmen zu vergleichen:
   - Quicksort: Worst-case $\mathcal{O}(n^2)$
   - Mergesort: Worst-case $\mathcal{O}(n \log n)$

AHU, CLRS

1. Vorteile:
   1. Bedeutend einfacher
   2. Garantierte Laufzeit
2. Nachteil:
   Kann generell nicht genutzt werden, um Algorithmen zu vergleichen:
   - Quicksort: Worst-case $\mathcal{O}(n^2)$
   - Mergesort: Worst-case $\mathcal{O}(n \log n)$
   - Jedoch läuft Quicksort im Schnitt doppelt so schnell und braucht halb so viel Speicherplatz!

Analytische Kombinatorik!

- Nachteile Knuth:

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert
- Nachteile AHU/CLRS:

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert
- Nachteile AHU/CLRS:
  - Worst-Case könnte irrelevant sein

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert
- Nachteile AHU/CLRS:
  - Worst-Case könnte irrelevant sein
  - Landau-Notation kann schlecht vorhersagen oder verleichen

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert
- Nachteile AHU/CLRS:
  - Worst-Case könnte irrelevant sein
  - Landau-Notation kann schlecht vorhersagen oder verleichen
- Analytische Kombinatorik kann

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert
- Nachteile AHU/CLRS:
  - Worst-Case könnte irrelevant sein
  - Landau-Notation kann schlecht vorhersagen oder verleichen
- Analytische Kombinatorik kann
  - Bessere Modelle bereitstellen

Analytische Kombinatorik!

- Nachteile Knuth:
  - Modell könnte unrealistisch sein
  - Zu detailliert
- Nachteile AHU/CLRS:
  - Worst-Case könnte irrelevant sein
  - Landau-Notation kann schlecht vorhersagen oder verleichen
- Analytische Kombinatorik kann
  - Bessere Modelle bereitstellen
  - Irrelevante Details vermeiden

Mathematische Grundlagen

- Asymptotische Approximation von Summen [SF96, Kapitel 4]

Mathematische Grundlagen

- Asymptotische Approximation von Summen [SF96, Kapitel 4]
- Grundbegriffe der Funktionentheorie [FS09, Kapitel IV]
  Hier sollen einige Elemente der Funktionentheorie vorgestellt werden, die aus Sicht der Analyse von Algorithmen wichtig sind.

Mathematische Grundlagen

- Asymptotische Approximation von Summen [SF96, Kapitel 4]
- Grundbegriffe der Funktionentheorie [FS09, Kapitel IV]
  Hier sollen einige Elemente der Funktionentheorie vorgestellt werden, die aus Sicht der Analyse von Algorithmen wichtig sind.
- Asymptotik regulärer Sprachen und andere Beispiele [FS09, Kapitel V]
  Aufbauend auf den vorherigen Vortrag sollen einige Beispiele vorgestellt werden, die die Methoden verdeutlichen.

Mathematische Grundlagen

- Asymptotische Approximation von Summen [SF96, Kapitel 4]
- Grundbegriffe der Funktionentheorie [FS09, Kapitel IV]
  Hier sollen einige Elemente der Funktionentheorie vorgestellt werden, die aus Sicht der Analyse von Algorithmen wichtig sind.
- Asymptotik regulärer Sprachen und andere Beispiele [FS09, Kapitel V]
  Aufbauend auf den vorherigen Vortrag sollen einige Beispiele vorgestellt werden, die die Methoden verdeutlichen.
- *Algebraic generating functions in enumerative combinatorics and context-free languages*
  In diesem Vortrag soll auf das gleichnamige Paper von Mireille Bousquet-Melou eingegangen werden, in welchem verschiedene Generating functions klassifiziert werden und Zusammenhänge zwischen diesen Klassen und regulären bzw. kontextfreien Sprachen gezogen werden.

Bäume, Catalan-Zahlen und Binäre Suchbäume

- [SF96, Kapitel 6]

Bäume, Catalan-Zahlen und Binäre Suchbäume

- [SF96, Kapitel 6]
- Es sollen die Grundeigenschaften von Bäumen vorgestellt werden. Insbesondere soll detailliert auf die Catalanzahlen eingegangen werden, inklusive einiger anderer Interpretationen.

Bäume, Catalan-Zahlen und Binäre Suchbäume

- [SF96, Kapitel 6]
- Es sollen die Grundeigenschaften von Bäumen vorgestellt werden. Insbesondere soll detailliert auf die Catalanzahlen eingegangen werden, inklusive einiger anderer Interpretationen.
- Der Vortrag ist wichtig und sollte auf keinen Fall fehlen.

Andere Baumklassen

- Hier sollen kombinatorische Eigenschaften von ungewurzelten, gelabellten und anderer Baumklassen vorgestellt werden.

Andere Baumklassen

- Hier sollen kombinatorische Eigenschaften von ungewurzelten, gelabellten und anderer Baumklassen vorgestellt werden.
- Zusammenarbeit mit Thema 1 erwünscht.

Die durchschnittliche Höhe von Bäumen

- [DBRK71]

Die durchschnittliche Höhe von Bäumen

- [DBRK71]
- Dieses Thema ist mathematisch anspruchsvoll.

Die durchschnittliche Höhe von Bäumen

- [DBRK71]
- Dieses Thema ist mathematisch anspruchsvoll.
- Kann gut in Einzelarbeit bearbeitet werden.

Die durchschnittliche Höhe von Bäumen

- [DBRK71]
- Dieses Thema ist mathematisch anspruchsvoll.
- Kann gut in Einzelarbeit bearbeitet werden.
- Übersichtlich, da nur genau ein Beweis gezeigt werden soll.

Die durchschnittliche Höhe von Bäumen

- [DBRK71]
- Dieses Thema ist mathematisch anspruchsvoll.
- Kann gut in Einzelarbeit bearbeitet werden.
- Übersichtlich, da nur genau ein Beweis gezeigt werden soll.
- Priorität: Niedrig.

Permutationen

Uniform verteilte Permutationen sind für viele Algorithmen ein natürliches Inputmodell. Die Laufzeit eines Algorithmus hängt dann beispielsweise von der durchschnittlichen Anzahl der Zyklen in einer Permutation ab. Literatur für beide Vorträge ist das siebte Kapitel von *Analysis of Algorithms*.

Grundeigenschaften von Permutationen

- *Grundeigenschaften von Permutationen*: In diesem Vortrag sollen verschiedene Parameter vorgestellt werden, die für Permutationen wichtig sind.

Grundeigenschaften von Permutationen

- *Grundeigenschaften von Permutationen*: In diesem Vortrag sollen verschiedene Parameter vorgestellt werden, die für Permutationen wichtig sind.
- *Average-Case Verhalten diverser Suchalgorithmen*: Aufbauend auf den vorherigen Vortrag, soll die durchschnittliche Komplexität von *Insertion Sort*, *Shellsort*, und *Selection Sort* vorgestellt werden.

Grundeigenschaften von Permutationen

- *Grundeigenschaften von Permutationen*: In diesem Vortrag sollen verschiedene Parameter vorgestellt werden, die für Permutationen wichtig sind.
- *Average-Case Verhalten diverser Suchalgorithmen*: Aufbauend auf den vorherigen Vortrag, soll die durchschnittliche Komplexität von *Insertion Sort*, *Shellsort*, und *Selection Sort* vorgestellt werden.
- Wenn möglich, sollen beide Vorträge zusammen bearbeitet werden.

Themenkomplex Strings

In diesem Themenkomplex sollen Algorithmen über Strings, also Sequenzen von Symbolen eines festen Alphabets, untersucht werden. Zentral ist die Suche nach einem String u in einem String v.

Themenkomplex Strings

1. *Grundeigenschaften von Strings*: Es sollen verschiedene simple Fragen beantwortet werden, beispielsweise wie oft ein String der Länge p in einem String der Länge n durchschnittlich vorkommt, oder wie lang die längste Sequenz von 0 in einem Bitstring durchschnittlich ist.

Themenkomplex Strings

1. *Grundeigenschaften von Strings*: Es sollen verschiedene simple Fragen beantwortet werden, beispielsweise wie oft ein String der Länge p in einem String der Länge n durchschnittlich vorkommt, oder wie lang die längste Sequenz von 0 in einem Bitstring durchschnittlich ist.
2. *Der Knuth-Morris-Pratt Algorithmus*: Der KMP-Algorithmus [KMP77] ist ein bekannter Algorithmus, der Vorkommen eines Strings u in v sucht. In diesem Vortrag soll der Algorithmus vorgestellt und analysiert werden.

Themenkomplex Strings

1. *Grundeigenschaften von Strings*: Es sollen verschiedene simple Fragen beantwortet werden, beispielsweise wie oft ein String der Länge p in einem String der Länge n durchschnittlich vorkommt, oder wie lang die längste Sequenz von 0 in einem Bitstring durchschnittlich ist.
2. *Der Knuth-Morris-Pratt Algorithmus*: Der KMP-Algorithmus [KMP77] ist ein bekannter Algorithmus, der Vorkommen eines Strings u in v sucht. In diesem Vortrag soll der Algorithmus vorgestellt und analysiert werden.
3. *Tries und LZW-Kompression*: *Tries* sind eine beliebte Datenstruktur zur Indexierung von Strings. Es sollen verschiedene *Tries* vorgestellt und analysiert werden.

Thememkomplex Abbildungen

1. *Das Geburtstagsparadoxon & das Sammelbilderproblem*: Hier sollen zwei bekannte Probleme der Kombinatorik vorgestellt werden.

Thememkomplex Abbildungen

1. *Das Geburtstagsparadoxon & das Sammelbilderproblem*: Hier sollen zwei bekannte Probleme der Kombinatorik vorgestellt werden.
2. *Hashing-Algorithmen*: Aufbauend auf den vorherigen Vortrag sollen verschiedene Hashing-Algorithmen vorgestellt und analysiert werden.

Thememkomplex Abbildungen

1. *Das Geburtstagsparadoxon & das Sammelbilderproblem*: Hier sollen zwei bekannte Probleme der Kombinatorik vorgestellt werden.
2. *Hashing-Algorithmen*: Aufbauend auf den vorherigen Vortrag sollen verschiedene Hashing-Algorithmen vorgestellt und analysiert werden.
3. *Zufällige Abbildungen und Faktorisieren via Pollard-Rho*: Zur Zufallszahlengenerierung könnten prinzipiell zufällige Methoden herangezogen werden. Hier soll dargestellt werden, wo dabei Probleme auftauchen können (Stichword: Erwartete Zyklenlänge). Aufbauend darauf soll die Laufzeit eines Faktorisierungsalgorithmus analysiert werden.

Thememkomplex Zahlentheoretische Algorithmen

1. Analyse von Pseudozufallszahlengeneratoren
   Es sollen verschiedene Pseudozufallszahlengeneratoren vorgestellt und analysiert werden.

Thememkomplex Zahlentheoretische Algorithmen

1. Analyse von Pseudozufallszahlengeneratoren
   Es sollen verschiedene Pseudozufallszahlengeneratoren vorgestellt und analysiert werden.
2. Anaylse von *Quadratisches Sieb*
   Das *quadratische Sieb* ist ein Algorithmus zur Faktorisierung zweier Zahlen. In diesem Vortrag soll der Algorithmus vorgestellt und analysiert werden.

Rekurrenzen

Definition

Eine *Rekurrenz* ist eine Gleichung, die rekursiv eine Zahlenfolge definiert.

