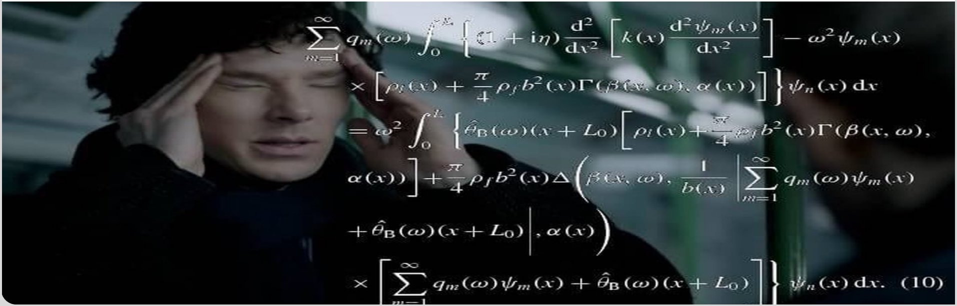
Karlsruher Institut für Technologie

Software Design and Quality

Mathe 1

Charlotte P., Lena W., Vera C., Christian K. | 20. Juni 2018

ITI WAGNER & IPD TICHY



Gliederung

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)
  - ...

Laufzeiten

- Addition, Subtraktion in $\mathcal{O}(n)$
- Multiplikation in $\Theta(n^{\log_2 3})$ (Karatsuba)

C++? Selbst implementieren!

- Addition: Die Tafel ist da $\longrightarrow$
- Multiplikation (z.B. Karazuba-Multiplikation)

Karatsuba-Ofman Multiplikation[1962]

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

Function recMult(a, b)
  assert a und b haben n Ziffern, sei $k = \lceil n/2 \rceil$
  if $n = 1$ **then return** $a \cdot b$
  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
    $c_{11} \cdot B^{2k} +$
    $(\text{recMult}((a_1 + a_0), (b_1 + b_0)) - c_{11} - c_{00})B^k$
    $+ c_{00}$

Naive Exponentiation

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

Algorithm 1 Bereche $y = x^n$ naiv

```
Require: n ≥ 0 ∨ x ≠ 0
Ensure: y = x^n
  y ← 1
  if n < 0 then
      X ← 1/x
      N ← −n
  else
      X ← x
      N ← n
  end if
  while N ≠ 0 do
      if N is even then
          X ← X × X
          N ← N/2
      else {N is odd}
          y ← y × X
          N ← N − 1
      end if
  end while
```

Idee

Beobachtung:

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

Exponentiation by squaring, Laufzeit

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

Algorithm 2 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 · x)
else
    return x · Exponentiation((n − 1)/2, x · x)
end if
```

Algorithm 3 Exponentiation(n, x) (rekursiv)

```
if n < 0 then
    n = −n
    x = 1/x
end if
if n = 0 then
    return 1
    y = 1
end if
while n > 1 do
    if n modulo 2 = 0 then
        x = x · x
        n = n/2
    else
        y = y · x
        x = x · x
        n = (n − 1)/2
    end if
end while
```

Kombinatorik

Definition

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

[a] Competitive Programming 3

Bei ICPC-Aufgaben erkennbar an:

- "Wie viele Moeglichkeiten gibt es, ..?"
- "Berechne die Anzahl an X.."
- Alles, was mit Zaehlen 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

Gänige 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
  $\phi = \frac{(1+\sqrt{5})}{2}$
  ϕ gerundet nutzen. Anzahl der Nachkommastellen entscheidet ber Genauigkeit!
- oder vorberechnen!
- Achtung: Wird sehr schnell sehr gro.

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?

$$C\left(n,k\right) = \binom{n}{k} = \frac{n!}{(n-k)! \times k!}$$

Rekursive Definition:

$$\begin{aligned} C\left(n,0\right) &= C\left(n,n\right) = 1 \\ C\left(n,k\right) &= C\left(n-1,k-1\right) + C\left(n-1,k\right) \end{aligned}$$

Binomialkoeffizient - Visualisierung

Abbildung: Visualisierung Binomialkoeffizient[1]

$$
\begin{array}{ccccccccccccccc}
 & & & & & & & 1 & & & & & & & \\
 & & & & & & 1 & & 1 & & & & & & \\
 & & & & & 1 & & 2 & & 1 & & & & & \\
 & & & & 1 & & 3 & & 3 & & 1 & & & & \\
 & & & 1 & & 4 & & 6 & & 4 & & 1 & & & \\
 & & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & & \\
 & 1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1 & \\
1 & & 7 & & 21 & & 35 & & 35 & & 21 & & 7 & & 1
\end{array}
$$

[1] Quelle: Wikipedia

Binomialkoeffizient - Implementierung

- Naiv rekursiv
  - $\rightarrow$ Viel zu langsam!
- Vorberechnen
  - Meist interessieren nicht alle Werte
  - $\rightarrow$ Top-Down mit Zwischenspeichern
  - Lineare Laufzeit
- Mit nicht-rekursiver Formel
  - Lineare Laufzeit

Algorithm 4 Binomialkoeffizient(n, k)

