



#### Mathe 1

Charlotte P., Lena W., Vera C., Christian K. | 21. Juni 2018

# ITI WAGNER & IPD TICHY $\sum_{m=1}^{\infty} q_m(\omega) \int_0^{A_n} \left\{ (1+\mathrm{i}\eta) \frac{\mathrm{d}^2}{\mathrm{d}x^2} \left[ k(x) \frac{\mathrm{d}^2 \psi_{m'}(x)}{\mathrm{d}x^2} \right] - \omega^2 \psi_m(x) \right.$ $\times \left[ \rho_l(x) + \frac{\pi}{4} \rho_f b^2(x) \Gamma(\beta(x,\omega),\alpha(x)) \right] \right\} \psi_n(x) \, \mathrm{d}x$ $= \omega^2 \int_0^{A_n} \left\{ \hat{\theta}_\mathrm{B}(\omega)(x+L_0) \left[ \rho_l(x) + \frac{M}{4} \psi_l b^2(x) \Gamma(\beta(x,\omega),\alpha(x)) \right] \right.$ $\left. \alpha(x) \right] + \frac{\pi}{4} \rho_f b^2(x) \Delta \left( \beta(x,\omega), \frac{1}{b(x)} \left| \sum_{m=1}^{\infty} q_m(\omega) \psi_m(x) \right. \right.$ $\left. + \hat{\theta}_\mathrm{B}(\omega)(x+L_0) \right|, \alpha(x) \right)$ $\times \left[ \sum_{m=1}^{\infty} q_m(\omega) \psi_m(x) + \hat{\theta}_\mathrm{B}(\omega)(x+L_0) \right] \right\} \psi_n(x) \, \mathrm{d}x. \quad (10)$

## Gliederung



- Big Integer
- Exponentiation by squaring
- Kombinatorik
- Spieltheorie

Spieltheorie

## Big integer



- die maximale Zahl ist größer als integer?
- nehme long long
- die Zahl ist größer als long long
- ????????????????????????????(Panik)

## Big integer - Java nutzen



- import java.math.BigInteger
- Konstruktor: BigInteger(String val)
- Methoden:
  - BigInteger add(BigInteger val)
  - BigInteger multiply(BigInteger val)
  - BigInteger subtract(BigInteger val)
  - BigInteger abs()
  - BigInteger compareTo(BigInteger val)
  - BigInteger[] divideAndRemainder(BigInteger val) (Division mit Rest)
  - BigInteger gcd(BigInteger val)





#### Schriftliche Addition, Beispiel:

String x = "12035"

#### String y = "389"

vector 
$$v_X = (5,3,0,2,1)$$

vector 
$$v_y = (9,8,3,0,0)$$

vector 
$$V_Z = ($$

$$v_z = (4)$$
, Ubertrag = 1

$$v_z = (4, 2)$$
, Übertrag = 1

$$v_z = (4, 2, 4)$$
, Übertrag = 0

$$v_z = (4, 2, 4, 2)$$
, Übertrag = 0

$$v_z = (4, 2, 4, 2, 1)$$
, Übertrag = 0



```
Schriftliche Addition, Beispiel:
```

String x = "12035"

String y = "389"

vector  $v_x = (5,3,0,2,1)$ 

vector  $v_v = (9,8,3,0,0)$ 

vector  $v_z = ()$ 



21. Juni 2018



```
Schriftliche Addition, Beispiel:
```

String x = "12035"

String y = "389"

vector  $v_x = (5,3,0,2,1)$ 

vector  $v_y = (9,8,3,0,0)$ 

vector  $v_z = ()$ 

 $v_z = (4)$ , Übertrag = 1

 $v_z = (4, 2)$ , Ubertrag = 1

 $v_z = (4, 2, 4)$ , Ubertrag = 0

 $v_z = (4, 2, 4, 2)$ , Ubertrag = 0

 $v_z = (4, 2, 4, 2, 1)$ , Übertrag = 0

In  $v_{\scriptscriptstyle Z}$  steht das gespiegelte Ergebnis der Additior





```
Schriftliche Addition, Beispiel:
```

String x = "12035"

String y = "389"

vector  $v_x = (5,3,0,2,1)$ 

vector  $v_y = (9,8,3,0,0)$ 

vector  $v_z = ()$ 

 $v_z = (4)$ , Übertrag = 1

 $v_z = (4, 2)$ , Übertrag = 1

 $V_z = (4, 2, 4)$ , Ubertrag = 0

 $v_z = (4, 2, 4, 2)$ , Ubertrag = 0

 $v_z = (4, 2, 4, 2, 1)$ , Übertrag = 0

In  $v_z$  steht das gespiegelte Ergebnis der Addition



5/63



```
Schriftliche Addition, Beispiel:
```

String x = "12035"

