Get Connected to ICPC 2009

UKA ICPC Team

6. Februar 2009

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Überblick

Structure

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

ICPC: Die Veranstalter

Association for Computing Machinery

Advancing Computing as a Science & Profession

- älteste und größte Informatikervereinigung der Welt
- gegründet 1947, Hauptsitz in New York
- etwa 85 500 Mitglieder weltweit, darunter
  - 20 000 Studenten
  - 69% US-Amerikaner
  - 14% Frauen
- teilweise weltbekannte „Special Interest Groups“, z. B. SIGGRAPH
- Stifter des *A. M. Turing Award*, „der Nobelpreis der Informatik“ (Wired Magazine)

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

ICPC: Überblick

- ältester, größter und angesehenster Programmierwettbewerb der Welt
- erstmalig 1989; Vorläufer seit 1970
- knapp 2 000 teilnehmende Universitäten aus über 80 Ländern

Aus dem offiziellen Fact Sheet:

The contest pits teams of three university students against eight or more complex, real-world problems, with a grueling five-hour deadline. Huddled around a single computer, competitors race against the clock in a battle of logic, strategy, and mental endurance.

ICPC: offizielle Wettbewerbe

Zwei oder drei Runden

1. Local Contests
2. Regional Contests, für uns ab jetzt: Northwestern Europe
   - 2004 und 2005: Paris
   - 2006 und 2007: Lissabon
   - 2008 und 2009: Nürnberg
3. World Finals
   - 2008: Banff, Kanada
   - davor: San Antonio, USA; Tokio, Japan;
   - die ersten Teams der Regionals sind qualifiziert für die World Finals

- Team: drei Contestants, ein Coach (und ein Ersatzmann)
- zwei Teams pro Uni garantiert, drei Teams erlaubt

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

ICPC: Ablauf eines Wettbewerbs

- **Dauer:** fünf Stunden
- **Aufgaben:** mindestens acht, bis zu zehn
- keine Kommunikation mit anderen Teams
- Fragen an Jury über das Online-Bewertungssystem (offenes Frage-Antwort-System)
- **Gewinner** ist das Team mit den meisten gelösten Problemen
  - bei Gleichstand: mit weniger Punkten (Zeit)
  - Punktzahl pro gelöster Aufgabe: Minuten vom Beginn bis zur gültigen Einreichung, plus 20 pro ungültiger Einreichung
  - nach zweieinhalb Stunden im dritten Versuch gelöst: $2{,}5 \cdot 60 + 2 \cdot 20 = 190$ Punkte

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

ICPC: weitere Regeln

- zugelassene Programmiersprachen: C, C++, Java
- Rechenzeit, Arbeitsspeicher, Dateigröße des Quellcodes begrenzt
- Klassenbibliotheken der Sprachen dürfen und sollen benutzt werden
- das Problem muss nicht algorithmisch gelöst werden, die richtigen Antworten in einer gespeicherten Tabelle nachschlagen ist erlaubt!

Structure

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Die Aufgaben

- die Aufgabenbeschreibungen sind grundsätzlich auf Englisch
- jede Aufgabenbeschreibungen besteht aus drei Teilen:
  1. längere Beschreibung des Problems
  2. kurze Beschreibung des Ein- und Ausgabeformats
  3. Beispieleingaben zusammen mit den jeweiligen korrekten Ausgaben
- die Beispiele sind **immer** Teil des offiziellen Tests, also unbedingt selber testen
- Testeingaben können völliger Unsinn sein, solange die Problemstellung formal nicht verlassen wird (Rand- und Sonderfälle)
- nur heute: **Aufgaben nach Schwierigkeit sortiert, leichtere zuerst!**

Aufgaben einsenden

Wenn euer Programm fertig ist,

1. testet es mit der Beispieleingabe und
2. reicht den Quelltext beim Bewertungssystem „DOMjudge“ ein.