Beispiel: Fibonacci-Zahlen

$$f_n = f_{n-1} + f_{n-2} \text{ für } n \geq 2 \text{ und } f_0 = 0,\ f_1 = 1 \tag{1}$$

Gesucht: Formel für f_n

Rekurrenzen

Rekurrenzen können Kosten eines Programms modellieren. Beispiel Quicksort:

$$\begin{aligned} \text{qsort}[] &= [] \\ \text{qsort}(x : xs) &= \text{qsort}[y \mid y \leftarrow xs, y <= x] + +[x] + +\text{qsort}[y \mid y \leftarrow xs, y > x] \end{aligned}$$

Wir haben

- $n - 1$ Vergleiche mit x und

Rekurrenzen

Rekurrenzen können Kosten eines Programms modellieren. Beispiel Quicksort:

$$\begin{aligned} \text{qsort}[] &= [] \\ \text{qsort}(x : xs) &= \text{qsort}[y \mid y \leftarrow xs, y <= x] + +[x] + +\text{qsort}[y \mid y \leftarrow xs, y > x] \end{aligned}$$

Wir haben

- $n - 1$ Vergleiche mit x und
- ein Subarray mit k, und eines mit $n - k - 1$ Elementen.

Rekurrenzen

Rekurrenzen können Kosten eines Programms modellieren. Beispiel Quicksort:

$$\begin{aligned} \text{qsort}[] &= [] \\ \text{qsort}(x : xs) &= \text{qsort}[y \mid y \leftarrow xs, y <= x] + +[x] + +\text{qsort}[y \mid y \leftarrow xs, y > x] \end{aligned}$$

Wir haben

- $n - 1$ Vergleiche mit x und
- ein Subarray mit k, und eines mit $n - k - 1$ Elementen.
- Jede Aufteilung tritt mit Wahrscheinlichkeit $1/n$ auf.

Rekurrenzen

Rekurrenzen können Kosten eines Programms modellieren. Beispiel Quicksort:

$$\begin{aligned} \text{qsort}[] &= [] \\ \text{qsort}(x : xs) &= \text{qsort}[y \mid y \leftarrow xs, y <= x] + +[x] + +\text{qsort}[y \mid y \leftarrow xs, y > x] \end{aligned}$$

Wir haben

- $n - 1$ Vergleiche mit x und
- ein Subarray mit k, und eines mit $n - k - 1$ Elementen.
- Jede Aufteilung tritt mit Wahrscheinlichkeit $1/n$ auf.

Daher

$$C_n = n - 1 + \frac{1}{n} \sum_{k=0}^{n-1} (C_k + C_{n-k-1}) \text{ und } C_0 = 0.$$

Rekurrenzen

Einfachster Versuch zur Kostenrechnung: Ein kleines Computerprogramm. Fibonaccizahlen

```
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)
```

(natürlich ist

```
memoizedFib = (map fib [0 ..] !!)
  where fib 0 = 0
        fib 1 = 1
        fib n = memoizedFib (n-2) + memoizedFib (n-1)
```

bedeutend schneller)

Lineare Rekurrenz erster Ordnung

Beispiel:

$$a_n = a_{n-1} + n \text{ mit } a_0 = 0$$

- $a_n = a_{n-2} + (n-1) + n$

Lineare Rekurrenz erster Ordnung

Beispiel:

$$a_n = a_{n-1} + n \text{ mit } a_0 = 0$$

- $a_n = a_{n-2} + (n-1) + n$
- $a_n = a_{n-3} + (n-2) + (n-1) + n$

Lineare Rekurrenz erster Ordnung

Beispiel:

$$a_n = a_{n-1} + n \text{ mit } a_0 = 0$$

- $a_n = a_{n-2} + (n-1) + n$
- $a_n = a_{n-3} + (n-2) + (n-1) + n$
- $a_n = a_0 + \sum_{i=1}^{n} i = n(n+1)/2$

Lineare Rekurrenz erster Ordnung

Beispiel:

$$a_n = 2a_{n-1} + 2^n \text{ mit } a_0 = 0$$

Division durch den Summationsfaktor 2^n!

Lineare Rekurrenz erster Ordnung

Wie finden wir den Summationsfaktor von $a_n = x_n a_{n-1} + \ldots$? Division durch $x_n x_{n-1} \cdots x_1$ Beispiel:

$$a_n = (1 + \frac{1}{n}) a_{n-1} + 2^n \text{ mit } a_0 = 0$$

Ergebnis: $a_n = n2^n$
Übung:

$$na_n = (n-2)a_{n-1} + 2, \text{ mit } a_1 = 1$$

Andere Rekurrenzen

- Erster Ordnung

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung
  - Linear $a_n = a_{n-1} + 2a_{n-2}$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung
  - Linear $a_n = a_{n-1} + 2a_{n-2}$
  - Nichtlinear $a_n = a_{n-1}a_{n-2} + \sqrt{a_{n-2}}$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung
  - Linear $a_n = a_{n-1} + 2a_{n-2}$
  - Nichtlinear $a_n = a_{n-1}a_{n-2} + \sqrt{a_{n-2}}$
  - variable Koeffizienten $a_n = na_{n-1} + (n-1)a_{n-2} + 1$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung
  - Linear $a_n = a_{n-1} + 2a_{n-2}$
  - Nichtlinear $a_n = a_{n-1}a_{n-2} + \sqrt{a_{n-2}}$
  - variable Koeffizienten $a_n = na_{n-1} + (n-1)a_{n-2} + 1$
- Höhere Ordnung $a_n = f(a_{n-1}, \ldots, a_{n-k})$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung
  - Linear $a_n = a_{n-1} + 2a_{n-2}$
  - Nichtlinear $a_n = a_{n-1}a_{n-2} + \sqrt{a_{n-2}}$
  - variable Koeffizienten $a_n = na_{n-1} + (n-1)a_{n-2} + 1$
- Höhere Ordnung $a_n = f(a_{n-1}, \ldots, a_{n-k})$
- *Full history* $a_n = f(a_{n-1}, \ldots, a_1)$

Andere Rekurrenzen

- Erster Ordnung
  - Linear $a_n = na_{n-1} + 1$
  - Nichtlinear $a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right)$
- Zweiter Ordnung
  - Linear $a_n = a_{n-1} + 2a_{n-2}$
  - Nichtlinear $a_n = a_{n-1}a_{n-2} + \sqrt{a_{n-2}}$
  - variable Koeffizienten $a_n = na_{n-1} + (n-1)a_{n-2} + 1$
- Höhere Ordnung $a_n = f(a_{n-1}, \ldots, a_{n-k})$
- *Full history* $a_n = f(a_{n-1}, \ldots, a_1)$
- *Divide-and-Conquer* $a_n = a_{\lfloor n/2 \rfloor} + a_{\lceil n/2 \rceil} + n$

Beispiel

Example

$$a_n = 5a_{n-1} - 6a_{n-2}, \text{ mit } a_0 = 0, a_1 = 1$$

Behauptung: $a_n = x^n$

Mit Erzeugenden Funktionen systematische Lösung!

Vorkommen von Divide-And-Conquer Rekurrenzen

- Binäre Suche

Vorkommen von Divide-And-Conquer Rekurrenzen

- Binäre Suche
- Mergesort

Vorkommen von Divide-And-Conquer Rekurrenzen

- Binäre Suche
- Mergesort
- Karatsuba

Vorkommen von Divide-And-Conquer Rekurrenzen

- Binäre Suche
- Mergesort
- Karatsuba
- Strassens Matrixmultiplikation

Das Mastertheorem

Theorem

Die Lösung zur Rekurrenz

$$a_n = \underbrace{a_{n/\beta+O(1)} + \ldots a_{n/\beta+O(1)}}_{\alpha \text{ viele}} + \Theta(n^\gamma (\log n)^\delta)$$

ist

$$a_n = \begin{cases} \Theta(n^\gamma (\log n)^\delta) & \text{falls } \gamma < \log_\beta \alpha \\ \Theta(n^\gamma (\log n)^{\delta+1}) & \text{falls } \gamma = \log_\beta \alpha \\ \Theta(n^{\log_\beta \alpha}) & \text{falls } \gamma > \log_\beta \alpha \end{cases} \tag{2}$$

Erzeugende Funktion

Definition (Erzeugende Funktion)

Die (gewöhnliche) *Erzeugende Funktion* einer Sequenz $(A_n)_{n\geq 0}$ (formal mit $A_n \in \mathbb{C}$, hier mit $A_n \in \mathbb{R}_0^+$) ist die formale Potenzreihe

$$A(z) = \sum_{n\geq 0} A_n z^n.$$

Erzeugende Funktion

Die EF ist einerseits ein algebraisches Objekt von eigenem Interesse. Interessiert uns andererseits A_n, so können wir das via

$$A_n := \frac{1}{n!} \frac{d^n}{dz^n} A(z)$$

berechnen (Stichwort: Taylorentwicklung). Schreibweise:

$$[z^n]A(z) = A_n$$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$
2. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$
2. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
3. $zA(z)$ die EF von $0, a_1, a_2, a_3, \ldots$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$
2. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
3. $zA(z)$ die EF von $0, a_1, a_2, a_3, \ldots$
4. $A'(z)$ die EF von $((n+1)a_{n+1})_{n\geq 0} = a_1, a_2, a_3, \ldots$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$
2. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
3. $zA(z)$ die EF von $0, a_1, a_2, a_3, \ldots$
4. $A'(z)$ die EF von $((n+1)a_{n+1})_{n\geq 0} = a_1, a_2, a_3, \ldots$
5. $\int_0^z A(t)\,dt$ die EF von $0, a_0, \frac{a_1}{2}, \frac{a_2}{3}, \ldots$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$
2. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
3. $zA(z)$ die EF von $0, a_1, a_2, a_3, \ldots$
4. $A'(z)$ die EF von $((n+1)a_{n+1})_{n\geq 0} = a_1, a_2, a_3, \ldots$
5. $\int_0^z A(t)\,dt$ die EF von $0, a_0, \frac{a_1}{2}, \frac{a_2}{3}, \ldots$
6. $A(z)B(z)$ die EF von $(\sum_{i=0}^{n} a_i b_{n-i})_{n\geq 0}$

Rechenregeln von EFs

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(cz)$ die EF von $(c^n a_n)_{n\geq 0}$
2. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
3. $zA(z)$ die EF von $0, a_1, a_2, a_3, \ldots$
4. $A'(z)$ die EF von $((n+1)a_{n+1})_{n\geq 0} = a_1, a_2, a_3, \ldots$
5. $\int_0^z A(t)\,dt$ die EF von $0, a_0, \frac{a_1}{2}, \frac{a_2}{3}, \ldots$
6. $A(z)B(z)$ die EF von $(\sum_{i=0}^{n} a_i b_{n-i})_{n\geq 0}$
7. $\frac{1}{1-z}A(z)$ die EF von $a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots$

Einige einfache Erzeugende Funktionen

$$1, 1, 1, 1, \ldots, 1, \ldots : \quad \frac{1}{1-z} = \sum_{n \geq 0} z^n$$

Einige einfache Erzeugende Funktionen

$$1, 1, 1, 1, \ldots, 1, \ldots : \quad \frac{1}{1-z} = \sum_{n \geq 0} z^n$$

$$0, 1, 2, 3, \ldots, n, \ldots : \quad \frac{z}{(1-z)^2} = \sum_{n \geq 1} n z^n$$

Einige einfache Erzeugende Funktionen

$$1, 1, 1, 1, \ldots, 1, \ldots: \quad \frac{1}{1-z} = \sum_{n \geq 0} z^n$$

$$0, 1, 2, 3, \ldots, n, \ldots: \quad \frac{z}{(1-z)^2} = \sum_{n \geq 1} n z^n$$

$$0, 0, 1, 3, 6, 10, \ldots, \binom{n}{2}, \ldots: \quad \frac{z^2}{(1-z)^3} = \sum_{n \geq 0} \binom{n}{2} z^n$$