String y = "389"

vector  $v_x = (5,3,0,2,1)$ 

vector  $v_y = (9,8,3,0,0)$ 

vector  $v_z = ()$ 

 $v_z = (4)$ , Übertrag = 1

 $v_z = (4, 2)$ , Übertrag = 1

 $v_z = (4, 2, 4)$ , Übertrag = 0

 $v_z = (4, 2, 4, 2)$ , Ubertrag = 0

 $v_z = (4, 2, 4, 2, 1)$ , Übertrag = 0

In  $v_z$  steht das gespiegelte Ergebnis der Addition



Spieltheorie



Schriftliche Addition, Beispiel:

String x = "12035"

String y = "389"

vector  $v_x = (5,3,0,2,1)$ 

vector  $v_y = (9,8,3,0,0)$ 

vector  $v_z = ()$ 

 $v_z = (4)$ , Übertrag = 1

 $v_z = (4, 2)$ , Übertrag = 1

 $v_z = (4, 2, 4)$ , Übertrag = 0

 $v_z = (4, 2, 4, 2)$ , Übertrag = 0

 $v_z = (4, 2, 4, 2, 1)$ , Ubertrag = 0

In  $v_z$  steht das gespiegelte Ergebnis der Addition





Schriftliche Addition, Beispiel:

String x = "12035"

String y = "389"

vector  $v_x = (5,3,0,2,1)$ 

vector  $v_y = (9,8,3,0,0)$ 

vector  $v_z = ()$ 

 $v_z = (4)$ , Übertrag = 1

 $v_z = (4, 2)$ , Übertrag = 1

 $v_z = (4, 2, 4)$ , Übertrag = 0

 $v_z = (4, 2, 4, 2)$ , Übertrag = 0

 $v_z = (4, 2, 4, 2, 1)$ , Übertrag = 0

In  $v_z$  steht das gespiegelte Ergebnis der Addition



Spieltheorie

## Karazuba-Multiplikation



Beobachtung: 
$$(a_0 + a_1) \cdot (b_0 + b_1) = a_0 \cdot b_0 + a_1 \cdot b_1 + a_1 \cdot b_0 + a_0 \cdot b_1$$

#### Algorithm 1 recMult(int a, int b)

```
Require: a und b haben n Ziffern, sei k = \lfloor n/2 \rfloor if n = 1 then return a \cdot b end if schreibe a als a_1 \cdot B^k + a_0 schreibe b als b_1 \cdot B^k + b_0 c_{11} = recMult(a_1, b_1) c_{00} = recMult(a_0, b_0) return c_{11} \cdot B^{2k} + (recMult((a_1 + a_0), (b_1 + b_0)) - c_{11} - c_{00}) \cdot B^k + c_{00}
```



## **Naive Exponentiation**



#### **Algorithm 2** Bereche $y = x^n$ naiv

```
Require: n > 0 \lor x \neq 0
Ensure: v = x^n
  v \leftarrow 1
  if n < 0 then
       X \leftarrow 1/x
       N \leftarrow -n
  else
       X \leftarrow x
       N \leftarrow n
  end if
  while N \neq 0 do
       y \leftarrow y \cdot X
       N \leftarrow N - 1
  end while
  return y
```

Bei ICPC gehen wir davon aus, dass Multiplikation zweier Zahlen in  $\mathcal{O}(1)$  liegt, also naive Exponentiation in  $\mathcal{O}(n)$ 



#### **Idee**



Beobachtung:

$$x^{n} = \begin{cases} (x^{2})^{n/2} & \text{für n gerade} \\ x \cdot (x^{2})^{(n-1)/2} & \text{für n ungerade} \end{cases}$$
 (1)

## **Exponentiation by Squaring, rekursiv**



#### **Algorithm 3** Exponentiation(n, x) (rekursiv)

```
if n < 0 then
return Exponentiation(-n, 1/x)
else if n = 0 then
return 1
else if n = 1 then
return x
else if n modulo 2 = 0 then
return Exponentiation(n/2, x \cdot x)
else
return x \cdot Exponentiation((n-1)/2, x \cdot x)
end if
```

Da Multiplikation konstant viel Zeit benötigt, liegt die Exponentiation in  $\mathcal{O}(log(n))$ 



