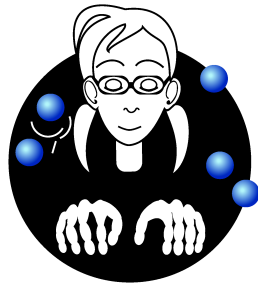
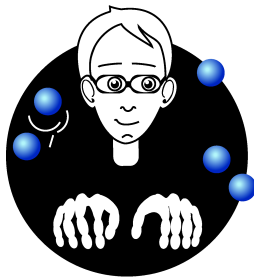


SOI 2010

DIE SCHWEIZER INFORMATIKOLYMPIADE





SOI 2010

Offizielles

Was ist die SOI?

Die Schweizer Informatikolympiade ist die nationale Ausscheidung für die Teilnahme an der IOI (der Internationalen Informatikolympiade). Die SOI trainiert und erkürt vier Teilnehmer, die die Schweiz an der Internationalen Olympiade vertreten werden.

Was ist die IOI?

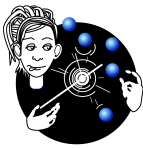
Die Internationale Informatikolympiade ist eine von den Wissenschaftsolympiaden, sozusagen die Weltmeisterschaft im Programmieren. Jedes Jahr treten knapp 300 Teilnehmer aus über 80 verschiedenen Ländern gegeneinander an, um Gold, Silber oder Bronze nach Hause zu tragen.

Weshalb du teilnehmen solltest

- Du lernst alles mögliche über Informatik.
- Du triffst Leute, die ähnliche Interessen haben wie du.
- Du hast die Chance, dein Können auch an der internationalen Olympiade zu zeigen; die Reise wird dir von der SOI bezahlt!
- Du hast die Chance, an drei Lagern teilzunehmen, in denen es nicht nur ums Programmieren geht.

Was hat es mit den verschiedenen Richtungen auf sich?

Die erste Runde ist in zwei Richtungen aufgeteilt: eine *praxisorientierte Richtung*, in welcher du dich vor allem auf das Programmieren konzentrieren kannst, und eine *theoretieorientierte Richtung*, welche mehr auf die mathematischen und wissenschaftlichen Aspekte der Informatik eingeht. Du kannst wählen, wo du mitmischen willst (du darfst natürlich auch an beiden teilnehmen).



Der Weg zur IOI

Die Teilnehmer mit den höchsten Punktzahlen in der praktischen und theoretischen Runde (jeweils gleich viele von beiden) können an der zweiten Runde teilnehmen. Desweiteren qualifizieren sich die besten Teilnehmer der ersten Runde für eine Trainingswoche im Februar 2010 in Davos, wo die Teilnehmer nochmals speziell mit IOI-ähnlichen Aufgaben trainiert werden und spannende Vorträge zu sehen bekommen. Desweiteren triffst du dort andere Teilnehmer und ihr habt allgemein viel Spass.

Runde 2 besteht aus zwei Teilen, einem Online-Contest mit praktischen Aufgaben und einem theoretischen Teil in Zürich. Die besten Teilnehmer der zweiten Runde werden nach Zürich an die Finalrunde eingeladen. Diese besteht aus 4 Programmier-Contests und wird an zwei Wochenenden in Zürich ausgetragen.

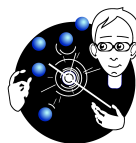
Die vier besten Programmierer der Finalrunde reisen nach *Kanada* an die IOI 2010.

Regeln

- Du kannst nur teilnehmen, wenn du am oder nach dem 1. Juli 1990 geboren wurdest und du von September 2009 bis Dezember 2009 an einer Schule (Gymnasium, Berufsschule, Sekundarschule, etc.) eingeschrieben bist.
- Um teilzunehmen, musst du dich auf www.soi.ch registrieren und als Teilnehmer registrieren.
- Du musst unter deinem eigenen Namen teilnehmen.
- Es ist untersagt, Lösungen oder Quellcode direkt von Büchern oder dem Internet zu kopieren. Du darfst selbstverständlich Bücher oder das Web zu Hilfe nehmen, jedoch musst du deine Lösungen selbst verfassen!