Einige einfache Erzeugende Funktionen

$$1, 1, 1, 1, \ldots, 1, \ldots: \quad \frac{1}{1-z} = \sum_{n \geq 0} z^n$$

$$0, 1, 2, 3, \ldots, n, \ldots: \quad \frac{z}{(1-z)^2} = \sum_{n \geq 1} n z^n$$

$$0, 0, 1, 3, 6, 10, \ldots, \binom{n}{2}, \ldots: \quad \frac{z^2}{(1-z)^3} = \sum_{n \geq 0} \binom{n}{2} z^n$$

$$0, 1, 1/2, 1/3, 1/4, \ldots, \frac{1}{n}, \ldots: \quad \ln\left(\frac{1}{1-z}\right) = \sum_{n \geq 1} \frac{z^n}{n}$$

Einige einfache Erzeugende Funktionen

$$1, 1, 1, 1, \ldots, 1, \ldots: \quad \frac{1}{1-z} = \sum_{n \geq 0} z^n$$

$$0, 1, 2, 3, \ldots, n, \ldots: \quad \frac{z}{(1-z)^2} = \sum_{n \geq 1} n z^n$$

$$0, 0, 1, 3, 6, 10, \ldots, \binom{n}{2}, \ldots: \quad \frac{z^2}{(1-z)^3} = \sum_{n \geq 0} \binom{n}{2} z^n$$

$$0, 1, 1/2, 1/3, 1/4, \ldots, \frac{1}{n}, \ldots: \quad \ln\left(\frac{1}{1-z}\right) = \sum_{n \geq 1} \frac{z^n}{n}$$

$$0, 1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12} \ldots, H_n, \ldots: \quad \frac{1}{1-z} \ln\left(\frac{1}{1-z}\right) = \sum_{n \geq 1} H_n z^n$$

EF der harmonischen Zahlen

$$\sum_{n\geq 1} H_n z^n = \frac{1}{1-z} \ln\left(\frac{1}{1-z}\right) \tag{3}$$

Beweis.

$$\sum_{n\geq 0} z^n = \frac{1}{1-z}$$

$$\sum_{n\geq 0} \frac{1}{n} z^n = \int_0^z \frac{1}{1-t} dt = \ln\left(\frac{1}{1-z}\right)$$

$$\sum_{n\geq 1} H_n z^n = \frac{1}{1-z} \ln\left(\frac{1}{1-z}\right)$$

CAS: Maple, Sage

Mit Maple:

```
seq(sum(1/t, t=1..n), n=1..10);
series(1/(1-z) * ln(1/(1-z)), z=0, 11);
```

Mit Sage:

```
Sequence(sum(1/i for i in [1..n]) for n in [1..10])
(1/(1-x) * ln (1/(1-x))).taylor(x,0,10)
```

Weiteres Beispiel

Example (Ex. 3.3 in [SF96])

Beweise, dass

$$\sum_{i=1}^{n} H_i = (n+1)(H_{n+1} - 1).$$

Weiteres Beispiel

Example (Ex. 3.3 in [SF96])

Beweise, dass

$$\sum_{i=1}^{n} H_i = (n+1)(H_{n+1} - 1).$$

Da $\frac{1}{1-z} \ln\left(\frac{1}{1-z}\right)$ die EF der harmonischen Zahlen ist, ist

$$\frac{1}{(1-z)^2} \ln\left(\frac{1}{1-z}\right)$$

die EF der oben genannten Zahlen.

Weiteres Beispiel

Example (Ex. 3.3 in [SF96])

Beweise, dass

$$\sum_{i=1}^{n} H_i = (n+1)(H_{n+1} - 1).$$

Da $\frac{1}{1-z} \ln\left(\frac{1}{1-z}\right)$ die EF der harmonischen Zahlen ist, ist

$$\frac{1}{(1-z)^2} \ln\left(\frac{1}{1-z}\right)$$

die EF der oben genannten Zahlen.
Außerdem (siehe Folie 35):

$$\frac{1}{(1-z)^2} = \sum_{n\geq 0}(n+1)z^n \quad \text{und} \quad \ln\left(\frac{1}{1-z}\right) = \sum_{n\geq 1}\frac{z^n}{n}$$

Weiteres Beispiel

Somit

$$\begin{aligned}
[z^n]\frac{1}{(1-z)^2}\ln\left(\frac{1}{1-z}\right) &= \sum_{k=1}^{n}\frac{1}{k}(n+1-k) \\
&= (n+1)H_n - n \\
&= (n+1)(H_{n+1} - \frac{1}{n+1}) - n \\
&= (n+1)(H_{n+1} - 1)
\end{aligned}$$

EF für Rekurrenzen

Generelles Framework

- Mache die Rekurrenz gültig für alle n.

EF für Rekurrenzen

Generelles Framework

- Mache die Rekurrenz gültig für alle n.
- Multipliziere beide Seiten mit z^n und summiere über alle $n \in \mathbb{N}$.

EF für Rekurrenzen

Generelles Framework

- Mache die Rekurrenz gültig für alle n.
- Multipliziere beide Seiten mit z^n und summiere über alle $n \in \mathbb{N}$.
- Werte die Summe aus, um eine Gleichung für die EF zu erhalten.

EF für Rekurrenzen

Generelles Framework

- Mache die Rekurrenz gültig für alle n.
- Multipliziere beide Seiten mit z^n und summiere über alle $n \in \mathbb{N}$.
- Werte die Summe aus, um eine Gleichung für die EF zu erhalten.
- Löse die Gleichung, um eine explizite Formel für die EF zu erhalten (mit den Startwerten).

EF für Rekurrenzen

Generelles Framework

- Mache die Rekurrenz gültig für alle n.
- Multipliziere beide Seiten mit z^n und summiere über alle $n \in \mathbb{N}$.
- Werte die Summe aus, um eine Gleichung für die EF zu erhalten.
- Löse die Gleichung, um eine explizite Formel für die EF zu erhalten (mit den Startwerten).
- Entwickle die EF, um die Koeffizienten zu erhalten.

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n: $a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n: $a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$

Summation über alle n: $A(z) = 5zA(z) - 6z^2A(z) + z$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n:	$a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$
Summation über alle n:	$A(z) = 5zA(z) - 6z^2A(z) + z$
Auflösen nach $A(z)$:	$A(z) = \frac{z}{1-5z+6z^2}$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n:	$a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$
Summation über alle n:	$A(z) = 5zA(z) - 6z^2A(z) + z$
Auflösen nach $A(z)$:	$A(z) = \frac{z}{1-5z+6z^2}$
Faktorisieren:	$A(z) = \frac{c_0}{1-3z} + \frac{c_1}{1-2z}$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n:	$a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$
Summation über alle n:	$A(z) = 5zA(z) - 6z^2A(z) + z$
Auflösen nach $A(z)$:	$A(z) = \frac{z}{1-5z+6z^2}$
Faktorisieren:	$A(z) = \frac{c_0}{1-3z} + \frac{c_1}{1-2z}$
Somit schließlich:	$c_0 = 1, \quad c_1 = -1$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n:	$a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$
Summation über alle n:	$A(z) = 5zA(z) - 6z^2A(z) + z$
Auflösen nach $A(z)$:	$A(z) = \frac{z}{1-5z+6z^2}$
Faktorisieren:	$A(z) = \frac{c_0}{1-3z} + \frac{c_1}{1-2z}$
Somit schließlich:	$c_0 = 1, \quad c_1 = -1$
Also	$a_n = 3^n - 2^n$

EF für Rekurrenzen

Example

$$a_n = 5a_{n-1} - 6a_{n-2} \text{ mit } a_0 = 0,\ a_1 = 1$$

Rekurrenz für alle n:	$a_n = 5a_{n-1} - 6a_{n-2} + \delta_{n,1}$
Summation über alle n:	$A(z) = 5zA(z) - 6z^2A(z) + z$
Auflösen nach $A(z)$:	$A(z) = \frac{z}{1-5z+6z^2}$
Faktorisieren:	$A(z) = \frac{c_0}{1-3z} + \frac{c_1}{1-2z}$
Somit schließlich:	$c_0 = 1, \quad c_1 = -1$
Also	$a_n = 3^n - 2^n$

wegen $\sum_{n\geq 0} c^n z^n = \frac{1}{1-cz}$

EF für Rekurrenzen

Theorem (Theorem 3.3 in [SF96])

Falls a_n der Rekurrenz

$$a_n = x_1 a_{n-1} + x_2 a_{n-2} + \cdots + x_t a_{n-t}$$

für $n \geq t$ genügt, so ist die erzeugende Funktion

$$A(z) = \sum_{n \geq 0} a_n z^n$$

eine rationale Funktion $A(z) = f(z)/g(z)$, wobei das Nennerpolynom durch

$$g(z) = 1 - x_1 z - x_2 z^2 - \cdots - x_t z^t$$

gegeben ist und das Zählerpolynom durch die Anfangswerte gegeben ist.

EF für Rekurrenzen

Beweis. Multiplizieren wir beide Seiten der Rekurrenz mit z^n und summieren für $n \geq t$ erhalten wir

$$\sum_{n \geq t} a_n z^n = x_1 \sum_{n \geq t} a_{n-1} z^n + \ldots x_t \sum_{n \geq t} a_{n-t} z^n$$

EF für Rekurrenzen

Beweis. Multiplizieren wir beide Seiten der Rekurrenz mit z^n und summieren für $n \geq t$ erhalten wir

$$\sum_{n \geq t} a_n z^n = x_1 \sum_{n \geq t} a_{n-1} z^n + \ldots x_t \sum_{n \geq t} a_{n-t} z^n$$

Die linke Seite ist gleich $A(z)$ minus das Erzeugende Polynom für die Anfangswerte. Die erste Summe auf der rechten Seite ist gleich $zA(z)$ minus einem Polynom etc, und so gilt

$$A(z) - u_0(z) = x_1 z A(z) - u_1(z) + \cdots + x_t z^t A(z) - u_t(z)$$

wobei die Polynome $u_0(z), \ldots, u_t(z)$ alle vom Grad höchstens $t-1$ sind und nur von den Anfangswerten $a_0, \ldots, a_{t-1}$ abhängen.

EF für Rekurrenzen

Diese Funktionalgleichung ist linear. Auflösen nach $A(z)$ ergibt die explizite Form $A(z) = f(z)/g(z)$, wobei $g(z)$ wie im Theorem angekündigt ist und $f(z)$ gleich

$$f(z) = u_0(z) - u_1(z) - \dots u_t(z)$$

ist. □

EF für Rekurrenzen

Thm. 7 ermöglicht eine andere Formulierung für die Abhängigkeit von $f(z)$ von den Anfangswerten. Wir haben:

$$f(z) = A(z)g(z) \quad \text{und} \quad \deg(f) < t$$

Also

$$f(z) = g(z) \sum_{0 \le n < t} a_n z^n \; (\mod z^t)$$

EF und Rekurrenzen

Beispiel: Die Fibonacci-Zahlen

$$f_n = f_{n-1} + f_{n-2}$$

für $n \geq 2$ mit $f_0 = 0$ und $f_1 = 1$.