## Beispiel



```
2^{10638} = (2^{5319})^2
2^{5319} = 2 \cdot (2^{2659})^2
2^{2659} = 2 \cdot (2^{1329})^2
2^{1329} = 2 \cdot (2^{664})^2
2^{664} = (2^{332})^2
2^{332} = (2^{166})^2
2^{166} = (2^{83})^2
2^{83} = 2 \cdot (2^{41})^2
2^{41} = 2 \cdot (2^{20})^2
2^{20} = (2^{10})^2
2^{10} = (2^5)^2
2^5 = 2 \cdot (2^2)^2
2^2 = (2^1)^2
```

## **Exponentiation by Squaring, iterativ**



### Algorithm 4 Exponentiation(n, x) (iterativ)

```
if n < 0 then
   n = -n
   x = 1/x
end if
if n=0 then
   return 1
   v=1
end if
while n > 1 do
   if n \mod 2 = 0 then
      x = x \cdot x
      n = n/2
   else
      y = y \cdot x
      x = x \cdot x
      n = (n-1)/2
   end if
end while
```

21. Juni 2018

#### Kombinatorik



#### Definition

"Combinatorics is a branch of discrete mathematics concerning the study of countable discrete structures"

<sup>a</sup>Competitive Programming 3

#### Bei ICPC-Aufgaben erkennbar an:

- "Wie viele Moeglichkeiten gibt es, ..?"
- "Berechne die Anzahl an X."
- Alles, was mit Zählen zu tun hat



#### Kombinatorik bei ICPC



Die Lösung für eine Kombinatorik-ICPC-Aufgabe ist meist eine kurze rekursive Formel, oft in Verbindung mit Greedy oder DP. Der Aufwand liegt nicht in der Implementierung, sondern im Aufstellen der Formel.

- Kombinatorik-Aufgaben von einer Person bearbeiten lassen
  - bestenfalls mit guten mathematischen Kenntnissen
- Sobald die Formel fertig ist, Lösung coden und abgeben!

#### Kombinatorik bei ICPC



Gängige Formeln sollte man kennen... ... oder ausprobieren!

#### On-Line Encyclopedia of Integer Sequences

Unter http://oeis.org/ kann man die ersten Lösungen für kleine Probleminstanzen eingeben und so prüfen, ob bereits eine Formel für diese Folge existiert.

## Aufgabe - Mauerbau



- Baue eine Mauer aus bestimmten Ziegeln.
- jeder Ziegel ist 2 Einheiten breit und 1 Einheit hoch und kann beliebig gedreht werden.
- jede Mauer ist 2 Einheiten hoch und m Einheiten breit (0 < m <= 50).
- Aufgabe: Wie viele Kombinationen an Ziegelsteinen gibt es?

#### **Fibonacci**



#### Definition:

$$f(0) = 0$$
  
 $f(1) = 1$   
 $n > 1 : f(n) = f(n-1) + f(n-2)$ 

Also: 0, 1, 1, 2, 3, 4, 8, 13, 21, 34, 55, 89...

Sollte man erkennen!



## Fibonacci - Implementierung



- Mit DP in O(n)
- Binet's Formel:

$$f(n) = \frac{(\phi^n - (-\phi)^{-n})}{\sqrt{5}}$$

 $\phi := goldener Schnitt$ 

$$\phi = \frac{(1+\sqrt{5})}{2}$$

 $\phi$  gerundet nutzen. Anzahl der Nachkommastellen entscheidet über Genauigkeit!

- oder vorberechnen!
- Achtung: Wird sehr schnell sehr groß.



## Aufgabe - Lieblingsschokolade



- Gegeben: Paket mit n Schokoladentafeln, alle gleich verpackt
- Davon sind k Tafeln in meiner Lieblingssorte
- Gesucht: Wahrscheinlichkeit, k Tafeln zu nehmen und nur Lieblingsschokolade zu ziehen

#### Binomialkoeffizient



Wie viele Möglichkeiten gibt es, *k* Objekte aus einer Menge von *n* verschiedenen Objekten zu ziehen?