Informationen zur theoretieorientierten Richtung

Die Aufgaben der theoretischen Runde befinden sich in diesem Dokument. Wenn du noch weitere Tipps und Informationen suchst, wie du am besten Lösungen zur theoretischen Runde verfasst, besuche unserer Homepage <http://www.soi.ch/contest/round1>.



Wie sende ich meine Lösung ein?

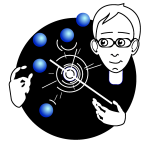
Sobald du auf unserer Homepage <http://www.soi.ch> registriert bist, gibt es vier Möglichkeiten, wie du deine Lösungen einsenden kannst:

1. Per Webpage: lade die Lösungen direkt auf unserer Homepage hoch. Die Seiten für den Upload sind auf der Homepage verlinkt.
2. Per Email an: theorie@soi.ch
Vergewissere dich, dass du nur **ein** Mail sendest, welches deine endgültige Lösung enthält. Bitte schicke die Lösung entweder als PDF, als PostScript-Datei, OpenOffice- or Word-Datei oder als eingescanntes Bild (bitte vernünftige Dateigrösse!).
3. Sende einen Brief an:
Schweizer Informatikolympiade
Theorie A
ETH Zürich
CAB F 14
Universitätstrasse 6
8092 Zürich
4. Per Fax: +41 44 632 13 90.
In der Kopfzeile des Dokuments sollte "SOI 2010" stehen!

Vergiss nicht, den Namen anzugeben, den du bei der Registrierung verwendet hast!

Bis wann sollte ich meine Lösung einsenden?

Der Einsendeschluss der theoretischen Runde ist der **30. November 2009**.



Aufgaben der theoretischen Richtung

1. Unpassende Klammern

Die Schulferien sind zu Ende und Stoff sitzt wieder an seinen Mathe-Hausaufgaben. Er ist schon fast fertig und möchte nun seine Resultate überprüfen. Da die Formeln ziemlich kompliziert und nicht einfach zu verifizieren sind, will Stoff erstmal testen, ob die Klammern wenigstens richtig gesetzt sind. Schreibe ein Programm, welches diesen Job für Stoff übernimmt.

AUFGABE: Dein Programm soll eine Sequenz von Klammern einlesen, also einen Text, der lediglich aus den Zeichen '(' und ')' besteht. Dein Programm soll dann entscheiden, ob die gegebene Klammerung korrekt ist, das heisst sie von einer richtigen Formel stammen kann, von welcher alle Zeichen (ausser den Klammern) gelöscht wurden.

BEISPIEL:

Input:

((()))()

Output:

Sequenz ist korrekt.

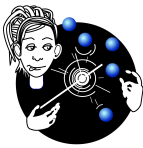
Input:

)(

Output:

Sequenz ist nicht korrekt.

ERKLÄRUNG: Im ersten Beispiel erhalten wir diese Klammerung beispielsweise von folgender Formel: $((1 + 2) * (3 + 4)) / (1 + 1)$. Die zweite Klammerung ist nicht korrekt, da vor der schliessenden Klammer mindestens eine öffnende Klammer stehen muss.



2. Der neue Schal

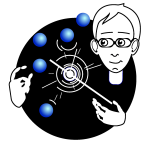
Der Winter rückt näher und die Grossmutter von Maus Stoff macht sich grosse Sorgen um ihn, denn sie meint er ziehe sich einfach nicht warm genug an. Deshalb will seine Grossmutter ihm einen Schal stricken. Stoff wünscht sich einen ganz speziellen Schal: er möchte, dass zwei verschiedene Strickmuster (wir nennen sie 0 und 1) verwendet werden. Jede Reihe des Schals soll aus N Maschen bestehen und jede dieser Maschen ist entweder mit dem Strickmuster 0 oder dem Strickmuster 1 gestrickt. Es gibt somit 2^N verschiedene Möglichkeiten wie die beiden Muster 0 und 1 in einer Reihe angeordnet werden können. Maus Stoff möchte einen Schal, in dem jede der 2^N Reihen genau einmal vorkommt und bei dem sich zwei aufeinanderfolgende Reihen an genau einer Stelle unterscheiden.