EF und Rekurrenzen

Beispiel: Die Fibonacci-Zahlen

$$f_n = f_{n-1} + f_{n-2}$$

für $n \geq 2$ mit $f_0 = 0$ und $f_1 = 1$.
Mit Theorem 7 erhalten wir so

$$g(z) = 1 - z - z^2, \quad f(z) = z$$

und somit schließlich mit $\phi = \frac{1+\sqrt{5}}{2}$ und $\psi = 1 - \phi$

$$F(z) = \frac{1}{\sqrt{5}} \left(\frac{\phi}{1 - \phi z} - \frac{\psi}{1 - \psi z} \right),$$

was zeigt, dass $f_n \sim \frac{1}{\sqrt{5}} \phi^n$ ist.

EF für Rekurrenzen

Example

$$a_n = 2a_{n-1} + a_{n-2} - 2a_{n-3}$$

für $n \geq 2$ und mit

1. $a_0 = 0,\ a_1 = a_2 = 1$
2. $a_0 = a_1 = a_2 = 1$

Quicksort

Wir wollen die durchschnittliche Laufzeit für Quicksort ermitteln (Buch S.109). Die Rekurrenz ist gegeben durch

$$nC_n = n(n+1) + 2 \sum_{1 \leq k \leq n} C_{k-1}$$

für $n \geq 1$ mit $C_0 = 0$. Wir definieren

$$C(z) = \sum_{n \geq 0} C_n z^n.$$

Quicksort

Wir wollen die durchschnittliche Laufzeit für Quicksort ermitteln (Buch S.109). Die Rekurrenz ist gegeben durch

$$nC_n = n(n+1) + 2 \sum_{1 \leq k \leq n} C_{k-1}$$

für $n \geq 1$ mit $C_0 = 0$. Wir definieren

$$C(z) = \sum_{n \geq 0} C_n z^n.$$

Multiplizieren mit z^n und Summation über alle n ergibt

$$\sum_{n \geq 1} nC_n z^n = \sum_{n \geq 1} n(n+1) z^n + 2 \sum_{n \geq 1} \sum_{1 \leq k \leq n} C_{k-1} z^n$$

Quicksort continued

$$\sum_{n\geq 1} nC_n z^n = \sum_{n\geq 1} n(n+1)z^n + 2\sum_{n\geq 1}\sum_{1\geq k\geq n} C_{k-1}z^n$$

Quicksort continued

$$\sum_{n\geq 1} nC_n z^n = \sum_{n\geq 1} n(n+1)z^n + 2\sum_{n\geq 1}\sum_{1\geq k\geq n} C_{k-1}z^n$$

Die linke Seite ist gleich $zC'(z)$ und der erste Term der rechten Seite ist gleich $2z/(1-z)^3$. Der letzte Term ist eine Konvolution partieller Summen, und somit gleich $zC(z)/(1-z)$ (siehe Folie 34).
Somit erhalten wir die gewöhnliche Differentialgleichung

Quicksort continued

$$\sum_{n\geq 1} nC_n z^n = \sum_{n\geq 1} n(n+1)z^n + 2\sum_{n\geq 1}\sum_{1\geq k\geq n} C_{k-1}z^n$$

Die linke Seite ist gleich $zC'(z)$ und der erste Term der rechten Seite ist gleich $2z/(1-z)^3$. Der letzte Term ist eine Konvolution partieller Summen, und somit gleich $zC(z)/(1-z)$ (siehe Folie 34).
Somit erhalten wir die gewöhnliche Differentialgleichung

$$C'(z) = \frac{2}{(1-z)^3} + 2\frac{C(z)}{1-z}$$

Lösen der Differentialgleichung ergibt

$$C(z) = \frac{2}{(1-z)^2} \ln \frac{1}{1-z}$$

Quicksort continued

$$\sum_{n\geq 1} nC_n z^n = \sum_{n\geq 1} n(n+1)z^n + 2\sum_{n\geq 1}\sum_{1\geq k\geq n} C_{k-1}z^n$$

Die linke Seite ist gleich $zC'(z)$ und der erste Term der rechten Seite ist gleich $2z/(1-z)^3$. Der letzte Term ist eine Konvolution partieller Summen, und somit gleich $zC(z)/(1-z)$ (siehe Folie 34).
Somit erhalten wir die gewöhnliche Differentialgleichung

$$C'(z) = \frac{2}{(1-z)^3} + 2\frac{C(z)}{1-z}$$

Lösen der Differentialgleichung ergibt

$$C(z) = \frac{2}{(1-z)^2} \ln \frac{1}{1-z}$$

Somit

$$C_n = [z^n]\frac{2}{(1-z)^2} \ln \frac{1}{1-z} = 2(N+1)(H_{n+1}-1)$$

EF und Zählen

Ein klassisches Beispiel ist die *Wechselgeldfrage* (nach Polya): Auf wie viele Möglichkeiten kann ein Dollar/Euro bei gegebener Münzmenge zurückgegeben werden? Für den Eurofall erhalten wir

$$D(z) = \sum_{c_1,c_2,c_5,c_{10},c_{20},c_{50}} z^{c_1+2c_2+5c_5+10c_{10}+20c_{20}+50c_{50}}$$

und somit

EF und Zählen

Ein klassisches Beispiel ist die *Wechselgeldfrage* (nach Polya): Auf wie viele Möglichkeiten kann ein Dollar/Euro bei gegebener Münzmenge zurückgegeben werden? Für den Eurofall erhalten wir

$$D(z) = \sum_{c_1,c_2,c_5,c_{10},c_{20},c_{50}} z^{c_1+2c_2+5c_5+10c_{10}+20c_{20}+50c_{50}}$$

und somit

$$\begin{aligned} D(z) &= \sum_n z^n \sum_n z^{2n} \sum_n z^{5n} \sum_n z^{10n} \sum_n z^{20n} \sum_n z^{50n} \\ &= \frac{1}{(1-z)(1-z^2)(1-z^5)(1-z^{10})(1-z^{20})(1-z^{50})} \end{aligned}$$

EF und Zählen

Ein klassisches Beispiel ist die *Wechselgeldfrage* (nach Polya): Auf wie viele Möglichkeiten kann ein Dollar/Euro bei gegebener Münzmenge zurückgegeben werden? Für den Eurofall erhalten wir

$$D(z) = \sum_{c_1,c_2,c_5,c_{10},c_{20},c_{50}} z^{c_1+2c_2+5c_5+10c_{10}+20c_{20}+50c_{50}}$$

und somit

$$\begin{aligned} D(z) &= \sum_n z^n \sum_n z^{2n} \sum_n z^{5n} \sum_n z^{10n} \sum_n z^{20n} \sum_n z^{50n} \\ &= \frac{1}{(1-z)(1-z^2)(1-z^5)(1-z^{10})(1-z^{20})(1-z^{50})} \end{aligned}$$

Damit schließlich

$$[z^{100}]D(z) = 4562$$

Das amerikanische System (1,5,10,25 Cent) bietet 242 Möglichkeiten (mit 2 Cent-Münze 3546 Möglichkeiten).

Exponentielle EF

Betrachte die EF der Permutationen

$$\sum_{n \geq 0} n! z^n.$$

Wie wir aus der Analysis wissen, hat diese Potenzreihe einen Konvergenzradius von 0.

Exponentielle EF

Betrachte die EF der Permutationen

$$\sum_{n \geq 0} n! z^n.$$

Wie wir aus der Analysis wissen, hat diese Potenzreihe einen Konvergenzradius von 0.
Häufig, wenn wir einem Objekt der Größe n einzelne Label verpassen, konvergiert die entsprechende EF möglicherweise nicht.

Exponentielle EF

Betrachte die EF der Permutationen

$$\sum_{n\geq 0} n! z^n.$$

Wie wir aus der Analysis wissen, hat diese Potenzreihe einen Konvergenzradius von 0.
Häufig, wenn wir einem Objekt der Größe n einzelne Label verpassen, konvergiert die entsprechende EF möglicherweise nicht. Daher definieren wir die sogenannte *exponentielle erzeugende Funktion* einer Sequenz $(A_n)_{n\geq 0}$ als

$$A(z) = \sum_{k\geq 0} \frac{A_k}{k!} z^k \tag{4}$$

Prinzipiell können beide EF verwendet werden, jedoch wird für gelabellte Objekte meist die exponentielle EF verwendet.

Wichtige eEF

$$1, 1, 1, \cdots : \quad e^z = \sum_{n \geq 0} \frac{z^n}{n!}$$

Wichtige eEF

$1, 1, 1, \cdots$: $\quad e^z = \sum_{n \geq 0} \frac{z^n}{n!}$

$1, 2, 3, \cdots$: $\quad ze^z = \sum_{n \geq 0} \frac{z^n}{(n-1)!}$

Wichtige eEF

$1, 1, 1, \cdots: \quad e^z = \sum_{n \geq 0} \frac{z^n}{n!}$

$1, 2, 3, \cdots: \quad ze^z = \sum_{n \geq 0} \frac{z^n}{(n-1)!}$

$1, 1, 2, 6, 24, \ldots, n!, \cdots: \quad \frac{1}{1-z} = \sum_{n \geq 0} \frac{n! z^n}{n!}$

Rechenregeln eEF

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$

Rechenregeln eEF

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
2. $zA(z)$ die EF von $(a_n)_{n\geq 1}$

Rechenregeln eEF

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
2. $zA(z)$ die EF von $(a_n)_{n\geq 1}$
3. $A'(z)$ die EF von $(na_n)_{n\geq 0}$

Rechenregeln eEF

Sei $A(z) = \sum_{n\geq 0} a_n z^n$, $B(z) = \sum_{n\geq 0} b_n z^n$ und $c \in \mathbb{R}$. Dann ist

1. $A(z) + B(z)$ die EF von $(a_n + b_n)_{n\geq 0}$
2. $zA(z)$ die EF von $(a_n)_{n\geq 1}$
3. $A'(z)$ die EF von $(na_n)_{n\geq 0}$
4. $A(z)B(z)$ die EF von $(\sum_{i=0}^{n} \binom{n}{k} a_i b_{n-i})_{n\geq 0}$

Bivariate EF

Häufig müssen wir eine Funktion zweier Variablen n, k analysieren. Bei der Analyse eines Algorithmus ist dann n die Größe und k sind die Kosten eines Prozesses.

Bivariate EF

Definition

Gegeben eine doppelt indexierte Sequenz $(a_{nk})_{n,k\geq 0}$, heißt die Funktion

$$A(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} a_{nk} z^n u^k$$

die *bivariate* Erzeugendenfunktion (bEF) der Sequenz $(a_{nk})_{n,k\geq 0}$.

Bivariate EF

Definition

Gegeben eine doppelt indexierte Sequenz $(a_{nk})_{n,k\geq 0}$, heißt die Funktion

$$A(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} a_{nk} z^n u^k$$

die *bivariate* Erzeugendenfunktion (bEF) der Sequenz $(a_{nk})_{n,k\geq 0}$.

Sei $\mathcal{P}$ eine kombinatorische Struktur und für $p \in \mathcal{P}$ sei $\mathrm{cost}(p)$ eine Kostenfunktion. Dann wird

$$P(z,u) = \sum_{p\in\mathcal{P}} z^{|p|} u^{\mathrm{cost}(p)} = \sum_n \sum_k p_{nk} z^n u^k.$$

Bivariate EF

Wir schreiben auch

$$P(z,u) = \sum_n p_n(u) z^n \quad \text{wobei } p_n(u) = [z^n]P(z,u) = \sum_k p_{nk} u^k$$

und

$$P(z,u) = \sum_n q_k(z) u^k \quad \text{wobei } q_k(z) = [u^k]P(z,u) = \sum_n p_{nk} z^n.$$

Es sei angemerkt, dass

$$P(z,1) = \sum_{p \in \mathcal{P}} z^{|p|} = \sum_n p_n(1) z^n = \sum_k q_k(z)$$

die gewöhnliche EF ist, die $\mathcal{P}$ aufzählt.

