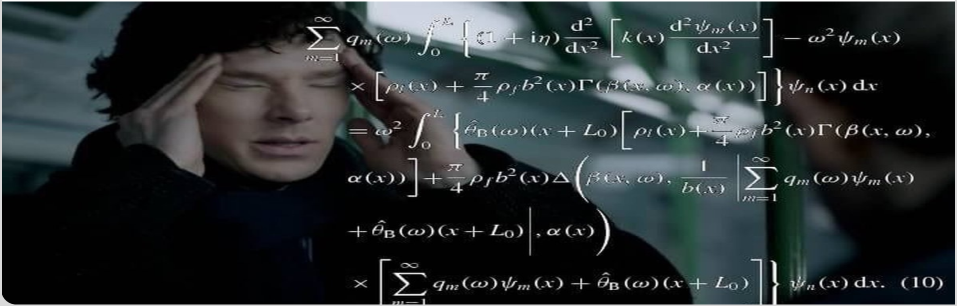
Karlsruher Institut für Technologie

Software Design and Quality

Mathe-1

Charlotte P., Lena W., Vera C., Christian K. | 12. 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

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

Big integer

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

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!

Karlsruher Institut für Technologie

- 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

```
int exp(int x, int n) {
        int result = 1;
        for (int i = 0; i < n; i++) {
                result *= x;
        }
        return result;
}
```

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

Idee

Beobachtung:

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

Exponentiation by squaring, rekursive Implementierung

```
int exponentiationBySquaring(int n, int x) {
        if (n < 0)
                return exponentiationBySquaring(−n, 1/x);
        if (n == 0)
                return 1;
        if (n == 1)
                return x;
        if (n % 2 == 0)
                return exponentiationBySquaring(n/2, x∗x);
        return x∗exponentiationBySquaring((n−1)/2, x∗x);
}
```

Exponentiation by squaring, iterative Implementierung

```
int exponentiationBySquaring(int n, int x) {
        if (n < 0) {
                n = -n;
                x = 1/x;
        }
        if (n == 0)
                return 1;
        int y = 1;
        while (n > 1) {
                if (n % 2 == 0) {
                        x = x * x;
                        n = n/2;
                } else {
                        y = y * x;
                        x = x * x;
                        n = (n - 1) / 2;
                }
        }
        return x*y;
}
```

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

Hier kommt ein kleines Beispiel auf dem Tafel

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

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

Aufgabe - Mauerbau

Karlsruher Institut für Technologie

- Baue eine Mauer aus bestimmten Ziegeln.
- jeder Ziegel ist 2 Einheiten breit und 1 Einheit hoch und kann beliebig gedreht werden.
- jede Mauer is 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 über Genauigkeit!
- oder vorberechnen!
- Achtung: Wird sehr schnell sehr groß.

Der Mathetest

Aufgabe

Lisa macht ein Austauschsemester in Australien. Um fuer einen Mathetest zu lernen, loest sie Rechen-Aufgaben, die ihr eine Kommilitonin diktiert hat. Leider hat die Kommilitonin nicht gesagt, wie die Aufgaben geklammert sind.
Gegeben die Anzahl an Faktoren, wie viele verschiedene Wege gibt es diese zu klammern?

Beispiel:

- Gegeben: $\{a, b, c, d\}$
- Gesucht: Moeglichkeiten fuer Klammerung
- $a\,(b\,(cd))$, $(ab)\,(cd)$, $((ab)\,c)\,d$, $(a\,(bc))\,d$, $a\,((bc)\,d)$

Der Mathetest

Aufgabe

Lisa macht ein Austauschsemester in Australien. Um fuer einen Mathetest zu lernen, loest sie Rechen-Aufgaben, die ihr eine Kommilitonin diktiert hat. Leider hat die Kommilitonin nicht gesagt, wie die Aufgaben geklammert sind.
Gegeben die Anzahl an Faktoren, wie viele verschiedene Wege gibt es diese zu klammern?