Die Grossmutter von Stoff möchte ihm diesen Wunsch erfüllen und hat angefangen zu stricken. Doch schon nach ein paar Reihen ist sie sich nicht mehr sicher, welche Reihen sie nun schon gestrickt hat. Da sie sich nicht gleichzeitig auf das Muster und das Stricken konzentrieren kann, sollst du ihr helfen. Schreibe ein Programm, welches einen Plan für den Schal ausgibt.

AUFGABE: Schreibe ein Programm, welches für ein gegebenes N einen Plan für den Schal ausgibt, welchen die Grossmutter stricken soll. Der Plan besteht aus mehreren 0en und 1en, die in 2^N Reihen aufgeteilt sind. Jede der Reihen hat die Länge N und jede Reihe muss unterschiedlich sein. Zwei aufeinanderfolgende Reihen müssen sich an genau einer Position unterscheiden.

EINGABEFORMAT: Die einzige Eingabe für dein Programm ist N , die Länge einer Reihe des Schals.

AUSGABEFORMAT: Gib genau 2^N Zeilen aus, wobei jede Zeile aus N 0en und 1en besteht und für eine Reihe des Schals, wie oben beschrieben, steht. Falls mehrere Lösungen existieren, kannst du eine beliebige ausgeben.



BEISPIEL:

Input:

3

Output:

100

101

001

011

111

110

010

000

3. Klebeband

Stoffs alter Schulfreund Maus Gyver bastelt enorm gerne. Er ist gerade an einem neuen Bastelprojekt, für welches er eine ganz bestimmte Menge Klebeband benötigt. Wie lange das Klebeband genau sein muss, lässt er durch ein selbst geschriebenes Computerprogramm berechnen. Leider ist Maus Gyver kein all zu guter Programmierer, denn obwohl er absolut sicher ist, dass sein Programm richtig ist, wartet er nun schon mehrere Stunden auf die Ausgabe. Da er nicht mehr länger warten will, fragt er Maus Stoff um Hilfe, damit dieser sein Programm optimiert. Jedoch ist der Code des Programms so unleserlich, dass selbst Stoff nicht weiter kommt.

Alles hängt nun an dir! Hilf Maus Gyver sein Computerprogramm zu optimieren.

AUFGABE: Du kennst das Programm von Maus Gyver (einmal als C-Code und einmal als Pascal-Code; die beiden Programme sind äquivalent). Finde die Ausgabe des Programms und erkläre was das Programm genau berechnet.

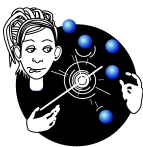
```
#include <stdio.h>
```

```
#define N 314
```

```
#define M 3143149
```

```
int A[N], B[N], c;
```