Beispiel: Die Binomialverteilung

Sei $\mathcal{B}$ die Menge aller Binärstrings, und sei die *Kosten* eines Strings gleich der Anzahl der Vorkommen von 1. Dann ist $(a_{nk})_{n,k\geq 0}$ die Anzahl der n-bit Strings mit k-vielen 1-Vorkommen und die zugehörige bEF ist

$$P(z,u) = \sum_n \sum_k \binom{n}{k} u^k z^n = \sum_n (1+u)^n z^n = \frac{1}{1-(1+u)z}$$

Beispiel: Die Binomialverteilung

Die horizontale Expansion.
Werden die Funktionswerte nach n mit Hilfe von $[z^n]P(z,u) = p_n(u)$ aufgeschlüsselt, so wird dies auch die *horizontale* Expansion genannt. Für die Binomialverteilung ist das dann durch

$$\begin{aligned}
&z^0(u^0)+\\
&z^1(u^0+u^1)+\\
&z^2(u^0+2u^1+u^2)+\\
&z^3(u^0+3u^1+3u^2+u^3)+\ldots
\end{aligned}$$

gegeben.

Beispiel: Die Binomialverteilung

Die horizontale Expansion.
Werden die Funktionswerte nach n mit Hilfe von $[z^n]P(z,u) = p_n(u)$ aufgeschlüsselt, so wird dies auch die *horizontale* Expansion genannt. Für die Binomialverteilung ist das dann durch

$$\begin{aligned} &z^0(u^0)+\\ &z^1(u^0+u^1)+\\ &z^2(u^0+2u^1+u^2)+\\ &z^3(u^0+3u^1+3u^2+u^3)+\dots \end{aligned}$$

gegeben.
Die vertikale Expansion. Analog ist die *vertikale* Expansion

$$\begin{aligned} &u^0(z^0+z^1+z^2+z^3\dots)+\\ &u^1(z^1+2z^2+3z^3\dots)+\\ &u^2(z^2+3z^3+6z^4\dots)+\dots \end{aligned}$$

Horizontale Momentenberechnung

Differenzieren wir $p_n(u)$ nach u und werten wir für $u = 1$ aus, erhalten wir

$$p_n'(1) = \sum_{k \geq 0} k p_{nk}$$

Die zugehörige EF ist

$$\frac{\partial P(z,u)}{\partial u}|_{u=1} = \sum_{p \in \mathcal{P}} \text{cost}(p) z^{|p|}$$

und wird auch *kumulative Erzeugendenfunktion* genannt.

Horizontale Momentenberechnung

Differenzieren wir $p_n(u)$ nach u und werten wir für $u = 1$ aus, erhalten wir

$$p'_n(1) = \sum_{k \geq 0} k p_{nk}$$

Die zugehörige EF ist

$$\frac{\partial P(z,u)}{\partial u}|_{u=1} = \sum_{p \in \mathcal{P}} \text{cost}(p) z^{|p|}$$

und wird auch *kumulative Erzeugendenfunktion* genannt.
Des Weiteren ist

$$\frac{p'_n(1)}{p_n(1)}$$

gleich den durchschnittlichen Kosten für eine Struktur der Größe n.

Horizontale Momentenberechnung

Zusammengefasst:

Theorem

Gegeben eine bivariate EF $P(z,u)$ für eine kombinatorische Struktur, so sind die durchschnittlichen Kosten für alle Strukturen der Größe n gleich

$$\frac{[z^n]\frac{\partial P(z,u)}{\partial u}|_{u=1}}{[z^n]P(z,1)}.$$

Horizontale Momentenberechnung

Zusammengefasst:

Theorem

Gegeben eine bivariate EF $P(z,u)$ für eine kombinatorische Struktur, so sind die durchschnittlichen Kosten für alle Strukturen der Größe n gleich

$$\frac{[z^n]\frac{\partial P(z,u)}{\partial u}|_{u=1}}{[z^n]P(z,1)}.$$

Beispielsweise ist

$$[z^n]\frac{1}{1-(1+u)z}|_{u=1} = [z^n]\frac{1}{1-2z} = 2^n$$

und

$$[z^n]\frac{\partial}{\partial u}\frac{1}{1-(1+u)z}|_{u=1} = [z^n]\frac{z}{(1-2z)^2} = n2^{n-1},$$

also ist die durchschnittliche Anzahl der 1-Bits gleich $n/2$.

Horizontale Momentenberechnung

Ähnlich kann gezeigt werden, dass die Varianz gleich

$$\frac{p_n''(1)}{p_n(1)} + \frac{p_n'(1)}{p_n(1)} - \left(\frac{p_n'(1)}{p_n(1)}\right)^2$$

ist.

Asymptotik linearer Rekurrenzen

Theorem (Thm. 4.1 in [SF96])

Sei $A(z) = f(z)/g(z)$ eine rationale EF, wobei $f(z)$ und $g(z)$ teilerfremd sind und $g(0) \neq 0$. Hat $A(z)$ einen eindeutigen Pol $1/\beta$ kleinsten Modulus mit Multiplizität ν, so ist

$$[z^n]\frac{f(z)}{g(z)} \sim C\beta^n n^{\nu-1}, \text{ wobei } C = \nu\frac{(-\beta)^\nu f(1/\beta)}{g^{(\nu)}(1/\beta)}.$$

Asymptotik linearer Rekurrenzen

Somit haben wir für lineare Rekurrenzen folgenden Algorithmus:

1. Leite $g(z)$ aus der Rekurrenz ab.

Asymptotik linearer Rekurrenzen

Somit haben wir für lineare Rekurrenzen folgenden Algorithmus:

1. Leite $g(z)$ aus der Rekurrenz ab.
2. Bestimme das Polynom $f(z)$.

Asymptotik linearer Rekurrenzen

Somit haben wir für lineare Rekurrenzen folgenden Algorithmus:

1. Leite $g(z)$ aus der Rekurrenz ab.
2. Bestimme das Polynom $f(z)$.
3. Kürze $f(z)/g(z)$.

Asymptotik linearer Rekurrenzen

Somit haben wir für lineare Rekurrenzen folgenden Algorithmus:

1. Leite $g(z)$ aus der Rekurrenz ab.
2. Bestimme das Polynom $f(z)$.
3. Kürze $f(z)/g(z)$.
4. Finde die Nullstelle kürzesten Modulus.

Asymptotik linearer Rekurrenzen

Somit haben wir für lineare Rekurrenzen folgenden Algorithmus:

1. Leite $g(z)$ aus der Rekurrenz ab.
2. Bestimme das Polynom $f(z)$.
3. Kürze $f(z)/g(z)$.
4. Finde die Nullstelle kürzesten Modulus.
5. Ermittle die Koeffizienten mit Theorem 11.

Asymptotik linearer Rekurrenzen

Example

$$a_n = 2a_{n-1} + a_{n-2} - 2a_{n-3}$$

für $n \geq 2$ und mit $a_0 = 0$, $a_1 = a_2 = 1$

Analytische Kombinatorik - Übersicht

- Die Analysis beginnt mit kombinatorischen Konstruktionen.

Analytische Kombinatorik - Übersicht

- Die Analysis beginnt mit kombinatorischen Konstruktionen.
- Diese erzeugt eine EF.

Analytische Kombinatorik - Übersicht

- Die Analysis beginnt mit kombinatorischen Konstruktionen.
- Diese erzeugt eine EF.
- Transfertheorme ermöglichen quantitative Aussagen.

Analytische Kombinatorik - Übersicht

- Die Analysis beginnt mit kombinatorischen Konstruktionen.
- Diese erzeugt eine EF.
- Transfertheorme ermöglichen quantitative Aussagen.

Abbildung: Analytische Kombinatorik

Die symbolische Methode

Wir möchten nun, anstatt über Rekurrenzen, direkt über kombinatorische Struktur eines Objekts (beispielsweise eines Baums) eine EF erhalten.
Wir konzentrieren uns zunächst auf *ungelabellte Objekte*.

Erzeugende Funktion

Definition (Kombinatorische Klasse)

Eine *kombinatorische Klasse* $\mathcal{A}$ ist eine abzählbare Menge, auf welcher eine Größenfunktion $|\cdot| : \mathcal{A} \to \mathbb{N}_0$ definiert ist, so dass $\mathcal{A}_n := \{a \in \mathcal{A} \mid |a| = n\}$ für alle $n \in \mathbb{N}_0$ endlich ist.

Erzeugende Funktion

Definition (Kombinatorische Klasse)

Eine *kombinatorische Klasse* $\mathcal{A}$ ist eine abzählbare Menge, auf welcher eine Größenfunktion $|\cdot| : \mathcal{A} \to \mathbb{N}_0$ definiert ist, so dass $\mathcal{A}_n := \{a \in \mathcal{A} \mid |a| = n\}$ für alle $n \in \mathbb{N}_0$ endlich ist.

Definition (Zählsequenz)

Die *Zählsequenz* einer kombinatorische Klasse $\mathcal{A}$ ist die Sequenz $(A_n)_{n \geq 0}$, mit $A_n = \text{card}(\mathcal{A}_n)$.

Erzeugende Funktion

Definition (Kombinatorische Klasse)

Eine *kombinatorische Klasse* $\mathcal{A}$ ist eine abzählbare Menge, auf welcher eine Größenfunktion $|\cdot| : \mathcal{A} \to \mathbb{N}_0$ definiert ist, so dass $\mathcal{A}_n := \{a \in \mathcal{A} \mid |a| = n\}$ für alle $n \in \mathbb{N}_0$ endlich ist.

Definition (Zählsequenz)

Die *Zählsequenz* einer kombinatorische Klasse $\mathcal{A}$ ist die Sequenz $(A_n)_{n\geq 0}$, mit $A_n = \mathrm{card}(\mathcal{A}_n)$.

Die erzeugende Funktion einer kombinatorischen Klasse $\mathcal{A}$ ist die erzeugende Funktion der Sequenz $A_n = \mathrm{card}(\mathcal{A}_n)$.

Beispiele

Ein *Atom* (mit $\bullet$ notiert) ist ein Objekt der Größe 1. Die zugehörige EF ist $F(z) = z$.
Ein *neutrales Element* (mit ε notiert) ist ein Objekt der Größe 0. Die zugehörige EF ist $F(z) = 1$.

- Natürliche Zahlen

Beispiele

Ein *Atom* (mit $\bullet$ notiert) ist ein Objekt der Größe 1. Die zugehörige EF ist $F(z) = z$.
Ein *neutrales Element* (mit ε notiert) ist ein Objekt der Größe 0. Die zugehörige EF ist $F(z) = 1$.

- Natürliche Zahlen
- Endliche Mengen

Beispiele

Ein *Atom* (mit $\bullet$ notiert) ist ein Objekt der Größe 1. Die zugehörige EF ist $F(z) = z$.
Ein *neutrales Element* (mit ε notiert) ist ein Objekt der Größe 0. Die zugehörige EF ist $F(z) = 1$.

- Natürliche Zahlen
- Endliche Mengen
- Wörter über $\{0, 1\}$

Zulässige Konstruktionen

Seien $\mathcal{F}$ und $\mathcal{G}$ zwei kombinatorische Klassen, $F(z)$ und $G(z)$ die zugehörigen erzeugende Funktionen. Wir haben folgende Konstruktoren:

- Disjunkte Vereinigung: Ist

$$\mathcal{H} = \mathcal{F} \cup \mathcal{G} \quad \text{mit } \mathcal{F} \cap \mathcal{G} = \emptyset,$$

so ist $H(z) = F(z) + G(z)$.