Beispiel:

- Gegeben: $\{a, b, c, d\}$
- Gesucht: Moeglichkeiten fuer Klammerung
- $a(b(cd))$, $(ab)(cd)$, $((ab)c)d$, $(a(bc))d$, $a((bc)d)$

Binomialkoeffizient

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

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

Rekursive Definition:

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

Binomialkoeffizient

Karlsruher Institut für Technologie

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

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

Rekursive Definition:

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

Binomialkoeffizient

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

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

Rekursive Definition:

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

Binomialkoeffizient

Tipps:

- Meist interessieren nicht alle Werte von $C\left(n,k\right)$
  - Implementierung deshalb mit top-down
- Fakultaet kann sehr gross werden
  - benutze BigInteger
  - bei grossem k: $C\left(n,k\right) = C\left(n,n-k\right)$

Catalan Numbers

Definition:

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

Catalan Numbers

Cat (n) entspricht zum Beispiel:

- Anzahl verschiedener Binaer-Baeume mit n Knoten
- Anzahl korrekter Klammerausdruecke mit n Klammerpaaren
- Anzahl verschiedener Moeglichkeiten, $n + 1$ Faktoren korrekt zu klammern

Der Mathetest

Karlsruher Institut für Technologie

Aufgabe

Lisa macht ein Austauschsemester in Australien. Um fuer einen Mathetest zu lernen, loest sie Rechen-Aufgaben, die ihr eine Kommilitonin diktiert hat. Leider hat die Kommilitonin nicht gesagt, wie die Aufgaben geklammert sind.
Gegeben die Anzahl an Faktoren, wie viele verschiedene Wege gibt es diese zu klammern?

Loesung:

- Sei n die Anzahl an Faktoren
- $Cat\,(n-1)$ loest die Aufgabe

Der Mathetest

Aufgabe

Lisa macht ein Austauschsemester in Australien. Um fuer einen Mathetest zu lernen, loest sie Rechen-Aufgaben, die ihr eine Kommilitonin diktiert hat. Leider hat die Kommilitonin nicht gesagt, wie die Aufgaben geklammert sind.
Gegeben die Anzahl an Faktoren, wie viele verschiedene Wege gibt es diese zu klammern?

Loesung:

- Sei n die Anzahl an Faktoren
- $Cat(n-1)$ loest die Aufgabe

Zusammenfassung - Kombinatorik bei ICPC

Die Loesung fuer 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, Loesung coden und abgeben!

Zusammenfassung - Kombinatorik bei ICPC

Die Loesung fuer 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, Loesung coden und abgeben!

Zusammenfassung - Kombinatorik bei ICPC

Die Loesung fuer 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, Loesung coden und abgeben!

Zusammenfassung - Kombinatorik bei ICPC

Die Loesung fuer 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, Loesung coden und abgeben!

Zusatz-Tipp!

Gaenige Formeln sollte man kennen...

Zusatz-Tipp!

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

Zusatz-Tipp!

On-Line Encyclopedia of Integer Sequences

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

Spieltheorie allgemein

- Nullsummenspiel
- Genau ein Gewinner
- Alle spielen perfekt
- Gibt es eine Gewinnstrategie?

Spieltheorie allgemein

- Nullsummenspiel
- Genau ein Gewinner
- Alle spielen perfekt
- Gibt es eine Gewinnstrategie?

Spieltheorie allgemein

- Nullsummenspiel
- Genau ein Gewinner
- Alle spielen perfekt
- Gibt es eine Gewinnstrategie?

Spieltheorie allgemein

- Nullsummenspiel
- Genau ein Gewinner
- Alle spielen perfekt
- Gibt es eine Gewinnstrategie?