- DOMjudge kompiliert das Programm und führt es aus
  - auf `stdin` erhält es die Testeingabe im angegebenen Format
  - auf `stdout` muss es die korrekte Antwort im angegebenen Format ausgeben.
- verboten sind Ausgaben auf `stderr` und Beenden mit Exitstatus $\neq 0$
- erlaubt ist, zuerst alles einzulesen und dann zu antworten

DOMjudge

- `http://icpc.ira.uka.de/getconnected/`
- beim ersten Login speichert DOMjudge die IP eures Teams, Einsendungen werden nur von diesem Rechner angenommen
- das Interface ist sehr einfach; vorhandene Funktionen:
  - Lösungen einschicken
  - Punktestand des eigenen und der anderen Teams einsehen
  - Fragen an die Jury stellen
  - ~~Quellcode für Debugzwecke ausdrucken~~
- zusätzlich auf der Webseite:
  - Aufgabenstellungen einsehen

DOMjudge: Ablauf der Bewertung

1. Programm kompilieren
   - Warnungen sind Fehler
   - Systemaufrufe sind Fehler, außer Ein-/Ausgabe der Testdaten
2. Programm testen mit den Beispieleingaben
3. Programm testen mit weiteren Eingaben

Eigenheiten der Bewertung

- Abbruch beim ersten Fehler
- Abbruch, falls das Zeit-, Ausgabe- oder Speicherlimit erreicht wird
- Test: „`diff`“ der Programmausgabe gegen eine Musterlösung

DOMjudge: Bewertungen

Accepted	korrekte Antwort
Presentation Error	richtige Antworten, aber falsch formatiert
Wrong Answer	falsche Antworten
Compile Time Error	ungültiger Quellcode
Runtime Error	Programmabsturz
Program Size Exceeded	zu viel Quellcode
Output Limit Exceeded	zu viel Ausgabe
Invalid Function	Stacküberlauf oder verbotener Systemaufruf
Invalid Submission	Webinterface fehlbedient oder Einsendung disqualifiziert

DOMjudge: weitere Hinweise

- bei echten Wettbewerben sind alle Bewertungen vorläufig („pending“), bis ein Judge sie überprüft hat
- das Zeitlimit gilt pro Aufruf des Programms
- es wird nicht gemessen, um wie viel das Zeitlimit überschritten wird
- „Presentation Error“: Abweichungen im Whitespace oder in der Groß-/Kleinschreibung
- alle anderen Abweichungen: „Wrong Answer“

C

- Compiler: GCC 4.2
- Aufruf:
  `gcc -O2 -g -Wall -lm -o program yourfile.c`
- Nützliche Header: <stdlib.h>, <stdio.h>, <string.h>, <ctype.h>, <math.h>, <limits.h>, <assert.h>

C++

- Compiler: GCC 4.2
- Aufruf:
  `g++ -O2 -g -Wall -lm -o program yourfile.cpp`
- Nützliche Header: <algorithm>, <iomanip>, <iostream>, <iterator>, <limits>, <list>, <map>, <numeric>, <queue>, <stack>, <string>, <valarray>, <vector>, <cstdlib>, <cstdio>, <cstring>, <cmath>, <cassert>
- `using namespace std;` bzw. `std::` nicht vergessen!

Java 6

- Sun Java Development Kit 1.5.0
- Aufruf: `javac -g -Xlint YourClass.java`
- Ausführung: `java -Xmx250m YourClass`
- die Hauptklasse muss `Main` heißen
- Java ist in echten Wettbewerben nur begrenzt hilfreich, da zu Quelltextintensiv

Entwicklungsumgebung

- Linux vs. Windows (vs. MacOS)
  - alles erlaubt
  - empfohlen: Linux
- Eclipse, KDevelop, Visual Studio, ...
  - alles erlaubt
  - empfohlen: normaler Texteditor
  - wir wollen hier keine „Enterprise Solution Developer“ werden, sondern 200-Zeilen-Programme schreiben
- Webbrowser
  - alles erlaubt

Fließkomma

- double vs. float
- printf verwenden!
- schreibt keine eigenen Rundungsfunktionen
- nein, ernsthaft: schreibt keine eigenen Rundungsfunktionen!

Vorlagen und Demoaufgabe

- Homepage

  http://icpc.ira.uka.de/getconnected/

- Aufgabenbeschreibungen als Textdatei
- Quellcodevorlagen in allen zugelassenen Programmiersprachen
- eine gültige Lösung für Aufgabe A zum Ausprobieren des DOMjudge, ohne Strafpunkte zu holen

- Damit bleibt nur noch zu sagen...