Zulässige Konstruktionen

Seien $\mathcal{F}$ und $\mathcal{G}$ zwei kombinatorische Klassen, $F(z)$ und $G(z)$ die zugehörigen erzeugende Funktionen. Wir haben folgende Konstruktoren:

- Disjunkte Vereinigung: Ist

$$\mathcal{H} = \mathcal{F} \cup \mathcal{G} \quad \text{mit } \mathcal{F} \cap \mathcal{G} = \emptyset,$$

  so ist $H(z) = F(z) + G(z)$.

- Kartesisches Produkt: Ist

$$\mathcal{H} = \mathcal{F} \times \mathcal{G}$$

  so ist $H(z) = F(z) \cdot G(z)$.

Zulässige Konstruktionen

Seien $\mathcal{F}$ und $\mathcal{G}$ zwei kombinatorische Klassen, $F(z)$ und $G(z)$ die zugehörigen erzeugende Funktionen. Wir haben folgende Konstruktoren:

- Disjunkte Vereinigung: Ist

$$\mathcal{H} = \mathcal{F} \cup \mathcal{G} \quad \text{mit } \mathcal{F} \cap \mathcal{G} = \emptyset,$$

  so ist $H(z) = F(z) + G(z)$.
- Kartesisches Produkt: Ist

$$\mathcal{H} = \mathcal{F} \times \mathcal{G}$$

  so ist $H(z) = F(z) \cdot G(z)$.
- Sequenzkonstruktion: Ist

$$\mathcal{H} = \{\epsilon\} + \mathcal{F} + \mathcal{F} \times \mathcal{F} + \mathcal{F} \times \mathcal{F} \times \mathcal{F} \ldots,$$

  so ist $H(z) = \frac{1}{1-F(z)}$. Dabei muss darauf geachtet werden, dass $\mathcal{F}$ kein Element der Größe 0 enthält.

Beispiele

- Binäre Strings: Ein Wort über $\{0, 1\}$ ist entweder das leere Wort oder ein Bit gefolgt von einem binären String.

Beispiele

- Binäre Strings: Ein Wort über $\{0, 1\}$ ist entweder das leere Wort oder ein Bit gefolgt von einem binären String.
- Fibonacci-Zahlen: Sei $\mathcal{H} = \{a, bb\}$ mit $|a| = 1, |bb| = 2$, $H(z) = z + z^2$.
  $\mathcal{F}(z) = \text{SEQ}(\mathcal{H})$ und $F(z) = \frac{1}{1-z-z^2}$.

Beispiele

- Binäre Strings: Ein Wort über $\{0, 1\}$ ist entweder das leere Wort oder ein Bit gefolgt von einem binären String.
- Fibonacci-Zahlen: Sei $\mathcal{H} = \{a, bb\}$ mit $|a| = 1, |bb| = 2$, $H(z) = z + z^2$.
  $\mathcal{F}(z) = \text{SEQ}(\mathcal{H})$ und $F(z) = \frac{1}{1-z-z^2}$.
- Binäre Bäume: Wir zählen die Anzahl der internen Knoten. Ein *Binärbaum* ist dann entweder ein einziges Blatt oder ein innerer Knoten, an welchen zwei Binärbäume angehängt sind.

$$B(z) = 1 + zB^2(z)$$

Beispiele

- Binäre Strings: Ein Wort über $\{0, 1\}$ ist entweder das leere Wort oder ein Bit gefolgt von einem binären String.
- Fibonacci-Zahlen: Sei $\mathcal{H} = \{a, bb\}$ mit $|a| = 1, |bb| = 2$,
  $H(z) = z + z^2$.
  $\mathcal{F}(z) = \text{SEQ}(\mathcal{H})$ und $F(z) = \frac{1}{1-z-z^2}$.
- Binäre Bäume: Wir zählen die Anzahl der internen Knoten. Ein *Binärbaum* ist dann entweder ein einziges Blatt oder ein innerer Knoten, an welchen zwei Binärbäume angehängt sind.

$$B(z) = 1 + zB^2(z)$$

- Planare Bäume: Ein planarer Baum ist eine Wurzel, an welchem eine Sequenz planarer Bäume angehängt ist.

$$H(z) = \frac{z}{1 - H(z)}$$

Beispiele

Wie viele binäre Strings gibt es, in welchen 00 nicht vorkommt?
Sei B_{00} die Menge aller solcher Strings, $Z_0 = \{0\}$, $Z_1 = \{1\}$

Beispiele

Wie viele binäre Strings gibt es, in welchen 00 nicht vorkommt?
Sei B_{00} die Menge aller solcher Strings, $Z_0 = \{0\}$, $Z_1 = \{1\}$ Dann

$$B_{00} = \varepsilon + Z_0 + (Z_1 + Z_0 \times Z_1) \times B_{00}$$

Beispiele

Wie viele binäre Strings gibt es, in welchen 00 nicht vorkommt?
Sei B_{00} die Menge aller solcher Strings, $Z_0 = \{0\}$, $Z_1 = \{1\}$ Dann

$$B_{00} = \varepsilon + Z_0 + (Z_1 + Z_0 \times Z_1) \times B_{00}$$

$$B_{00} = \frac{1+z}{1-z-z^2}$$

Beispiele

Wie viele binäre Strings gibt es, in welchen 00 nicht vorkommt?
Sei B_{00} die Menge aller solcher Strings, $Z_0 = \{0\}$, $Z_1 = \{1\}$ Dann

$$B_{00} = \varepsilon + Z_0 + (Z_1 + Z_0 \times Z_1) \times B_{00}$$

$$B_{00} = \frac{1+z}{1-z-z^2}$$

Fibonaccizahlen!

Labels

Ein Objekt besteht nun aus n Atomen, jedoch sind diese mit Zahlen $[1..n]$ gelabellt.

Labels

Ein Objekt besteht nun aus n Atomen, jedoch sind diese mit Zahlen $[1..n]$ gelabellt.

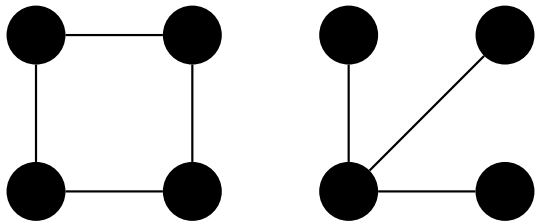


Abbildung: Unterschiedliche ungelabellte Graphen

Labels

Ein Objekt besteht nun aus n Atomen, jedoch sind diese mit Zahlen $[1..n]$ gelabellt.

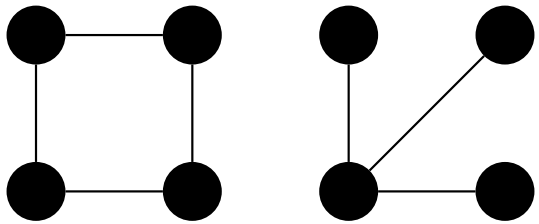


Abbildung: Unterschiedliche ungelabellte Graphen

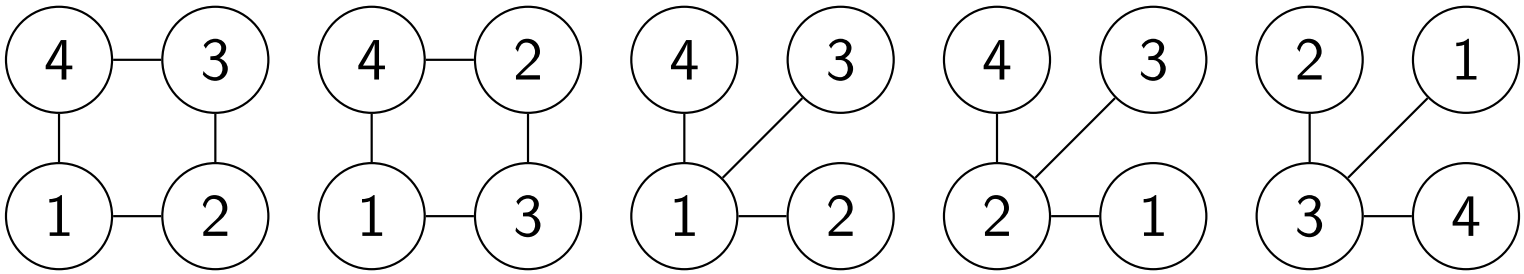


Abbildung: Unterschiedliche gelabellte Graphen

Labels

Ein Objekt besteht nun aus n Atomen, jedoch sind diese mit Zahlen $[1..n]$ gelabellt.

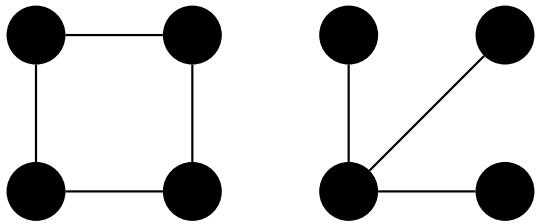


Abbildung: Unterschiedliche ungelabellte Graphen

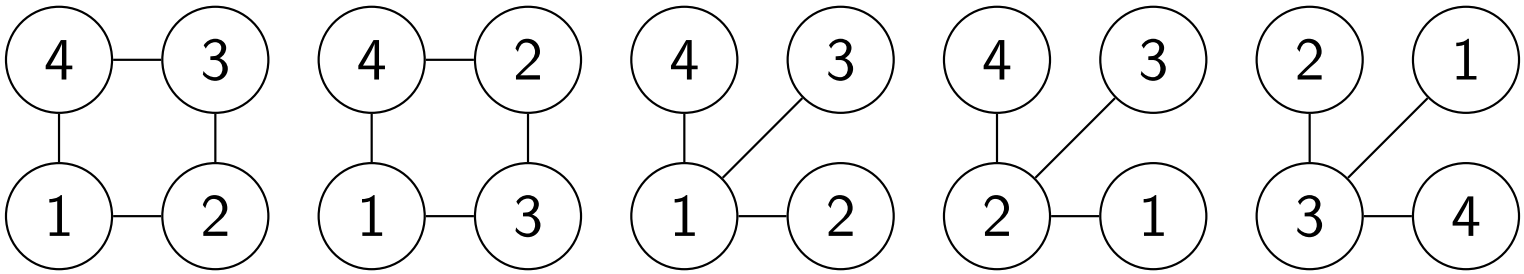


Abbildung: Unterschiedliche gelabellte Graphen

Die Anzahl der Möglichkeiten wächst rasant, wir verwenden daher die exponentielle EF.

Beispiele: Urne und Permutationen

Eine *Urne* ist eine Menge gelabellter Objekte.
Bei Urnen gibt es nur ein Objekt jeder Größe. Daher ist die entsprechende eEF

$$U(z) = \sum_{n \geq 1} \frac{z^n}{n!} = e^z$$

Beispiele: Urne und Permutationen

Eine *Urne* ist eine Menge gelabellter Objekte.
Bei Urnen gibt es nur ein Objekt jeder Größe. Daher ist die entsprechende eEF

$$U(z) = \sum_{n \geq 1} \frac{z^n}{n!} = e^z$$

Eine *Permutation* ist eine Sequenz gelabellter Objekte.
Bei Permutationen gibt es $n!$ viele Objekte jeder Größe.

$$P(z) = \sum_{n \geq 1} \frac{n! z^n}{n!} = \frac{1}{1-z}$$

Beispiel: Zyklus

Ein *Zyklus* ist eine zyklische Sequenz gelabellter Objekte.