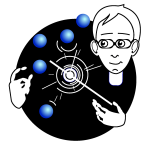
```
void Prepare() {
```



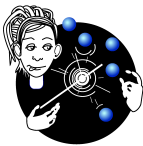
```
int i,j,ok,g;
g = N;
ok = 0;
while(g > 1 || ok == 0) {
    g = (int)((double)g/1.25);
    if (g < 1) g = 1;
    ok = 1;
    i = 0;
    while (i+g < N) {
        if (B[i] > B[i+g]) {
            j=B[i];
            B[i] = B[i+g];
            B[i+g] = j;
            ok = 0;
        }
        ++i;
    }
}

void Check() {
    int i;
    for (i=0; i<N; i++) B[i] = A[i];
    Prepare();
    for (i=1; i<N; i++)
        if (B[i-1] == B[i]) return;
    c++;
    if (c==M) c = 0;
}

void Generate(int i) {
    if (i==N) Check();
    else {
        int j;
        for (j=0; j<N; j++) {
            A[i] = j;
            Generate(i+1);
        }
    }
}
```



```
    }  
}  
  
int main() {  
    c = 0;  
    Generate(0);  
    printf("%d\n", c);  
    return 0;  
}  
  
program Gadget;  
  
const N = 314;  
      M = 3143149;  
  
var A: array[1..N] of integer;  
    B: array[1..N] of integer;  
    c: integer;  
  
procedure Prepare;  
var i,j,g,ok:integer;  
begin  
    g:= N;  
    ok:= 0;  
    while (g > 1) or (ok=0) do  
    begin  
        g:= trunc(real(g)/1.25);  
        if g < 1 then g:=1;  
        ok:=1;  
        i:=1;  
        while i+g <= N do  
        begin  
            if V[i] > V[i+g] then  
            begin  
                j:=V[i];  
                V[i]:=V[i+g];  
                V[i+g]:=j;  
                ok:=0;  
            end  
        end  
    end  
end
```

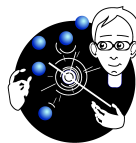


```
        end;  
        i:=i+1;  
    end;  
end;  
end;
```

```
procedure Check;  
var i:integer;  
begin  
    for i:=1 to N do B[i] := A[i];  
    Prepare;  
    for i:=2 to N do  
        if B[i-1] = B[i] then exit;  
    inc(c);  
    if c = M then c := 0;  
end;  
end;
```

```
procedure Generate(i:integer);  
var j:integer;  
begin  
    if i=N+1 then Check()  
    else  
        for j:=1 to N do begin  
            A[i] := j;  
            Generate(i+1);  
        end;  
end;  
end;
```

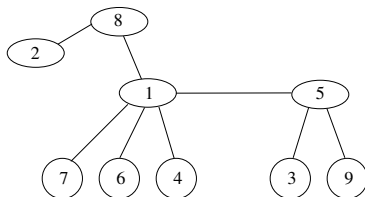
```
begin  
    c := 0;  
    Generate(1);  
    writeln(c);  
end.
```



4. Kastanien

Im Herbst reifen die Früchte von vielen verschiedenen Bäumen und die Leute fangen an sie zu pflücken. Es gibt mehrere Verwendungsgebiete für die Früchte: einige werden in Esswaren umgewandelt, aus anderen werden verschiedene Säfte hergestellt. Nochmals andere werden gebraucht um alltägliche Sachen wie Kleider oder Öl zu machen. Die Welt der Kinder ist da viel einfacher. Auch Kinder Sammeln Früchte doch sie haben nur zwei Sachen im Kopf: essen und spielen. Konsequenterweise essen sie die Früchte, falls sie können. Wenn die Früchte ungenießbar sind werden sie zum spielen gebraucht.

Maus Stoff und seine Schwester Lilly sind selbst noch Kinder. Sie laufen im Wald herum und haben nun schon N Rosskastanien gesammelt. Da Rosskastanien nicht essbar sind, werden sie zum spielen verwendet. Stoff nummeriert die Rosskastanien von 1 bis N , sucht sich ein paar Holzstöckchen und bastelt daraus eine Art Katze mit vielen Beinen:

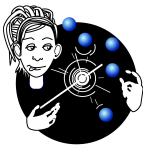


Stoffs mehr-Bein Katze besteht aus folgenden Kastanien:

- vier Kastanien bilden den Kopf, den Hals, das Vorderteil und das Hinterteil der Katze
- mindestens zwei Kastanien repräsentieren die vorderen Pfoten
- mindestens zwei Kastanien repräsentieren die hinteren Pfoten

Einige Paare von Kastanien sind durch Stöcke verbunden: Kopf und Hals, Hals und Vorderteil, Vorder- und Hinterteil. Desweiteren ist jede der vorderen Pfoten mit dem Vorderteil und jede der hinteren Pfoten mit dem Hinterteil verbunden.