Vorlagen und Demoaufgabe

- Homepage

  `http://icpc.ira.uka.de/getconnected/`

- Aufgabenbeschreibungen als Textdatei
- Quellcodevorlagen in allen zugelassenen Programmiersprachen
- eine gültige Lösung für Aufgabe A zum Ausprobieren des DOMjudge, ohne Strafpunkte zu holen

- Damit bleibt nur noch zu sagen...

Viel Spaß!

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Structure

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Contest

(Präsentation auf dieser Folie stehen lassen,
umschalten auf Live-Ranking-Ansicht des Contest.)

Structure

Aufgabe A: Sum

Beschreibung

- Ganzzahlen x_i einlesen, aufsummieren und ausgeben
- es gilt: $x_i \in [-1000,1000], 0 \leq i \leq 1000$

Aufgabe A: Sum

Beschreibung

- Ganzzahlen x_i einlesen, aufsummieren und ausgeben
- es gilt: $x_i \in [-1000,1000], 0 \leq i \leq 1000$

Lösung

- trivial, siehe oben

Aufgabe B: 3n + 1

Beschreibung

- Gegeben: Erzeugungsvorschrift für (endlichen) Folge von Zahlen
- Gesucht: maximale Länge der Folge für Startwerte $x \in [i,j]$

Aufgabe B: 3n + 1

Beschreibung

- Gegeben: Erzeugungsvorschrift für (endlichen) Folge von Zahlen
- Gesucht: maximale Länge der Folge für Startwerte $x \in [i,j]$

Lösung

- für alle Zahlen im Intervall $[i,j]$ die Folge generieren
- Maximum über alle Längen bestimmen
- Achtung: $j < i$ ist möglich

Aufgabe C: Mixing Milk

Beschreibung

- Bauern verkaufen ihre Milch zu verschiedenen Preisen
- kaufe eine gegebene Menge Milch möglichst günstig
- Menge der produzierten Milch pro Bauer ist beschränkt

Aufgabe C: Mixing Milk

Beschreibung

- Bauern verkaufen ihre Milch zu verschiedenen Preisen
- kaufe eine gegebene Menge Milch möglichst günstig
- Menge der produzierten Milch pro Bauer ist beschränkt

Lösung

- Greedy-Strategie: sortiere Daten nach dem Preis
- kaufe so viel Milch wie möglich beim günstigsten Bauern
- wiederhole dies, bis genügen Milch gekauft wurde

Aufgabe D: Ugly Numbers

Beschreibung

- „Ugly Numbers“ haben als einzige Primfaktoren 2, 3 oder 5
- die ersten 11 Ugly Numbers: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15
- Gesucht: die n-te Ugly Number, $n \leq 1000$

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Aufgabe D: Ugly Numbers

Beschreibung

- „Ugly Numbers“ haben als einzige Primfaktoren 2, 3 oder 5
- die ersten 11 Ugly Numbers: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15
- Gesucht: die n-te Ugly Number, $n \leq 1000$

Lösung

- Idee: die ersten 1000 Ugly Numbers vorberechnen und speichern
- teste nacheinander für ganze Zahlen 1, 2, 3, ... ob sie jeweils eine Ugly Number ist:
  - solange die Zahl durch 2, 3 oder 5 teilbar ist, teile sie durch 2, 3, 5
  - falls danach genau 1 übrigbleibt, ist es eine Ugly Number (da keine anderen Teiler außer 2, 3, 5)

Aufgabe E: 2D-Sum

Beschreibung

- Gegeben: rechteckiges Zahlenfeld der Größe $w \cdot h$ und Q Teilrechtecke
- Berechne die Summe der Zahlen in jedem Teilrechteck

Aufgabe E: 2D-Sum

Beschreibung

- Gegeben: rechteckiges Zahlenfeld der Größe $w \cdot h$ und Q Teilrechtecke
- Berechne die Summe der Zahlen in jedem Teilrechteck

Lösung

- Aufsummieren der Zahlen ist in $\mathcal{O}(Q \cdot w \cdot h) \Rightarrow$ zu langsam
- deshalb: dynamische Programmierung

Aufgabe E: 2D-Sum

Beschreibung

- Gegeben: rechteckiges Zahlenfeld der Größe $w \cdot h$ und Q Teilrechtecke
- Berechne die Summe der Zahlen in jedem Teilrechteck