$$C(n,k) = \binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}$$

**Rekursive Definition:** 

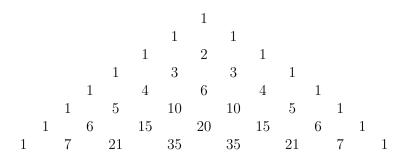
$$C(n,0) = C(n,n) = 1$$
  
 $C(n,k) = C(n-1,k-1) + C(n-1,k)$ 

21. Juni 2018

# Binomialkoeffizient - Visualisierung



Abbildung: Visualisierung Binomialkoeffizient<sup>1</sup>



<sup>1</sup>Quelle: Wikipedia

Big Integer

Kombinatorik

Spieltheorie

Exponentiation by squaring

# Binomialkoeffizient - Implementierung



- Naiv rekursiv
  - → Viel zu langsam!
- Vorberechnen



Charlotte P., Lena W., Vera C., Christian K. - Mathe 1

Spieltheorie

## Binomialkoeffizient DP Bottom-Up



#### **Algorithm 5** Binomialkoeffizient(n, k)

```
C \leftarrow int[n+1][k+1]
i \leftarrow 0
while i \le n do
   i \leftarrow 0
   while j <= min(i, k) do
       if j == 0 | j == i then
           C[i][i] \leftarrow 1
       else
           C[i][j] \leftarrow C[i-1][j-1] + C[i-1][j]
       end if
       i + +
   end while
   i + +
end while
return C[n][k]
```

Laufzeit: O(nk), Platzbedarf: O(nk)



Spieltheorie

## Binomialkoeffizient - Implementierung



- Naiv rekursiv
  - → Viel zu langsam!
- Vorberechnen
  - Meist interessieren nicht alle Werte
  - lacksquare o Top-Down rekursiv mit Zwischenspeichern
- Mit nicht-rekursiver Formel

$$C(n,k) = \frac{n!}{(n-k)! \cdot k!} = C(n,n-k)$$



#### **Binomialkoeffizient Nicht-Rekursiv**



#### **Algorithm 6** Binomialkoeffizient(n, k)

```
if k > n-k then k \leftarrow n-k end if result \leftarrow 1 i \leftarrow 0 while i < k do result \leftarrow result \leftarrow result \leftarrow (n-i) result \leftarrow result \div (i+1) i++ end while return result
```

Laufzeit:  $O(\frac{n}{2})$ , Platzbedarf: O(1)



Charlotte P., Lena W., Vera C., Christian K. - Mathe 1

Spieltheorie

## Aufgabe - Der Mathetest



- Gegeben: Anzahl an Faktoren
- Gesucht: Anzahl an Möglichkeiten, diese korrekt zu klammern
- Beispiel:
  - Gegeben: {a, b, c, d}
  - a(b(cd)), (ab)(cd), ((ab)c)d, (a(bc))d, a((bc)d)
  - Lösung: 5



21. Juni 2018

#### Catalan Nummern



Definition:

$$Cat(n) = \frac{1}{n+1} \binom{2n}{n}$$

Rekursiv:

$$Cat(0) = 1$$

$$Cat(n+1) = \sum_{i=0}^{n} Cat(i) \cdot Cat(n-i)$$

Also: 1, 1, 2, 5, 14, 42, 132, 429, ...



Spieltheorie

#### Catalan Nummern



#### Cat (n) entspricht zum Beispiel:

- Anzahl verschiedener Binär-Bäume mit n Knoten.
- Anzahl korrekter Klammerausdruecke mit *n* Klammerpaaren
- Anzahl verschiedener Möglichkeiten, n + 1 Faktoren korrekt zu klammern
- Anzahl Möglichkeiten, ein konvexes n + 2-Eck in Dreiecke aufzuteilen

21. Juni 2018

# Catalan Nummern - Implementierung



- Naiv rekursiv
  - → Viel zu langsam!
- Rekursiv mit DP

21. Juni 2018

## Catalan Nummern DP Bottom-Up



#### **Algorithm 7** Catalan(n)

```
 \begin{aligned}  & \textit{Cat} \leftarrow \textit{int}[n+1] \\ & \textit{Cat}[0] \leftarrow 1 \\ & \textit{Cat}[1] \leftarrow 1 \\ & \textit{i} \leftarrow 2 \\ & \textit{while } \textit{i} <= \textit{n} \textit{ do} \\ & \textit{Cat}[\textit{i}] \leftarrow 0 \\ & \textit{j} \leftarrow 0 \\ & \textit{while } \textit{j} < \textit{i} \textit{ do} \\ & \textit{Cat}[\textit{i}] \leftarrow \textit{Cat}[\textit{i}] + \textit{Cat}[\textit{j}] \cdot \textit{Cat}[\textit{i} - \textit{j} - 1] \\ & \textit{j} + + \\ & \textit{end while} \\ & \textit{i} + + \\ & \textit{end while} \\ & \textit{return } \textit{Cat}[\textit{n}] \end{aligned}
```

