D. Laser Strike

Problemname	Laser Strike
Zeitlimit	3 Sekunden
Speicherlimit	1 Gigabyte

Ann und ihre Freundin Kathrin haben vor Kurzem ein neues Brettspiel entdeckt, das zu ihrem Lieblingsspiel geworden ist: Laser Strike. In diesem Spiel arbeiten die beiden Spieler zusammen, um N Spielsteine vom Brett zu entfernen. Das Spiel läuft in zwei Phasen ab. Der Haken ist, dass Kathrin nicht alle Informationen über das Spiel hat. Um das Spiel zu gewinnen, müssen Ann und Kathrin zusammenarbeiten und dabei so wenig wie möglich kommunizieren.

Auf dem Brett befinden sich N einzigartige Spielsteine, nummeriert von 0 bis $N-1$. Beide Spieler können diese Steine sehen. Es gibt auch $N-1$ -Verbindungen zwischen Steinpaaren, sodass es möglich ist, jeden Stein von jedem anderen Stein aus zu erreichen, indem man diesen Verbindungen folgt. Mit anderen Worten: Diese Verbindungen bilden einen Baum. **Nur Ann kann diese Zusammenhänge erkennen, Kathrin kennt sie nicht.**

In der ersten Phase des Spiels legt Ann eine Reihenfolge $\ell_0, \ell_1, \ldots, \ell_{N-2}$ fest, in der Steine entfernt werden sollen, bis nur noch einer übrig ist. Dieser Befehl wird vor Kathrin geheim gehalten. Wenn sie ihn nachahmen kann, gewinnen sie das Spiel. Beim Entfernen von Spielsteinen muss folgende Regel beachtet werden: Jedes Mal, wenn ein Spielstein entfernt wird, muss dieser mit genau einem verbleibenden Spielstein verbunden sein. Mit anderen Worten muss das entfernte Stück ein Blatt des Baums sein, der aus den verbleibenden Stücken und sich selbst gebildet wird. (Nachdem die $N-1$ Steine entfernt wurden, wird der letzte Stein automatisch entfernt und die Spieler gewinnen.) Ann muss eine Reihenfolge wählen, die der obigen Regel entspricht.

Ann wird außerdem eine Nachricht an Kathrin aufschreiben, und zwar in Form einer Binärzeichenfolge. Ann kann wählen, wie lang diese Nachricht ist – aber je kürzer sie ist, desto mehr Punkte bekommt sie.

Danach beginnt die zweite Phase des Spiels. Das Ziel des Spiels besteht für Kathrin darin, $N-1$ Spielsteine in der Reihenfolge $\ell_0, \ell_1, \ldots, \ell_{N-2}$ vom Brett zu entfernen. Sie wird $N-1$ Züge machen. Vor Zug i teilt Ann Kathrin ein Paar ganzer Zahlen a , b mit den folgenden Eigenschaften mit:

- $a < b$;
- es gibt immer noch ein Paar direkt verbundener Steine mit den Nummern a und b; und
- entweder a oder b ist der richtige Stein ℓ_i, das in diesem Zug entfernt werden soll.

Beachten Sie, dass für Ann die Verbindung (a, b) eindeutig durch das Blatt ℓ_i im aktuellen Baum bestimmt wird.

Kathrin entfernt dann entweder a oder b vom Brett. Wenn dies das richtige Stück war, also ℓ_i, spielen sie weiter. Andernfalls verlieren sie das Spiel.

Ihre Aufgabe ist es, die Strategien von Ann und Kathrin so umzusetzen, dass sie das Spiel gewinnen.

Ihr Programm wird je nach Länge der Nachricht gewertet, die Ann in der ersten Phase des Spiels schreibt.

Implementierung

Dies ist ein Multi-Run-Problem, was bedeutet, dass Ihr Programm zweimal ausgeführt wird. Beim ersten Ausführen sollte Anns Strategie für die erste Phase des Spiels umgesetzt werden. Danach soll Kathrins Strategie für die zweite Spielphase umgesetzt werden.

Die erste Zeile der Eingabe enthält zwei Ganzzahlen, P und N, wobei P entweder 1 oder 2 ist (erste oder zweite Phase), und N ist die Anzahl der Teile.

Folgende Eingaben sind abhängig von der Phase:

Phase 1: Ann

Nach der ersten Zeile (oben beschrieben) beschreiben die nächsten $N - 1$ Zeilen der Eingabe den Baum. Jede Zeile enthält zwei Zahlen, a und b ($0 \leq a < b \leq N - 1$), was auf eine Verbindung zwischen den Teilen a und b hinweist.

Ihr Programm sollte mit der Ausgabe einer Binärzeichenfolge mit höchstens $1\,000$ Zeichen beginnen, jeweils `0` oder `1`, der von Ann geschriebenen Nachricht. Beachten Sie, dass zum Generieren einer Zeichenfolge der Länge 0 eine leere Zeile ausgegeben werden sollte.