Lösung

- Aufsummieren der Zahlen ist in $\mathcal{O}(Q \cdot w \cdot h) \Rightarrow$ zu langsam
- deshalb: dynamische Programmierung
- $s(x_1, y_1, x_2, y_2) :=$ Summe der Zahlen im Rechteck $(x_1, y_1), (x_2, y_2)$ und $d(x, y) := s(1, 1, x, y)$
- dann gilt (falls $1 < x_1 \leq x_2 \leq w$), $1 < y_1 \leq y_2 \leq h$)):
  $s(x_1, y_1, x_2, y_2) = d(x_2, y_2) - d(x_2, y_1 - 1) - d(x_1 - 1, y_2) + d(x_1 - 1, y_1 - 1)$

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Aufgabe E: 2D-Sum

Beschreibung

- Gegeben: rechteckiges Zahlenfeld der Größe $w \cdot h$ und Q Teilrechtecke
- Berechne die Summe der Zahlen in jedem Teilrechteck

Lösung

- Aufsummieren der Zahlen ist in $\mathcal{O}(Q \cdot w \cdot h) \Rightarrow$ zu langsam
- deshalb: dynamische Programmierung
- $s(x_1, y_1, x_2, y_2) :=$ Summe der Zahlen im Rechteck $(x_1, y_1), (x_2, y_2)$ und $d(x, y) := s(1, 1, x, y)$
- dann gilt (falls $1 < x_1 \leq x_2 \leq w$), $1 < y_1 \leq y_2 \leq h$)):
  $s(x_1, y_1, x_2, y_2) = d(x_2, y_2) - d(x_2, y_1 - 1) - d(x_1 - 1, y_2) + d(x_1 - 1, y_1 - 1)$
- also alle $d(x, y)$ vorberechnen $\Rightarrow$ einzelne Summe in $\mathcal{O}(1)$
- Gesamtlaufzeit in $\mathcal{O}(Q + w \cdot h)$

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Aufgabe F: ID Codes

Beschreibung

- Gegeben: Zeichenkette $s_1 s_2 \ldots s_L$ aus Kleinbuchstaben, $L \leq 50$
- Gesucht: die lexikographisch nächste Zeichenkette mit den gleichen Zeichen

Aufgabe F: ID Codes

Beschreibung

- Gegeben: Zeichenkette $s_1 s_2 \ldots s_L$ aus Kleinbuchstaben, $L \leq 50$
- Gesucht: die lexikographisch nächste Zeichenkette mit den gleichen Zeichen

Lösung

- in C++: Standardfunktion `next_permutation` verwenden

Aufgabe F: ID Codes

Beschreibung

- Gegeben: Zeichenkette $s_1 s_2 \ldots s_L$ aus Kleinbuchstaben, $L \leq 50$
- Gesucht: die lexikographisch nächste Zeichenkette mit den gleichen Zeichen

Lösung

- in C++: Standardfunktion `next_permutation` verwenden
- finde größte Position i mit $s_i < s_{i+1}$ (von hinten durchlaufen)
- falls keine solche Position existiert: fertig
- finde minimales s_j, so dass $j > i$ und $s_j > s_i$
- vertausche s_i und s_j und sortiere Zeichenkette ab Position $i + 1$

Aufgabe F: ID Codes

Lösung

- finde größte Position i mit $s_i < s_{i+1}$ (von hinten durchlaufen)
- finde minimales s_j, so dass $j > i$ und $s_j > s_i$
- vertausche s_i und s_j und sortiere Zeichenkette ab Position $i + 1$

Beispiel: $bcdfea$

Aufgabe F: ID Codes

Lösung

- finde größte Position i mit $s_i < s_{i+1}$ (von hinten durchlaufen)
- finde minimales s_j, so dass $j > i$ und $s_j > s_i$
- vertausche s_i und s_j und sortiere Zeichenkette ab Position $i + 1$

Beispiel: $bcdfea$

- finde Position $i = 3$, da $s_3 = d < f$

Aufgabe F: ID Codes

Lösung

- finde größte Position i mit $s_i < s_{i+1}$ (von hinten durchlaufen)
- finde minimales s_j, so dass $j > i$ und $s_j > s_i$
- vertausche s_i und s_j und sortiere Zeichenkette ab Position $i + 1$