Stoff hat nun folgendes Spiel erfunden: Er verwendet alle von seinen N Kastanien (die von 1 bis N nummeriert sind) und baut daraus eine gültige mehr-Bein Katze. Die Katze versteckt er vor Lilly und lässt Lilly die einzelne Körperteile der Katze erraten. Genauer: für jede Zahl von 1 bis N muss Lilly rausfinden, zu welchem Körperteil die Kastanie mit dieser Nummer gehört.



Lilly kann dazu Fragen der Form: “Sind Kastanien x und y durch einen Stock verbunden?” stellen. Wobei x und y die Nummern von zwei verschiedenen Kastanien sind. Da Stoff die ganze Katze kennt, wird er jede Frage wahrheitsgemäss beantworten.

Lilly mag dieses Spiel nicht besonders und möchte, dass du für sie ein Programm entwickelst, welches das Spiel für sie spielt.

AUFGABE:

Hilf Lilly und entwickle eine algorithmische Strategie, wie sie am besten spielt. Mit anderen Worten sollst du einen Algorithmus finden, der zuerst die Zahl N liest und danach durch möglichst wenige Fragen die Struktur der Katze findet.

Bei dieser Aufgabe reicht es, wenn du einen detaillierten Pseudocode¹ und eine Erklärung, warum die Strategie funktioniert, abgibst. Du musst das Programm nicht implementieren, wenn du sicher bist, dass der Pseudocode klar genug ist. Ein weiterer Teil deiner Lösung sollte eine Schätzung der Anzahl Fragen im Worst-Case (abhängig von N) sein. Versuche die Anzahl Fragen zu minimieren.

Für die Schätzung der Laufzeit des Algorithmus kannst du davon ausgehen, dass Stoff die Fragen jeweils umgehend und korrekt beantwortet.

Bemerke, dass eine Strategie, die in N quadratisch viele Fragen stellt, nicht die volle Anzahl Punkte erhält. Mit $O(N^2)$ vielen Fragen kannst du nämlich alle möglichen Paare von Kastanien abfragen, was nicht optimal ist.

BEISPIEL:

$N = 9$

Lilly: Sind Kastanien 2 und 1 verbunden?	Stoff: Ja
Lilly: Sind Kastanien 8 und 1 verbunden?	Stoff: Nein
Lilly: Sind Kastanien 1 und 5 verbunden?	Stoff: Ja
Lilly: Sind Kastanien 7 und 1 verbunden?	Stoff: Ja
Lilly: Sind Kastanien 6 und 1 verbunden?	Stoff: Ja
Lilly: Sind Kastanien 4 und 1 verbunden?	Stoff: Ja
Lilly: Sind Kastanien 3 und 5 verbunden?	Stoff: Ja
Lilly: Sind Kastanien 9 und 5 verbunden?	Stoff: Ja
Lilly: Kopf 2; Hals 8; Vorderteil 1; Hinterteil 5; vordere	Stoff: Ja
Beine 4, 6, 7; hintere Beine 3, 9.	

¹Mit anderen Worten, ein exakte high-level Schritt für Schritt Anleitung deines Algorithmus.



KLARSTELLUNG: Beim bewerten werden wir nicht zwischen Lösungen unterscheiden, die die gleiche asymptotische Anzahl Fragen stellen. Zum Beispiel sind für uns zwei Lösungen mit $47N^2$, resp. $2N^2$ gestellten Fragen gleich gut.² Desweiteren wird die Speicherkomplexität deiner Lösung nicht in die Bewertung einfließen.

5. Bäume

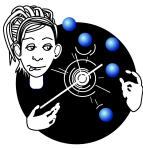
Als Maus Stoffls Uronkel starb, hat Stoff ein grosses Stück Land in Kanada geerbt. Sein Landstück ist unterteilt in $N \times N$ quadratische Zellen, von welchen jede $1m \times 1m$ gross ist.