Laufzeit:  $O(n^2)$ , Platzbedarf: O(n)



# Catalan Nummern - Implementierung



- Naiv rekursiv
  - → Viel zu langsam!
- Rekursiv mit DP
- Mit Binomialkoeffizient

#### Catalan Nummern



#### **Algorithm 8** Catalan(n)

 $result = Binomialkoeffizient(2 \cdot n, n)$ return result  $\div$  (n+1)

Laufzeit: O(n), Platzbedarf: O(1)

21. Juni 2018

# Catalan Nummern - Implementierung



- Naiv rekursiv
  - → Viel zu langsam!
- Rekursiv mit DP
- Mit Binomialkoeffizient
- Achtung: Catalan Nummern wachsen sehr schnell!
  - $Cat(10) > 10^5$
  - $Cat(19) > 10^{10}$



# Spieltheorie allgemein



- Formalisierung und Darstellung von Spielen
- Versuch, Spielausgang zu berechnen

#### Dabei muss gelten:

- Summe der Gewinne und Verluste aller Spieler beträgt 0 (Nullsummenspiel)
- Meistens ein Gewinner (+1) und ein Verlierer (-1)
- Spiel ist ohne Zufall
- Alle spielen perfekt



## Beispielspiel



#### simples Beispielspiel

Alice und Bob haben sechs Münzen in der Mitte liegen und nehmen abwechselnd je eine bis drei davon. Wer die letzte Münze nimmt, gewinnt.

Spielbaum benutzen





#### Schritt 1:

- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start



Charlotte P., Lena W., Vera C., Christian K. - Mathe 1



#### Schritt 1:

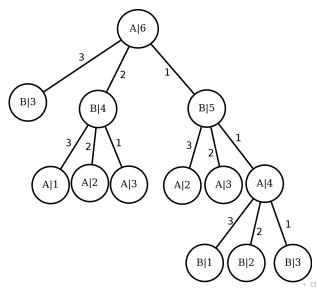
- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start

#### Schritt 2:

- An Blätter des Baumes Ergebnis schreiben
- Von unten nach oben Ergebnis berechnen