```
if k > n − k then
    k ← n − k
end if
result ← 1
i ← 0
while i < k do
    result ← result × (n − 1)
    result ← result ÷ (i + 1)
    i + +
end while
return result
```

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

Catalan Nummern

Karlsruher Institut für Technologie

Definition:

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

Rekursiv:

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

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

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

Catalan Nummern - Implementierung

- Naiv rekursiv
  - $\rightarrow$ Viel zu langsam!
- Rekursiv mit DP
  - $\rightarrow$ Immernoch quadratische Laufzeit!
- Mit Binomialkoeffizient
  - $\rightarrow$ Lineare Laufzeit!

Algorithm 5 Catalan(n)

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

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

Erstellen eines Spielbaumes

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

Erstellen eines Spielbaumes

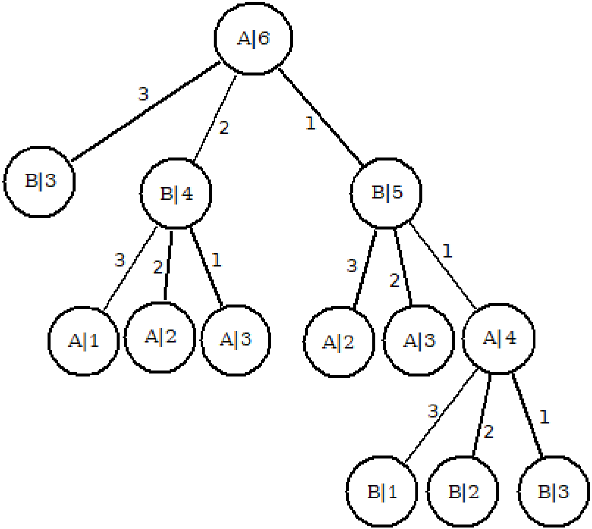
Schritt 1:

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

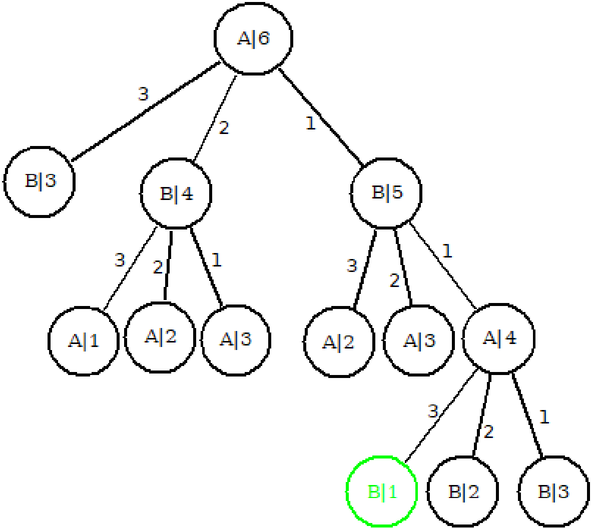
Schritt 2:

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

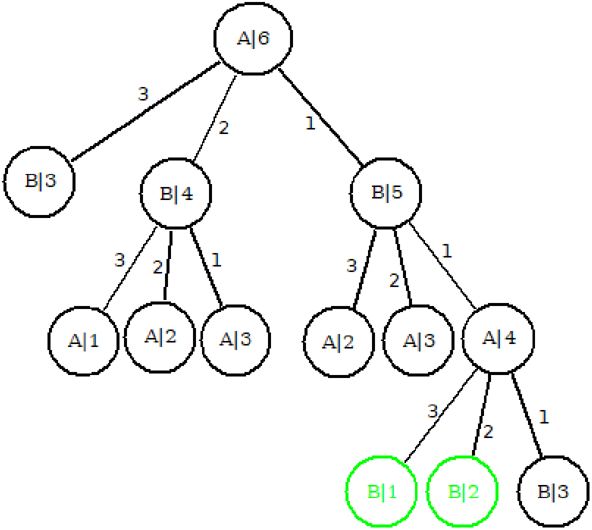
Erstellen eines Spielbaumes



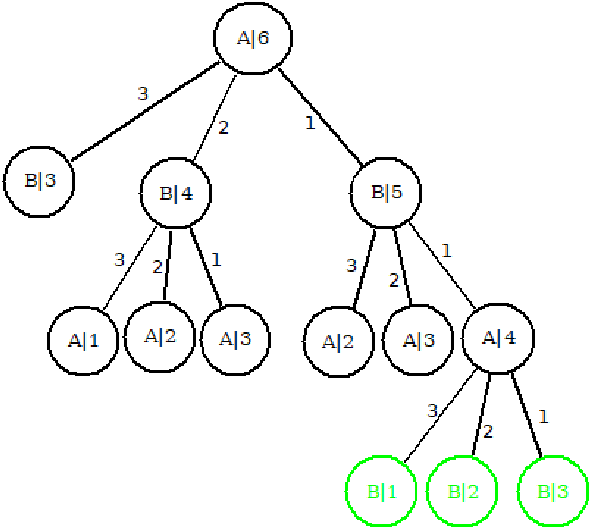
Erstellen eines Spielbaumes



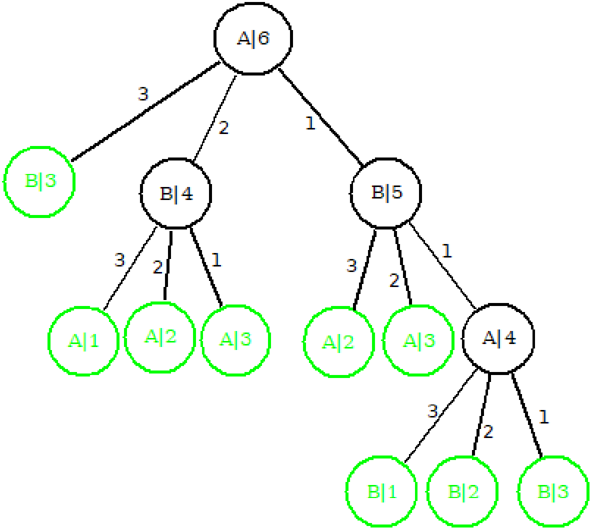
Erstellen eines Spielbaumes



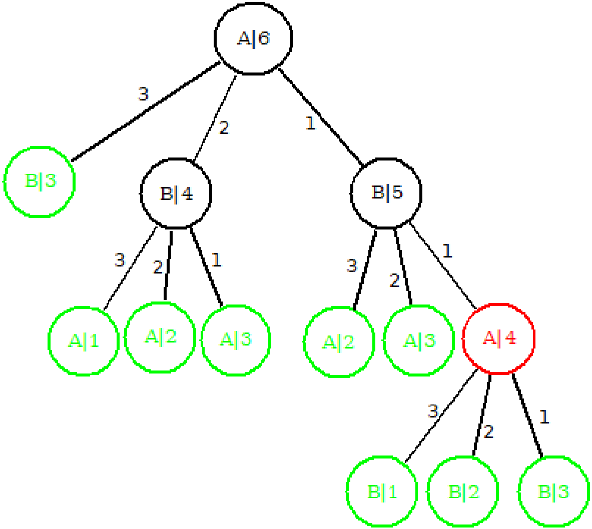
Erstellen eines Spielbaumes



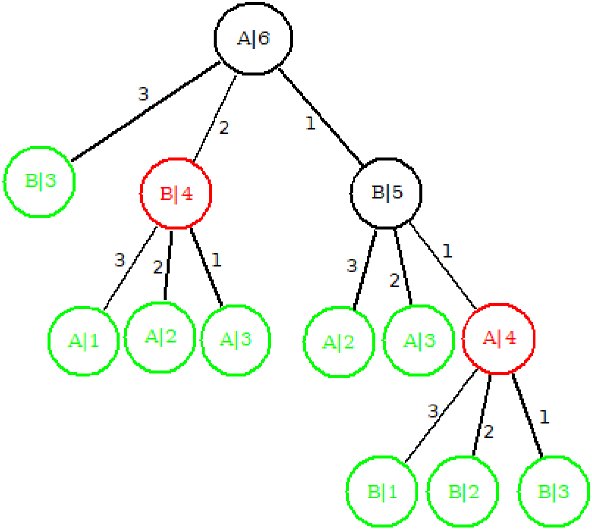