Beispiel: Zyklus

Ein *Zyklus* ist eine zyklische Sequenz gelabellter Objekte.

$$C_n = (n-1)!$$

Beispiel: Zyklus

Ein *Zyklus* ist eine zyklische Sequenz gelabellter Objekte.

$$C_n = (n-1)!$$

$$C(z) = \sum_n \frac{(n-1)!z^n}{n!} = \sum_n \frac{z^n}{n}$$

Beispiel: Zyklus

Ein *Zyklus* ist eine zyklische Sequenz gelabellter Objekte.

$$C_n = (n-1)!$$

$$C(z) = \sum_n \frac{(n-1)!z^n}{n!} = \sum_n \frac{z^n}{n}$$

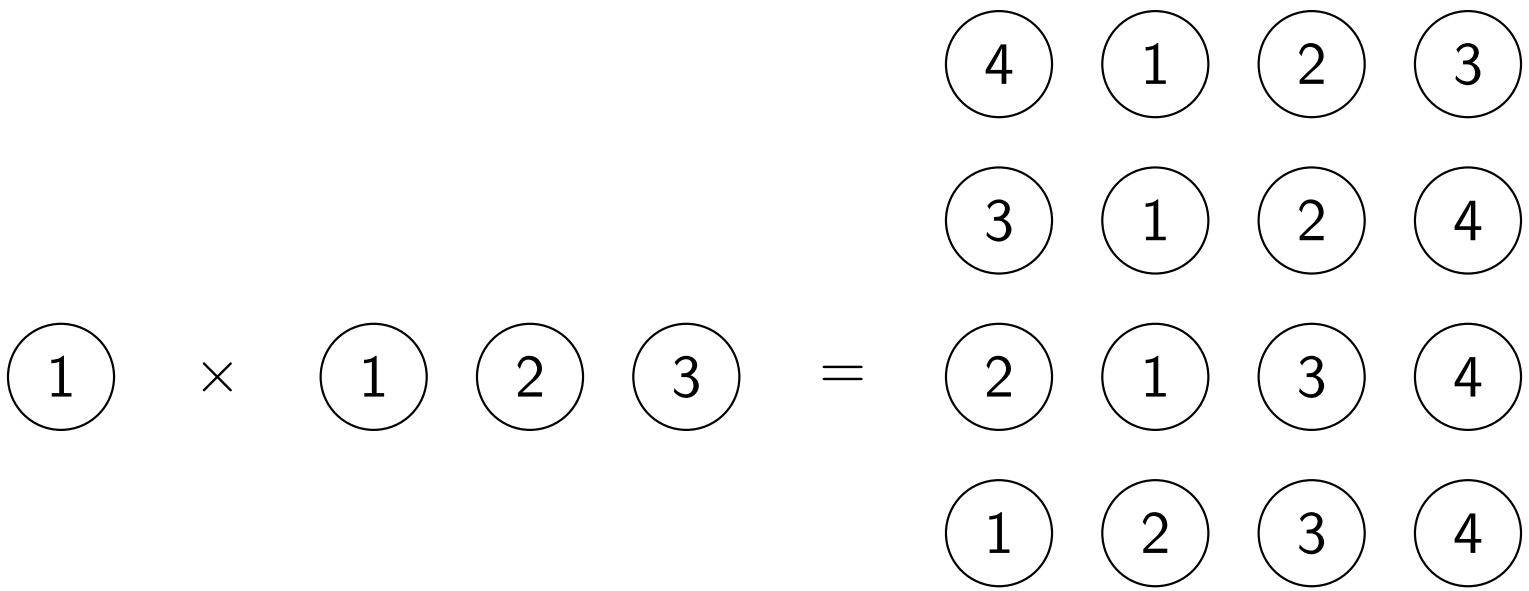
$$C(z) = \ln\left(\frac{1}{1-z}\right)$$

Karthesisches Produkt

Beim Karthesischen Produkt müssen wir *auf alle konsistenten Möglichkeiten neu labellen*. Beispiel:

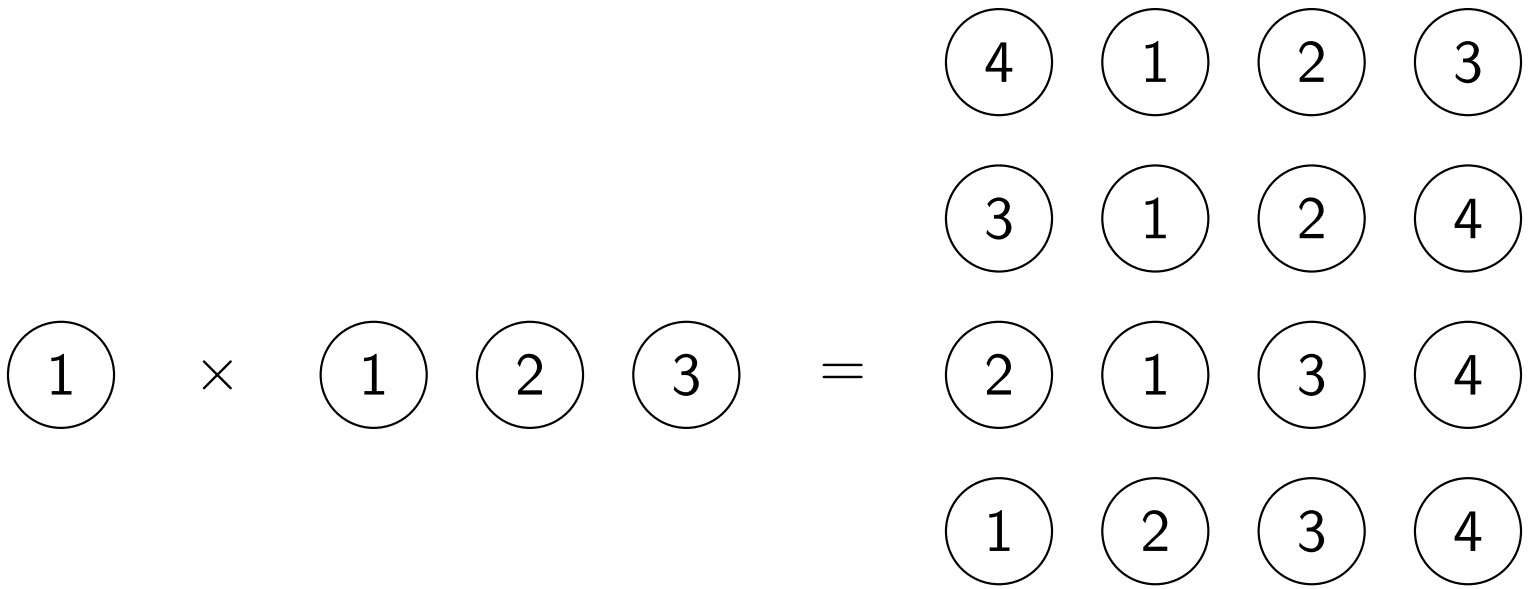
Karthesisches Produkt

Beim Karthesischen Produkt müssen wir *auf alle konsistenten Möglichkeiten neu labellen*. Beispiel:



Karthesisches Produkt

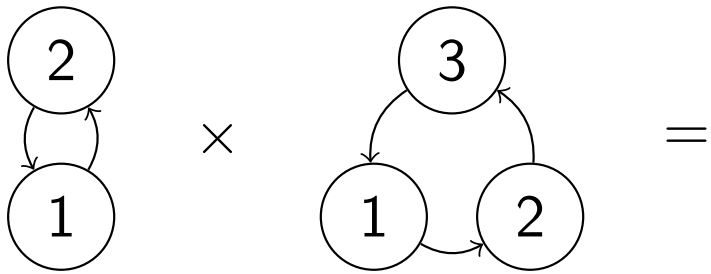
Beim Karthesischen Produkt müssen wir *auf alle konsistenten Möglichkeiten neu labellen*. Beispiel:



Die hinteren Elemente müssen aufsteigend sein.

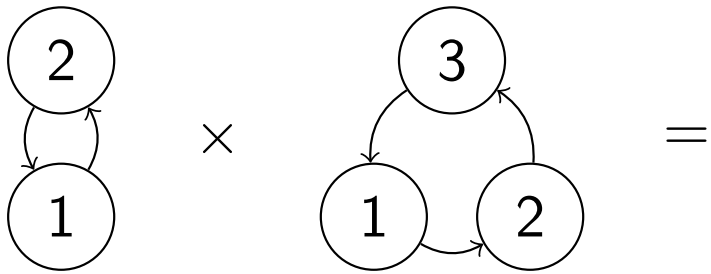
Karthesisches Produkt

Übung:



Karthesisches Produkt

Übung:



Zehn Möglichkeiten .

Konstruktoren

Seien $\mathcal{A}, \mathcal{B}$ komb. Kl. gelabelter Obj. und $A(z), B(z)$ die zugehörigen eEF.

Disjunkte Vereinigung: $A + B$ mit eEF

$$A(z) + B(z)$$

Konstruktoren

Seien $\mathcal{A}, \mathcal{B}$ komb. Kl. gelabelter Obj. und $A(z), B(z)$ die zugehörigen eEF.

Disjunkte Vereinigung: $A + B$ mit eEF

$$A(z) + B(z)$$

Labelled Product: $A \star B$ mit eEF

$$A(z) \cdot B(z)$$

Konstruktoren

Seien $\mathcal{A}, \mathcal{B}$ komb. Kl. gelabelter Obj. und $A(z), B(z)$ die zugehörigen eEF.

Disjunkte Vereinigung: $A + B$ mit eEF

$$A(z) + B(z)$$

Labelled Product: $A \star B$ mit eEF

$$A(z) \cdot B(z)$$

Sequenz: $\mathrm{SEQ}_k(A)$ (genau k Objekte) und $\mathrm{SEQ}(A)$

$$A^k(z) \text{ und } \frac{1}{1 - A(z)}$$

Konstruktoren

Seien $\mathcal{A}, \mathcal{B}$ komb. Kl. gelabelter Obj. und $A(z), B(z)$ die zugehörigen eEF.

Disjunkte Vereinigung: $A + B$ mit eEF

$$A(z) + B(z)$$

Labelled Product: $A \star B$ mit eEF

$$A(z) \cdot B(z)$$

Sequenz: $\mathrm{SEQ}_k(A)$ (genau k Objekte) und $\mathrm{SEQ}(A)$

$$A^k(z) \text{ und } \frac{1}{1 - A(z)}$$

Menge: $\mathrm{SET}_k(A)$ (genau k Objekte) und $\mathrm{SET}(A)$

$$\frac{A^k(z)}{n!} \text{ und } \exp(A(z))$$

Konstruktoren

Seien $\mathcal{A}, \mathcal{B}$ komb. Kl. gelabelter Obj. und $A(z), B(z)$ die zugehörigen eEF.