Als Stoff sein Landstück zum ersten mal sah, war er etwas enttäuscht, denn es war völlig leer. Maus Stoff hat entschieden, dass sich da etwas tun müsse und will nun mehrere Bäume pflanzen. Darum hat er sich extra N Wochen Ferien genommen um jede Woche einen Baum pflanzen zu können. Jedoch fällt es ihm schon nach kurzer Zeit schwer sich zu erinnern, wo er überall schon einen Baum gepflanzt hat und wo noch nicht. Desweiteren ist es möglich, dass einige Bäume im Laufe der Zeit absterben (das Klima in Kanada ist ziemlich rau).

AUFGABE: Schreibe ein Programm, welches sich für Stoff die Positionen merkt, an welchen er schon Bäume gepflanzt hat. Am Anfang liest dein Programm die Zahl N , die Grösse des Landstücks, sowie die Zahl M , die Anzahl Anfragen, die Stoff stellen wird. Es gibt zwei Arten von Anfragen: die erste Art teilt deinem Programm mit, dass ein neuer Baum an einer gewissen Position gepflanzt wurde. Falls bis dahin an besagter Position noch kein Baum gepflanzt war, soll dein Programm mit "Ok" antworten. Andernfalls soll dein Programm Stoff mitteilen, dass er an diesem Ort schon einen Baum gepflanzt hat (somit kann er in dieser Woche etwas anderes unternehmen). Anfragen der zweiten Art teilen deinem Programm die Position eines schon gepflanzten Baumes mit, der abgestorben ist. Du kannst annehmen, dass die Eingabe korrekt ist, das heisst, das an jeder Position eines sterbenden Baumes auch wirklich ein (lebendiger) Baum stand. Es werden genau N Anfragen der ersten Art gestellt.

EINGABEFORMAT: Dein Programm soll zuerst die beiden Zahlen N und M einlesen. Danach soll jede der M Anfragen gelesen werden. Eine Anfrage besteht aus 3 ganzen Zahlen: T , der Art der Anfrage (entweder 1 oder 2, wie

²Falls du wissen möchtest, was das Wort "asymptotisch" genau bedeutet, lies bitte unser How-To: <http://www.soi.ch/de/files/howto.de.pdf>.



oben beschrieben), X , der X -Koordinate und Y , der Y -Koordinate angefragten Zelle. Es gilt immer $1 \leq X, Y \leq N$.

AUSGABEFORMAT: Nach dem Lesen von jeder Anfrage der ersten Art soll dein Programm entweder “Ok” zurückgeben, falls kein (lebender) Baum an der angefragten Position stand, oder “Take a week off”, wenn schon einer dort steht. Dein Programm soll nichts ausgeben, wenn eine Anfrage zweiter Art behandelt wurde.

IMPORTANT NOTES: Du kannst davon ausgehen, dass deinem Programm sehr viel Speicher zur Verfügung steht (viel mehr als N^2). Jedoch ist dieser Speicher nicht vorinitialisiert (du weißt also nicht welcher Wert sich an einer noch nicht geschriebenen Speicherposition befindet). Versuche eine möglichst schnelle Lösung zu finden. Merke auch, dass das Initialisieren von N^2 Speicher $O(N^2)$ Zeit braucht und eine Lösung mit $O(N^2)$ Zeitkomplexität nicht die volle Anzahl Punkte erhalten wird.

BEISPIEL:

Input:

```
4 6
1 1 1
1 3 3
1 1 1
2 1 1
2 3 3
1 1 1
```

Output:

```
Ok
Ok
Take a week off
Ok
```

ERKLÄRUNG: Zu Beginn pflanzt Stoff Bäume an den Positionen $[1, 1]$ und $[3, 3]$. In der dritten Woche hat Stoff eine Woche frei, da schon ein Baum an der Position $[1, 1]$ steht. Danach sterben beide Bäume ab und Stoff pflanzt erneut einen Baum an der Position $[1, 1]$.