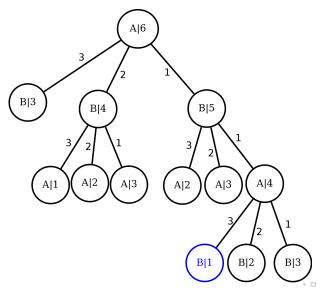


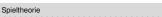




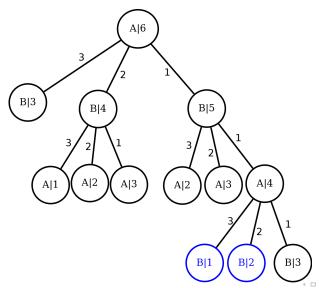






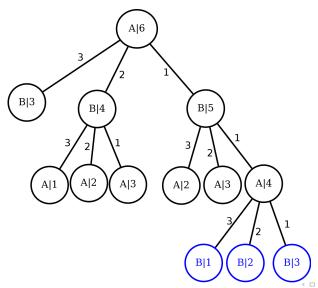










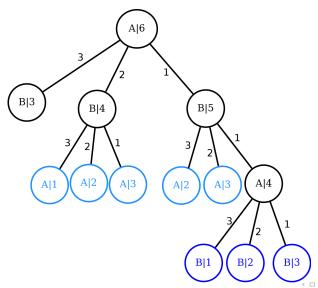




Exponentiation by squaring

Kombinatorik

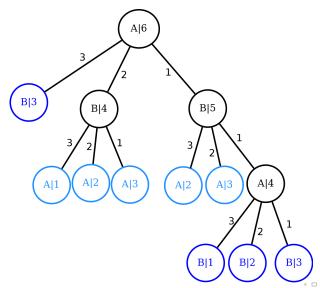






21. Juni 2018

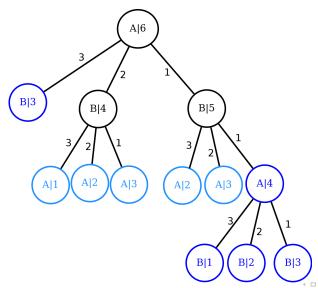






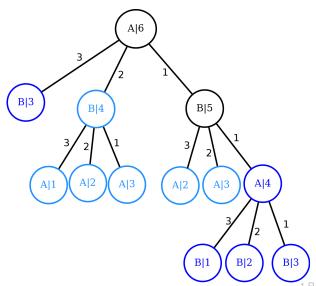
21. Juni 2018





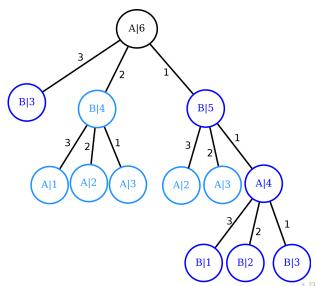






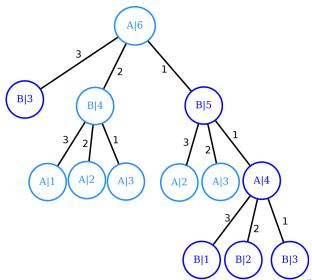














# Min-Max-Strategie

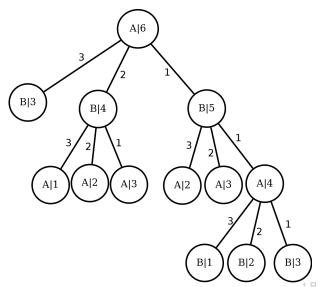


Min-Max-Strategie: Gewinn mit dem größtem Unterschied

$$minmax(s,k) = \begin{cases} k.Bewertung & \text{für k Blatt-Knoten} \\ min \{minmax(k')|k'Kindknoten\} & \text{falls s = A} \\ max \{minmax(k')|k'Kindknoten\} & \text{falls s = B} \end{cases}$$

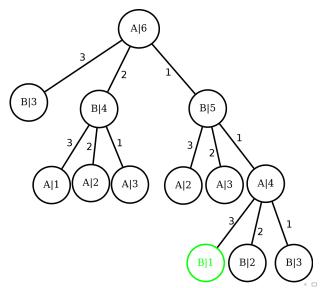
$$(2)$$













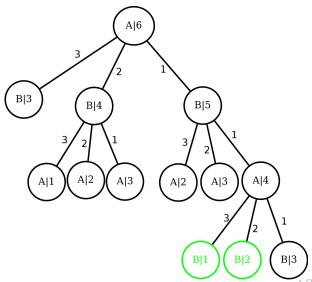
Exponentiation by squaring

Kombinatorik

Spieltheorie

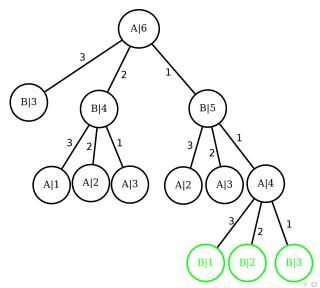
21. Juni 2018









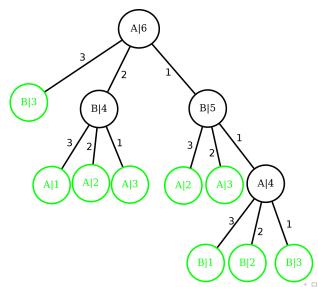




Exponentiation by squaring

Kombinatorik



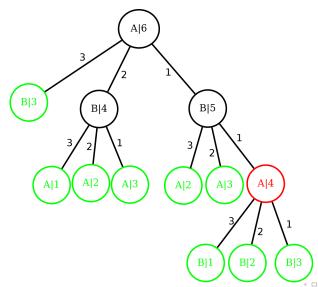




Exponentiation by squaring

Kombinatorik



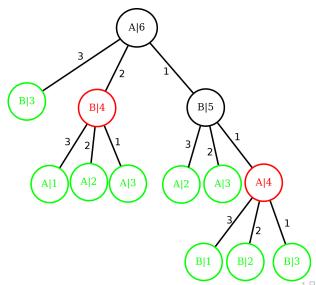




Exponentiation by squaring

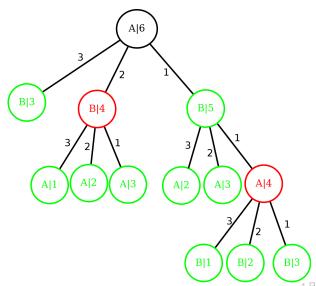
Kombinatorik













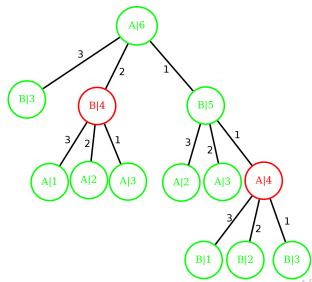
Exponentiation by squaring

Kombinatorik

Spieltheorie

900







# Min-Max-Strategie



Min-Max-Strategie: Gewinn mit dem größtem Unterschied

$$minmax(s,k) = \begin{cases} k.Bewertung & \text{für k Blatt-Knoten} \\ min\{minmax(k')|k'Kindknoten\} & \text{falls s = A} \\ max\{minmax(k')|k'Kindknoten\} & \text{falls s = B} \end{cases}$$

$$(3)$$

Man kann die Spieler-IDs auch weglassen:

$$minmax(k) = \begin{cases} k.Bewertung & \text{für k Blatt-Knoten} \\ -min\{minmax(k')|k'Kindknoten\} & \text{sonst} \end{cases}$$
(4)