Disjunkte Vereinigung: $A + B$ mit eEF

$$A(z) + B(z)$$

Labelled Product: $A \star B$ mit eEF

$$A(z) \cdot B(z)$$

Sequenz: $\mathrm{SEQ}_k(A)$ (genau k Objekte) und $\mathrm{SEQ}(A)$

$$A^k(z) \text{ und } \frac{1}{1 - A(z)}$$

Menge: $\mathrm{SET}_k(A)$ (genau k Objekte) und $\mathrm{SET}(A)$

$$\frac{A^k(z)}{n!} \text{ und } \exp(A(z))$$

Zyklus: $\mathrm{CYC}_k(A)$ und $\mathrm{CYC}(A)$

$$A^k(z)/k \text{ und } \ln \frac{1}{1 - A(z)}$$

Beispiele

Urnen $U = \mathsf{SET}(z)$

Beispiele

Urnen $U = \mathrm{SET}(z)$

Zyklen $C = \mathrm{CYC}(z)$

Beispiele

Urnen $U = \mathrm{SET}(z)$

Zyklen $C = \mathrm{CYC}(z)$

Permutationen $P = \mathrm{SEQ}(z)$ oder $P = \epsilon + z \star P$

Beweise

$$\sum_{\gamma \in \mathcal{A} \times \mathcal{B}} \frac{z^{|\gamma|}}{|\gamma|!} =$$

Beweise

$$\sum_{\gamma \in \mathcal{A} \times \mathcal{B}} \frac{z^{|\gamma|}}{|\gamma|!} = \sum_{\alpha \in \mathcal{A}} \sum_{\beta \in \mathcal{B}} \binom{(|\alpha| + |\beta|)!}{|\alpha|!} \frac{z^{|\alpha|+|\beta|}}{(|\alpha| + |\beta|)!}$$

Beweise

$$\sum_{\gamma\in\mathcal{A}\times\mathcal{B}} \frac{z^{|\gamma|}}{|\gamma|!} = \sum_{\alpha\in\mathcal{A}}\sum_{\beta\in\mathcal{B}} \binom{(|\alpha|+|\beta|)!}{|\alpha|!} \frac{z^{|\alpha|+|\beta|}}{(|\alpha|+|\beta|)!}$$

$$= \sum_{\alpha\in\mathcal{A}} \frac{z^{|\alpha|}}{|\alpha|!} \sum_{\beta\in\mathcal{B}} \frac{z^{|\beta|}}{|\beta|!}$$

Beweise

$$\sum_{\gamma\in\mathcal{A}\times\mathcal{B}} \frac{z^{|\gamma|}}{|\gamma|!} = \sum_{\alpha\in\mathcal{A}}\sum_{\beta\in\mathcal{B}} \binom{(|\alpha|+|\beta|)!}{|\alpha|!} \frac{z^{|\alpha|+|\beta|}}{(|\alpha|+|\beta|)!}$$

$$= \sum_{\alpha\in\mathcal{A}} \frac{z^{|\alpha|}}{|\alpha|!} \sum_{\beta\in\mathcal{B}} \frac{z^{|\beta|}}{|\beta|!}$$

$$= A(z)B(z)$$

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

$$SC_1 = 1, SC_2 = 2$$

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

$$SC_1 = 1, SC_2 = 2$$

$$SC_3 = 1(1 \star 1 \star 1) + 3(1 \star 2) + 2(3)$$

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

$$SC_1 = 1, SC_2 = 2$$

$$SC_3 = 1(1 \star 1 \star 1) + 3(1 \star 2) + 2(3)$$

$$SC_4 = 1(1 \star 1 \star 1 \star 1) + 6(1 \star 1 \star 2) + 3(2 \star 2) + 8(1 \star 3) + 6(4) = 24$$

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

$$SC_1 = 1, SC_2 = 2$$

$$SC_3 = 1(1 \star 1 \star 1) + 3(1 \star 2) + 2(3)$$

$$SC_4 = 1(1 \star 1 \star 1 \star 1) + 6(1 \star 1 \star 2) + 3(2 \star 2) + 8(1 \star 3) + 6(4) = 24$$

$$SC = \mathrm{SET}(\mathrm{CYC}(z))$$

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

$$SC_1 = 1, SC_2 = 2$$

$$SC_3 = 1(1 \star 1 \star 1) + 3(1 \star 2) + 2(3)$$

$$SC_4 = 1(1 \star 1 \star 1 \star 1) + 6(1 \star 1 \star 2) + 3(2 \star 2) + 8(1 \star 3) + 6(4) = 24$$

$$SC = \mathrm{SET}(\mathrm{CYC}(z))$$

$$SC(z) = \exp\left(\ln \frac{1}{1-z}\right) = \frac{1}{1-z}$$

Mengen von Zyklen

Wie viele Mengen von Zyklen gelabellter Atome gibt es?

$$SC_1 = 1, SC_2 = 2$$

$$SC_3 = 1(1 \star 1 \star 1) + 3(1 \star 2) + 2(3)$$

$$SC_4 = 1(1 \star 1 \star 1 \star 1) + 6(1 \star 1 \star 2) + 3(2 \star 2) + 8(1 \star 3) + 6(4) = 24$$

$$SC = \mathrm{SET}(\mathrm{CYC}(z))$$

$$SC(z) = \exp\left(\ln \frac{1}{1-z}\right) = \frac{1}{1-z}$$

Eine Permutation ist eine Menge von Zyklen.

Permutationen

Standardrepräsentation

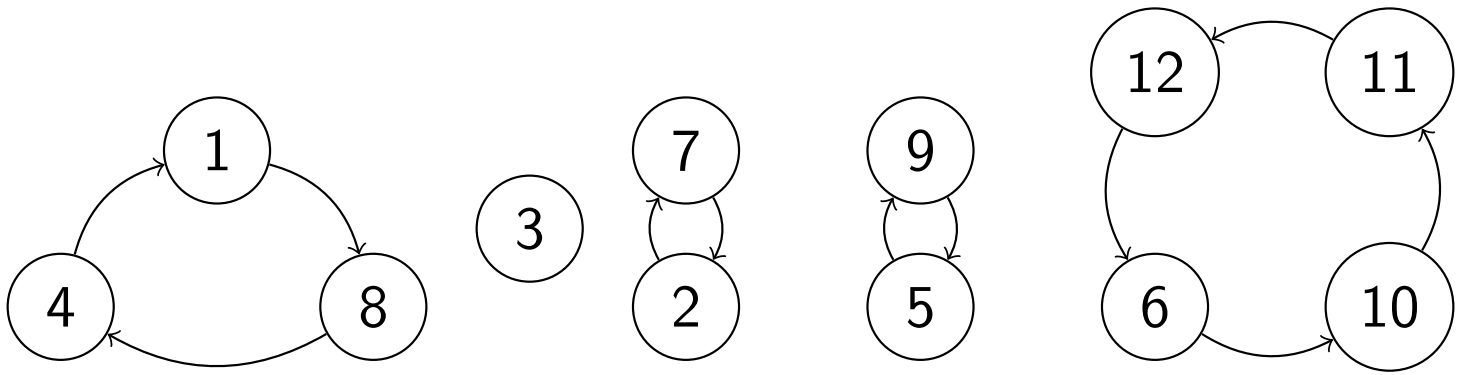
1	2	3	4	5	6	7	8	9	10	11	12
8	7	3	1	9	11	2	4	5	6	12	10

Permutationen

Standardrepräsentation

1	2	3	4	5	6	7	8	9	10	11	12
8	7	3	1	9	11	2	4	5	6	12	10

Zyklenrepräsentation



Analytische Transfers

Theorem (Taylors Theorem)

Falls $f(z)$ n-fach differenzierbar ist, so ist $[z^n]f(z) = \frac{f^{(n)}(0)}{n!}$.

Analytische Transfers

Theorem (Taylors Theorem)

Falls $f(z)$ n-fach differenzierbar ist, so ist $[z^n]f(z) = \frac{f^{(n)}(0)}{n!}$.

Zur Erinnerung: Theorem 11

Theorem

Sei $A(z) = f(z)/g(z)$ eine rationale EF, wobei $f(z)$ und $g(z)$ teilerfremd sind und $g(0) \neq 0$. Hat $A(z)$ einen eindeutigen Pol $1/\beta$ kleinsten Modulus mit Multiplizität ν, so ist

$$[z^n]\frac{f(z)}{g(z)} \sim C\beta^n n^{\nu-1}, \text{ wobei } C = \nu\frac{(-\beta)^\nu f(1/\beta)}{g^{(\nu)}(1/\beta)}.$$

Konvergenzradiustransfertheorem

Theorem (Konvergenzradiustransfertheorem, Thm. 5.5)

Falls $f(z)$ einen Konvergenzradius größer 1 hat, wobei $f(1) \neq 0$, dann ist

$$[z^n]\frac{f(z)}{(1-z)^\alpha} \sim f(1)\binom{n+\alpha-1}{n} \sim \frac{f(1)}{\Gamma(\alpha)}n^{\alpha-1}$$

für jegliche $\alpha \notin \{0, -1, -2, \dots\}$.

Konvergenzradiustransfertheorem

Theorem (Konvergenzradiustransfertheorem, Thm. 5.5)

Falls $f(z)$ einen Konvergenzradius größer 1 hat, wobei $f(1) \neq 0$, dann ist

$$[z^n]\frac{f(z)}{(1-z)^\alpha} \sim f(1)\binom{n+\alpha-1}{n} \sim \frac{f(1)}{\Gamma(\alpha)}n^{\alpha-1}$$

für jegliche $\alpha \notin \{0, -1, -2, \dots\}$.

Dabei ist

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,dt$$

die Verallgemeinerung der Fakultätsfunktion (es gilt $\alpha\Gamma(\alpha) = \Gamma(\alpha+1)$).
Wichtiger Wert: $\Gamma(1/2) = \sqrt{\pi}$.

Beispiele Konvergenzradiustransfertheorem

Korollar. Falls $f(z)$ einen Konvergenzradius größer ρ hat (mit $f(\rho) \neq 0$), so ist

$$[z^n]\frac{f(z)}{(1-z/\rho)^\alpha} \sim \frac{f(\rho)}{\Gamma(\alpha)}\rho^n n^{\alpha-1}$$

für jegliche $\alpha \notin \{0, -1, -2, \dots\}$.

Beispiele Konvergenzradiustransfertheorem

Korollar. Falls $f(z)$ einen Konvergenzradius größer ρ hat (mit $f(\rho) \neq 0$), so ist

$$[z^n]\frac{f(z)}{(1-z/\rho)^\alpha} \sim \frac{f(\rho)}{\Gamma(\alpha)}\rho^n n^{\alpha-1}$$

für jegliche $\alpha \notin \{0, -1, -2, \dots\}$.
Catalanzahlen:

$$C(z) = \frac{1-\sqrt{1-4z}}{2z}$$

Beispiele Konvergenzradiustransfertheorem

Korollar. Falls $f(z)$ einen Konvergenzradius größer ρ hat (mit $f(\rho) \neq 0$), so ist

$$[z^n]\frac{f(z)}{(1-z/\rho)^\alpha} \sim \frac{f(\rho)}{\Gamma(\alpha)}\rho^n n^{\alpha-1}$$

für jegliche $\alpha \notin \{0, -1, -2, \dots\}$.
Catalanzahlen:

$$C(z) = \frac{1-\sqrt{1-4z}}{2z}$$

$$\rho = \frac{1}{4}, \quad \alpha = -\frac{1}{2}, \quad f(z) = -\frac{1}{2},$$

Beispiele Konvergenzradiustransfertheorem

Korollar. Falls $f(z)$ einen Konvergenzradius größer ρ hat (mit $f(\rho) \neq 0$), so ist

$$[z^n]\frac{f(z)}{(1-z/\rho)^\alpha} \sim \frac{f(\rho)}{\Gamma(\alpha)}\rho^n n^{\alpha-1}$$

für jegliche $\alpha \notin \{0, -1, -2, \dots\}$.
Catalanzahlen:

$$C(z) = \frac{1-\sqrt{1-4z}}{2z}$$

$$\rho = \frac{1}{4}, \quad \alpha = -\frac{1}{2}, \quad f(z) = -\frac{1}{2},$$

Mit $\Gamma(-1/2) = -2\Gamma(1/2) = -2\sqrt{\pi}$ erhalten wir schließlich

$$C_n = [z^n]C(z) = \frac{1}{\sqrt{\pi n^3}}4^n$$