Beispiel: $bcdfea$

- finde Position $i = 3$, da $s_3 = d < f$
- finde minimales Zeichen $s_5 = e$

Aufgabe F: ID Codes

Lösung

- finde größte Position i mit $s_i < s_{i+1}$ (von hinten durchlaufen)
- finde minimales s_j, so dass $j > i$ und $s_j > s_i$
- vertausche s_i und s_j und sortiere Zeichenkette ab Position $i + 1$

Beispiel: $bcdfea$

- finde Position $i = 3$, da $s_3 = d < f$
- finde minimales Zeichen $s_5 = e$
- vertausche d und e, behalte bce und sortiere fda
- Ergebnis: $bceadf$

Aufgabe G: Bridges

Beschreibung

- t Städte sind nacheinander an einem Fluss platziert ($t < 77$) und besitzten Brücken, so dass gilt
  1. von zwei benachbarten Städten hat höchstens eine eine Brücken
  2. die Anzahl der Brücken ist maximal bzgl. Bedingung 1
- Gesucht: Anzahl der Kombinationen

Aufgabe G: Bridges

Beschreibung

- t Städte sind nacheinander an einem Fluss platziert ($t < 77$) und besitzten Brücken, so dass gilt
  1. von zwei benachbarten Städten hat höchstens eine eine Brücken
  2. die Anzahl der Brücken ist maximal bzgl. Bedingung 1
- Gesucht: Anzahl der Kombinationen

Lösung

- Annahme: wir kennen die Anzahl der Möglichkeiten A_n mit Stadt n als letzter Stadt
- schließe davon auf A_{n+2}: $A_{n+2} = A_n + A_{n-1}$
- initialisiere $A_0 = 1$, $A_1 = 1$, berechne alle A_n iterativ bis $n = 76$

Structure

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Einladung zum Praktikum

Lust auf mehr?

- im Sommersemester 2009: ICPC-Praktikum von IPD und ITI
- Teilnehmer sind (fast) alle erfolgreich (Schein oder 3 ECTS)
- möglicherweise Teilnahme am ICPC Northwestern Europe Regional Contest 2009 im November in Nürnberg
- allgemein eine gute Gelegenheit, die eigenen Fähigkeiten in der Algorithmik in Theorie und Praxis zu vertiefen

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Bisherige Erfolge

- regelmäßige Teilnahme am SWERC seit 2003
- bisherige Platzierungen:

Jahr	Ort	Platzierung	Teams
1996	Zürich	13	30
1997	Ulm	7	37
2000	Freiburg[1]	10	24
2003	Paris	8	56
2004	Paris	13	57
2005	Paris	19, 20, 35	46
2006	Lissabon	10, 13, 32	63
2007	Lissabon	2, 12, 22	64
2008	Nürnberg	6, 11, 17	56

[1] MWERC, nicht SWERC

Ablauf des Praktikums

- jede Woche eine Theorie- und Praxissitzung zu übergeordnetem Themengebiet (Anwesenheitspflicht)
- Praxistermin
  - voraussichtlich zwei Termine zu je drei Stunden zur Auswahl
  - Lösen von ICPC-Problemen wie heute, jedoch in Einzelarbeit
- Theorietermin
  - Vorstellung von Musterlösungen für Probleme aus Praxissitzung
  - Vermittlung weiterführender Kenntnisse, die für die Problemlösung hilfreich sind, z. B. in Graphen- und Zahlentheorie
- Teilnehmeraufgabe und -vortrag
  - jeder Teilnehmer soll an einem Vortrag zu einer komplizierteren Aufgabe mitwirken
  - Vorträge werden in kleinen Teams ausgearbeitet

Termine des Praktikums

- Praxistermin
  - mittwochs, 13 Uhr und 16 Uhr
  - ATIS Praktikumspoolraum, Infobau 50.34, Raum -143 (hier)
- Theorietermin
  - donnerstags, 09:45 Uhr
  - Seminarraum 301, Infobau 50.34
- Termine und Räume sind vorläufig und werden in der ersten Besprechung festgelegt
- weitere Informationen für Interessierte auf

`http://icpc.ira.uka.de/`

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825