Erstellen eines Spielbaumes



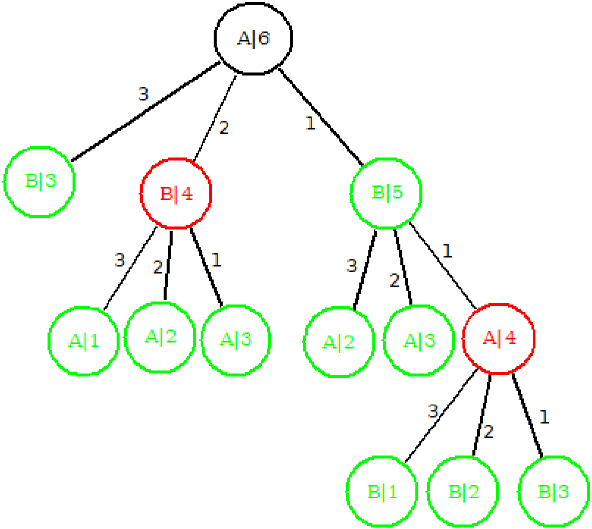
Erstellen eines Spielbaumes



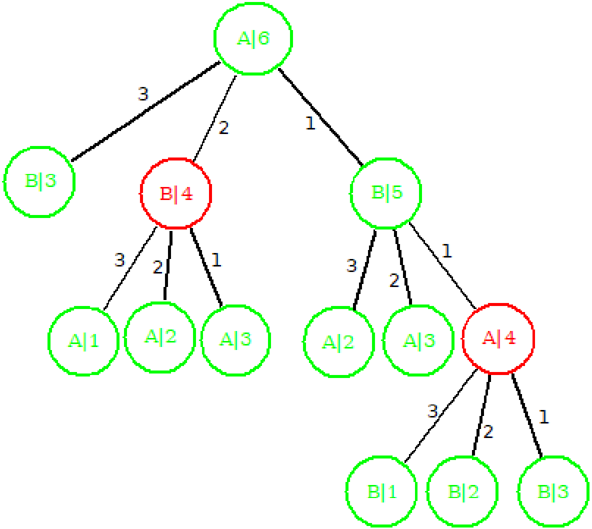
Erstellen eines Spielbaumes



Erstellen eines Spielbaumes



Erstellen eines Spielbaumes

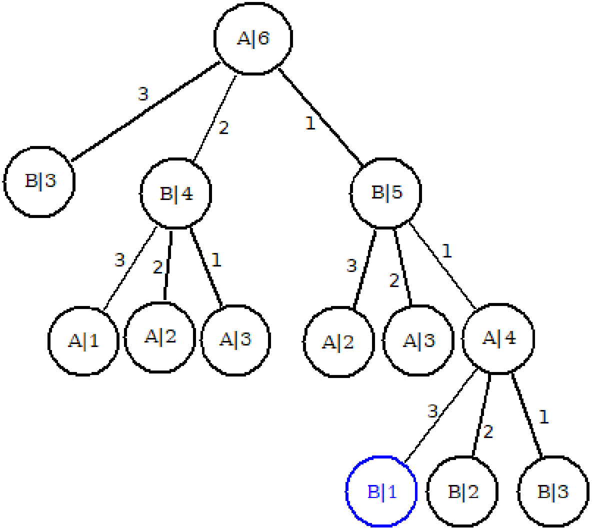


Min-Max-Strategie

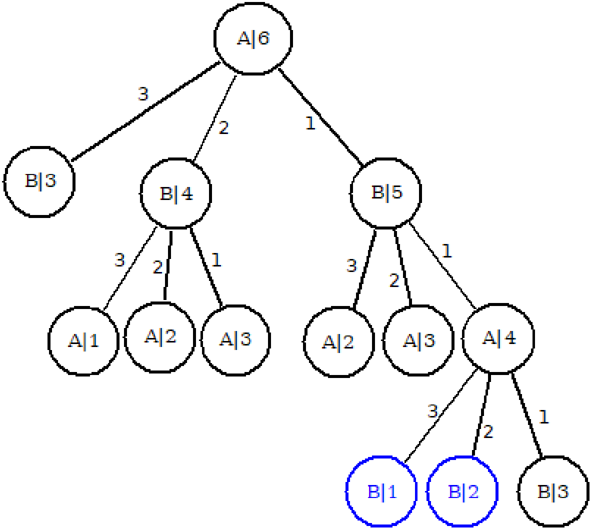
- Min-Max-Strategie: Gewinn mit größtem Unterschied

$$minmax(k) = \begin{cases} k.Bewertung & \text{für k Blatt-Knoten} \\ -min\left\{minmax(k')|k'Kindknoten\right\} & \text{sonst} \end{cases} \tag{2}$$

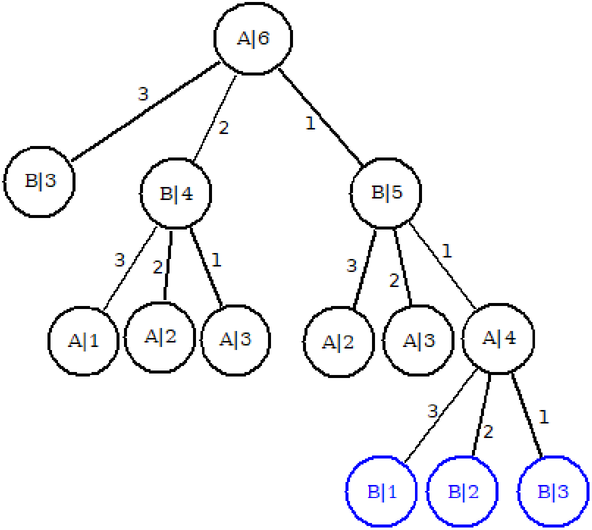
Andere Berechnungsmöglichkeit



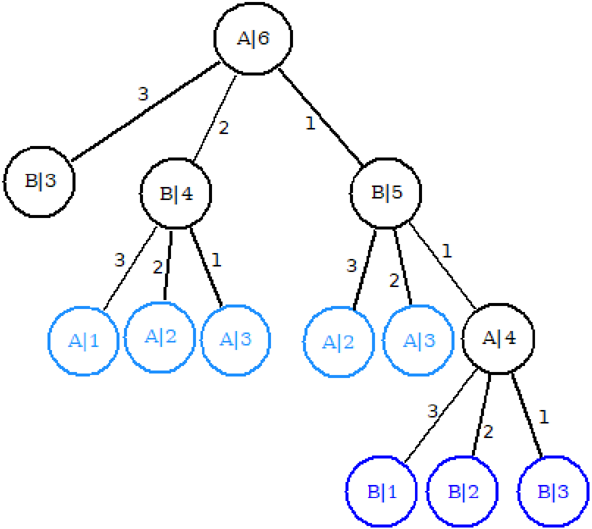
Andere Berechnungsmöglichkeit



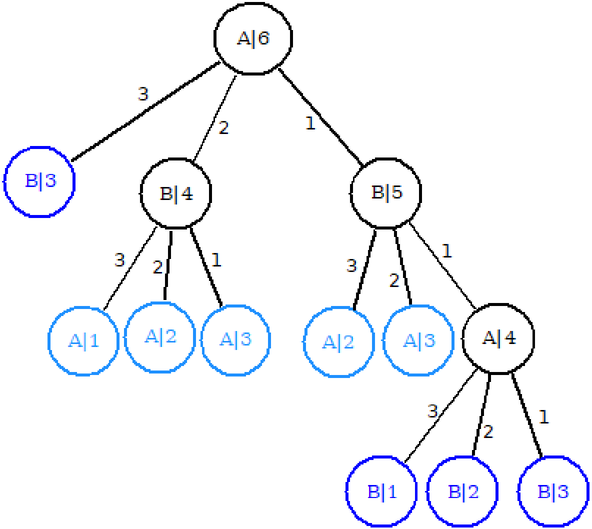