## **Implementierung**



Wie gewöhnt als Baum

```
struct Node {
  vector<int> children
  int Bewertung
}
```

Spieler-IDs können weggelassen werden

```
int minmax(Zustands-Knoten k):
   if(k ist Blatt){
     return k.getBewertung
} else {
      for(alle Kindknoten kind von k){
      res = - min(res, Bewertung(k))
    }
}
```

4□ > 4回 > 4 豆 > 4 豆 > 豆 の Q ○

# Nachdenken nicht vergessen



## Beispiel

Die Spieler A und B multiplizieren x abwechselnd mit einer Zahl von 2 bis 9. Am Anfang ist x=1. Wer zuerst über eine Grenze n kommt, gewinnt.

Big Integer



Exponentiation by squaring

# Nachdenken nicht vergessen



#### Beispiel

Die Spieler A und B multiplizieren x abwechselnd mit einer Zahl von 2 bis 9. Am Anfang ist x=1. Wer zuerst über eine Grenze n kommt, gewinnt.

- Problem: Je acht Kindknoten: Baum wird zu groß
- Lösung: Optimale Strategie anhand kleiner Bäume herleiten
- Im Beispiel: A nimmt immer 2, B immer 9 als Faktor



## Nachdenken nicht vergessen



## Beispiel

Die Spieler A und B multiplizieren x abwechselnd mit einer Zahl von 2 bis 9. Am Anfang ist x=1. Wer zuerst über eine Grenze n kommt, gewinnt.

- Problem: Je acht Kindknoten: Baum wird zu groß
- Lösung: Optimale Strategie anhand kleiner Bäume herleiten
- Im Beispiel: A nimmt immer 2, B immer 9 als Faktor

#### Fazit

Falls möglich, anhand kleiner Teilbäume Regel herleiten, statt direkt anzufangen, zu implementieren.



# Nim-Spiel



- Mehrere Haufen mit Objekten
- Zwei Spieler nehmen abwechselnd von einem Haufen
- Wer das letzte Objekt nimmt, gewinnt
- Für wenige Haufen mit Spielbaum modellierbar
- Für viele Haufen eigene Optimalstrategie nötig

21. Juni 2018

# Nim-Spiel: Optimalstrategie



- Nim-Zahl: Anzahl Objekte in Haufen binär mit XOR verknüpfen
- Gewinnstrategie: In jedem Zug die Nim-Zahl auf 0 bringen

#### Beispiel

5 Haufen mit 6, 3, 5, 2 und 7 Elemente

Binär: 110<sub>2</sub>, 011<sub>2</sub>, 101<sub>2</sub>, 010<sub>2</sub> und 111<sub>2</sub> Elemente

Dann: 110<sub>2</sub> XOR 011<sub>2</sub> XOR 101<sub>2</sub> XOR 010<sub>2</sub> XOR 111<sub>2</sub> = 101<sub>2</sub>

Der Spieler am Zug hat also die Möglichkeit, die Nim-Zahl auf 0 zu

bringen (z. B. indem er vom letzten Stapel 5 Elemente entfernt), und hat

somit eine Gewinnstrategie.



# **Grundy-Zahlen**



- Theorem von Sprague-Grundy: Jedes neutrale Spiel äquivalent zu Standard-Nim-Spiel
- Grundy-Zahlen: kleinste Zahl, die nicht Grundy-Zahl von Nachfolgerstellung ist
- Nim-Zahlen entsprechen Grundy-Zahlen
- Gewinnstrategie: Grundy-Zahl in jedem Zug auf 0 bringen