Beispielspiel

Karlsruher Institut für Technologie

- sechs Münzen, zwei Spieler
- immer abwechselnd maximal drei Münzen nehmen
- Wer die letzte Münze nimmt, gewinnt

Beispielspiel

- sechs Münzen, zwei Spieler
- immer abwechselnd maximal drei Münzen nehmen
- Wer die letzte Münze nimmt, gewinnt

Beispielspiel

- sechs Münzen, zwei Spieler
- immer abwechselnd maximal drei Münzen nehmen
- Wer die letzte Münze nimmt, gewinnt

Spielbaum

- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start
- Blätter: Geben Bewertung (-1 oder 1) an

Spielbaum

- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start
- Blätter: Geben Bewertung (-1 oder 1) an

Spielbaum

- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start
- Blätter: Geben Bewertung (-1 oder 1) an

Spielbaum

Karlsruher Institut für Technologie

- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start
- Blätter: Geben Bewertung (-1 oder 1) an

Spielbaum

- Knoten: aktueller Spieler und Spielsituation
- Kanten: legale Spielzüge
- Wurzel: Spielsituation beim Start
- Blätter: Geben Bewertung (-1 oder 1) an

Min-Max-Strategie

- Min-Max-Strategie: Gewinn mit größtem Unterschied

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

- Spieler-IDs können auch weggelassen werden:

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

Min-Max-Strategie

- Min-Max-Strategie: Gewinn mit größtem Unterschied

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

- Spieler-IDs können auch weggelassen werden:

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

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))
    }
  }
```

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))
    }
  }
```

manchmal Optimierung

- Beispiel: Zahlen abwechselnd mit 2-9 multiplizieren
- erster über n (Grenze) gewinnt
- je acht Kinder: Viel zu viel
- optimal: Immer abwechselnd 9 und 2

manchmal Optimierung

- Beispiel: Zahlen abwechselnd mit 2-9 multiplizieren
- erster über n (Grenze) gewinnt
- je acht Kinder: Viel zu viel
- optimal: Immer abwechselnd 9 und 2

manchmal Optimierung

- Beispiel: Zahlen abwechselnd mit 2-9 multiplizieren
- erster über n (Grenze) gewinnt
- je acht Kinder: Viel zu viel
- optimal: Immer abwechselnd 9 und 2

manchmal Optimierung

- Beispiel: Zahlen abwechselnd mit 2-9 multiplizieren
- erster über n (Grenze) gewinnt
- je acht Kinder: Viel zu viel
- optimal: Immer abwechselnd 9 und 2

Nim-Spiel

- mehrere Haufen mit Objekten
- zwei Spieler nehmen abwechselnd von einem Haufen
- Wer das letzte Objekt nimmt, gewinnt
- Für ein oder zwei Haufen mit Baum möglich

Nim-Spiel

- mehrere Haufen mit Objekten
- zwei Spieler nehmen abwechselnd von einem Haufen
- Wer das letzte Objekt nimmt, gewinnt
- Für ein oder zwei Haufen mit Baum möglich

Nim-Spiel

- mehrere Haufen mit Objekten
- zwei Spieler nehmen abwechselnd von einem Haufen
- Wer das letzte Objekt nimmt, gewinnt
- Für ein oder zwei Haufen mit Baum möglich

Nim-Spiel

- mehrere Haufen mit Objekten
- zwei Spieler nehmen abwechselnd von einem Haufen
- Wer das letzte Objekt nimmt, gewinnt
- Für ein oder zwei Haufen mit Baum möglich

Nim-Spiel: Optimalstrategie

Karlsruher Institut für Technologie

- Anzahl Objekte in Haufen binär mit xor verknüpfen
- Summe auf Null bringen, um zu gewinnen
- Immer möglich

Nim-Spiel: Optimalstrategie

- Anzahl Objekte in Haufen binär mit xor verknüpfen
- Summe auf Null bringen, um zu gewinnen
- Immer möglich

Nim-Spiel: Optimalstrategie

- Anzahl Objekte in Haufen binär mit xor verknüpfen
- Summe auf Null bringen, um zu gewinnen
- Immer möglich

Grundy-Zahlen

Karlsruher Institut für Technologie

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

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
- Gewinnstrategie: Grundy-Zahl möglichst immer auf 0 bringen

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
- Gewinnstrategie: Grundy-Zahl möglichst immer auf 0 bringen