Andere Berechnungsmöglichkeit



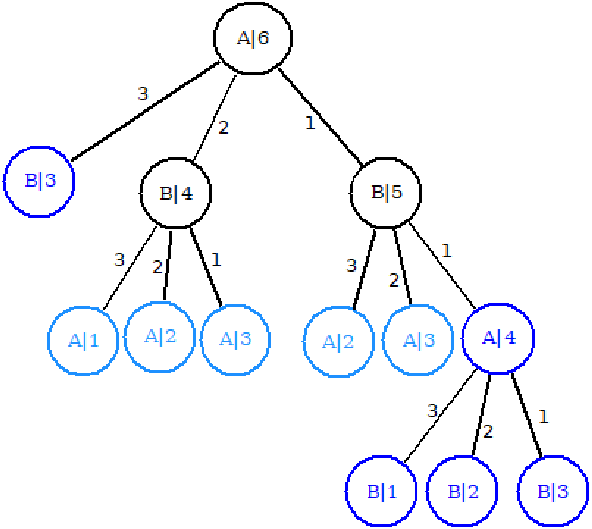
Andere Berechnungsmöglichkeit



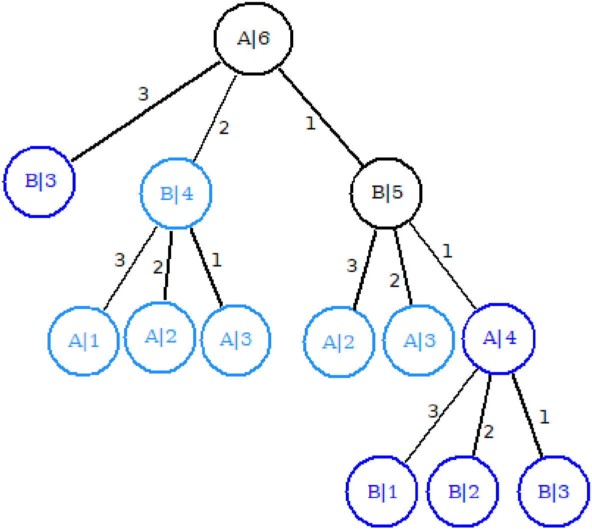
Andere Berechnungsmöglichkeit



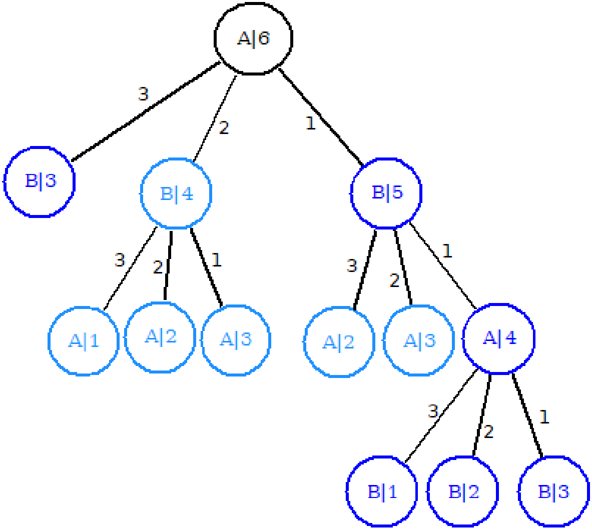
Andere Berechnungsmöglichkeit



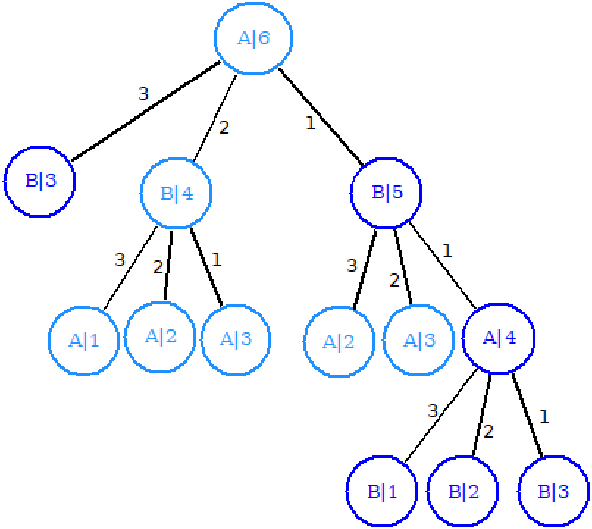
Andere Berechnungsmöglichkeit



Andere Berechnungsmöglichkeit



Andere Berechnungsmöglichkeit



Min-Max-Strategie

- Min-Max-Strategie: Gewinn mit größtem Unterschied

$$minmax(k) = \begin{cases} k.Bewertung & \text{für k Blatt-Knoten} \\ -min\left\{minmax(k')|k'Kindknoten\right\} & \text{sonst} \end{cases} \tag{3}$$

- Andere Möglichkeit der Berechnung:

$$minmax(s, k) = \begin{cases} k.Bewertung & \text{für k Blatt-Knoten} \\ min\left\{minmax(k')|k'Kindknoten\right\} & \text{falls s = A} \\ max\left\{minmax(k')|k'Kindknoten\right\} & \text{falls s = B} \end{cases} \tag{4}$$

Implementierung

- Wie gewohnt 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))
    }
  }
```

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.

Nachdenken nicht vergessen

Karlsruher Institut für Technologie

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.

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 wenigei Haufen mit Spielbaum modellierbar
- Für viele Haufen eigene Optimalstrategie nötig

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_2, 011_2, 101_2, 010_2 und 111_2 Elemente
Dann: 110_2 XOR 011_2 XOR 101_2 XOR 010_2 XOR $111_2 = 101_2$
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

Grundy-Zahlen - Beispiel

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

Grundy-Zahlen - Beispiel

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

Grundy-Zahlen - Beispiel

KIT
Karlsruher Institut für Technologie

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

Grundy-Zahlen - Beispiel

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

Grundy-Zahlen - Beispiel

Karlsruher Institut für Technologie

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

Grundy-Zahlen - Beispiel

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

Grundy-Zahlen - Beispiel

KIT
Karlsruher Institut für Technologie

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

Grundy-Zahlen - Beispiel

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

ICPC-Aufgabe