## Beispiel

Big Integer

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

#### Es gibt sechs mögliche Nachfolgerzustände:





#### Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

## Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
  - 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3
  - 1, 0 und 3 Elemente: 001<sub>2</sub> XOR 000<sub>2</sub> XOR 011<sub>2</sub> = 010<sub>2</sub> = 2
  - **1**, 2 und 0 Elemente:  $001_2$  XOR  $010_2$  XOR  $000_2 = 011_2 = 3$
  - **1**, 2 und 1 Elemente:  $001_2$  XOR  $010_2$  XOR  $001_2 = 010_2 = 2$
  - 1, 2 und 2 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 010<sub>2</sub> = 010<sub>2</sub> = 2
  - Kleinste nicht vorkommende Zahl ist 0, also keine Gewinnstrategie





## Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

## Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
- 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3





## Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

#### Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
- 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 0 und 3 Elemente:  $001_2 \text{ XOR } 000_2 \text{ XOR } 011_2 = 010_2 = 2$





## Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

#### Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
- 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 0 und 3 Elemente: 001<sub>2</sub> XOR 000<sub>2</sub> XOR 011<sub>2</sub> = 010<sub>2</sub> = 2
- 1, 2 und 0 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 000<sub>2</sub> = 011<sub>2</sub> = 3
- **1**, 2 und 1 Elemente:  $001_2$  XOR  $010_2$  XOR  $001_2$  =  $010_2$  = 2
- 1, 2 und 2 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 010<sub>2</sub> = 010<sub>2</sub> = 2
- Kleinste nicht vorkommende Zahl ist 0, also keine Gewinnstrategie





## Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

#### Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
- 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 0 und 3 Elemente: 001<sub>2</sub> XOR 000<sub>2</sub> XOR 011<sub>2</sub> = 010<sub>2</sub> = 2
- 1, 2 und 0 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 000<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 2 und 1 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 001<sub>2</sub> = 010<sub>2</sub> = 2





#### Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
  - 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3
  - 1, 0 und 3 Elemente: 001<sub>2</sub> XOR 000<sub>2</sub> XOR 011<sub>2</sub> = 010<sub>2</sub> = 2
- 1, 2 und 0 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 000<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 2 und 1 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 001<sub>2</sub> = 010<sub>2</sub> = 2
- 1, 2 und 2 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 010<sub>2</sub> = 010<sub>2</sub> = 2





## Beispiel

Es gibt drei Haufen mit einem, zwei und drei Elementen. Berechne die Grundy-Zahl dieser Situation

Es gibt sechs mögliche Nachfolgerzustände:

- 0, 2 und 3 Elemente: 000<sub>2</sub> XOR 010<sub>2</sub> XOR 011<sub>2</sub> = 001<sub>2</sub> = 1
- 1, 1 und 3 Elemente: 001<sub>2</sub> XOR 001<sub>2</sub> XOR 011<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 0 und 3 Elemente: 001<sub>2</sub> XOR 000<sub>2</sub> XOR 011<sub>2</sub> = 010<sub>2</sub> = 2
- 1, 2 und 0 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 000<sub>2</sub> = 011<sub>2</sub> = 3
- 1, 2 und 1 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 001<sub>2</sub> = 010<sub>2</sub> = 2
- 1, 2 und 2 Elemente: 001<sub>2</sub> XOR 010<sub>2</sub> XOR 010<sub>2</sub> = 010<sub>2</sub> = 2
- Kleinste nicht vorkommende Zahl ist 0, also keine Gewinnstrategie



## **ICPC-Aufgabe**



#### Schokoladenspiel

Bunty and Dolly are playing a game, described as follows.

There are two boxes full of chocolates. Both can eat 'L' ( $L \ge 1$ ) chocolates from any one box or 'L' chocolates from both the box in each step. They play the game alternatively and the last one to eat the chocolate will be the winner.

As Bunty wants to impress Dolly, he wants to make Dolly the winner. You have to help Bunty in deciding who should play first.

Assume that both the players play the game optimally.

#### Lösungsidee

Nim-Spiel mit zwei Stapeln, also beide Werte binär mit XOR verknüpfen. Bei 0 beginnt Dolly, sonst Bunty.



## **ICPC-Aufgabe**



#### Schokoladenspiel

Bunty and Dolly are playing a game, described as follows.

There are two boxes full of chocolates. Both can eat 'L' ( $L \ge 1$ ) chocolates from any one box or 'L' chocolates from both the box in each step. They play the game alternatively and the last one to eat the chocolate will be the winner.

As Bunty wants to impress Dolly, he wants to make Dolly the winner. You have to help Bunty in deciding who should play first.

Assume that both the players play the game optimally.

## Lösungsidee

Nim-Spiel mit zwei Stapeln, also beide Werte binär mit XOR verknüpfen. Bei 0 beginnt Dolly, sonst Bunty.