Danach sollten $N - 1$ Ganzzahlen $\ell_0, \ell_1, \ldots, \ell_{N-2}$ in separaten Zeilen ausgegeben werden, die die Reihenfolge angeben, in der Ann die Blätter des Baums entfernen möchte. Die Reihenfolge muss so sein, dass, wenn die Teile nacheinander in dieser Reihenfolge aus dem Baum entfernt werden, das entfernte Teil immer ein Blatt sein muss, d. h. der Baum muss immer verbunden bleiben.

Phase 2: Kathrin

Nach der ersten Zeile (wie oben beschrieben) enthält die nächste Eingabezeile die Binärzeichenfolge (Anns Nachricht) aus Phase 1.

Danach gibt es $N-1$ Interaktionsrunden, eine für jeden Zug von Kathrin.

Im i -ten Zug Ihr Programm sollte zuerst zwei Zahlen lesen, a und b ($0 \leq a < b \leq N-1$). Eines dieser Stücke ist das Blatt ℓ_i in Anns Reihenfolge, und das andere Stück ist das einzige verbleibende Stück, das mit ℓ_i verbunden ist. Anschließend sollte Ihr Programm ℓ_i ausgeben, was bedeutet, dass Kathrin dieses Blatt entfernt hat. Wenn Ihr Programm nicht das richtige Blatt ℓ_i ausgibt, verlieren die Mädchen das Spiel und Ihr Beitrag wird für diesen Testfall als Falsche Antwort gewertet.

Details

Wenn die *Summe* der Laufzeiten der beiden einzelnen Durchläufe Ihres Programms das Zeitlimit überschreitet, wird Ihre Einreichung als „Zeitlimit überschritten“ gewertet.

Stellen Sie sicher, dass die Standardausgabe nach dem Drucken jeder Zeile geleert wird, da Ihr Programm sonst möglicherweise als „Zeitlimit überschritten“ eingestuft wird. In Python geschieht dies automatisch, solange Sie `input()` zum Lesen von Zeilen verwenden. In C++: `cout << endl;` leert den Puffer durch Drucken einer neuen Zeile; bei Verwendung von printf verwenden Sie `fflush(stdout);`.

Beachten Sie, dass das korrekte Lesen einer leeren Zeichenfolge schwierig sein kann. Die bereitgestellten Vorlagen behandeln diesen Fall korrekt.

Einschränkungen und Bewertung

- $N = 1\,000$.
- $0 \leq a < b \leq N-1$ für alle Verbindungen.

Ihre Lösung wird in einer Reihe von Testgruppen getestet, die jeweils eine bestimmte Anzahl von Punkten wert sind. Jede Testgruppe enthält eine Reihe von Testfällen. Um die Punkte für eine Testgruppe zu erhalten, müssen Sie alle Testfälle in der Testgruppe lösen.

Gruppe	Maximale Punktzahl	Einschränkungen
1	8	Der Baum ist ein Stern. Das heißt, alle Knoten bis auf einen sind Blätter.
2	9	Der Baum ist eine Linie. Das heißt, alle Knoten außer zwei Blattknoten haben genau zwei benachbarte Knoten.
3	21	Der Baum ist ein Stern, von dem Linien ausgehen. Das heißt, alle Knoten haben entweder einen oder zwei benachbarte Knoten, mit Ausnahme eines Knotens mit mehr als zwei benachbarten Knoten.
4	36	Der Abstand zwischen zwei beliebigen Knoten beträgt höchstens 10 .
5	26	Keine weiteren Einschränkungen.

Für jede Testgruppe, die Ihr Programm richtig löst, erhalten Sie eine Punktzahl basierend auf der folgenden Formel:

$$\text{score} = S_g \cdot (1 - 0.3 \cdot \log_{10} \max(K, 1)) \, ,$$

wobei S_g die maximale Punktzahl für die Testgruppe ist, und K ist die maximale Länge von Anns Nachricht, die benötigt wird für jeden Testfall in der Testgruppe. **Ihre Punktzahl für jede Testgruppe wird auf die nächste Ganzzahl gerundet.**

Die folgende Tabelle zeigt für einige Werte von K die Anzahl der Punkte, die Ihr Programm erhält, wenn es alle Testgruppen mit diesem K löst. Um eine Punktzahl von 100 Punkten zu erreichen, muss Ihre Lösung insbesondere jeden Testfall mit $K \leq 1$ lösen.

K	1	5	10	50	100	500	1000
Punktzahl	100	79	70	49	39	20	11

Testwerkzeug

Um das Testen Ihrer Lösung zu erleichtern, haben wir ein einfaches Tool bereitgestellt, das Sie herunterladen können. Siehe „Anhänge“ unten auf der Kattis-Problemseite. Die Nutzung des Tools ist optional. Beachten Sie, dass sich das offizielle Bewertungsprogramm auf Kattis vom Testtool unterscheidet.

Um das Tool zu verwenden, erstellen Sie eine Eingabedatei, z. B. „sample1.in“, die mit einer Zahl N beginnen sollte, gefolgt von $N-1$ Zeilen, die den Baum beschreiben, im gleichen Format wie in Phase 1. Beispielsweise für das folgende Beispiel:

```
7
0 1
1 2
2 3
0 4
0 6
1 5
```

Sagen Sie für Python-Programme `solution.py` (normalerweise ausgeführt als `pypy3 solution.py`) und führen Sie Folgendes aus:

```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

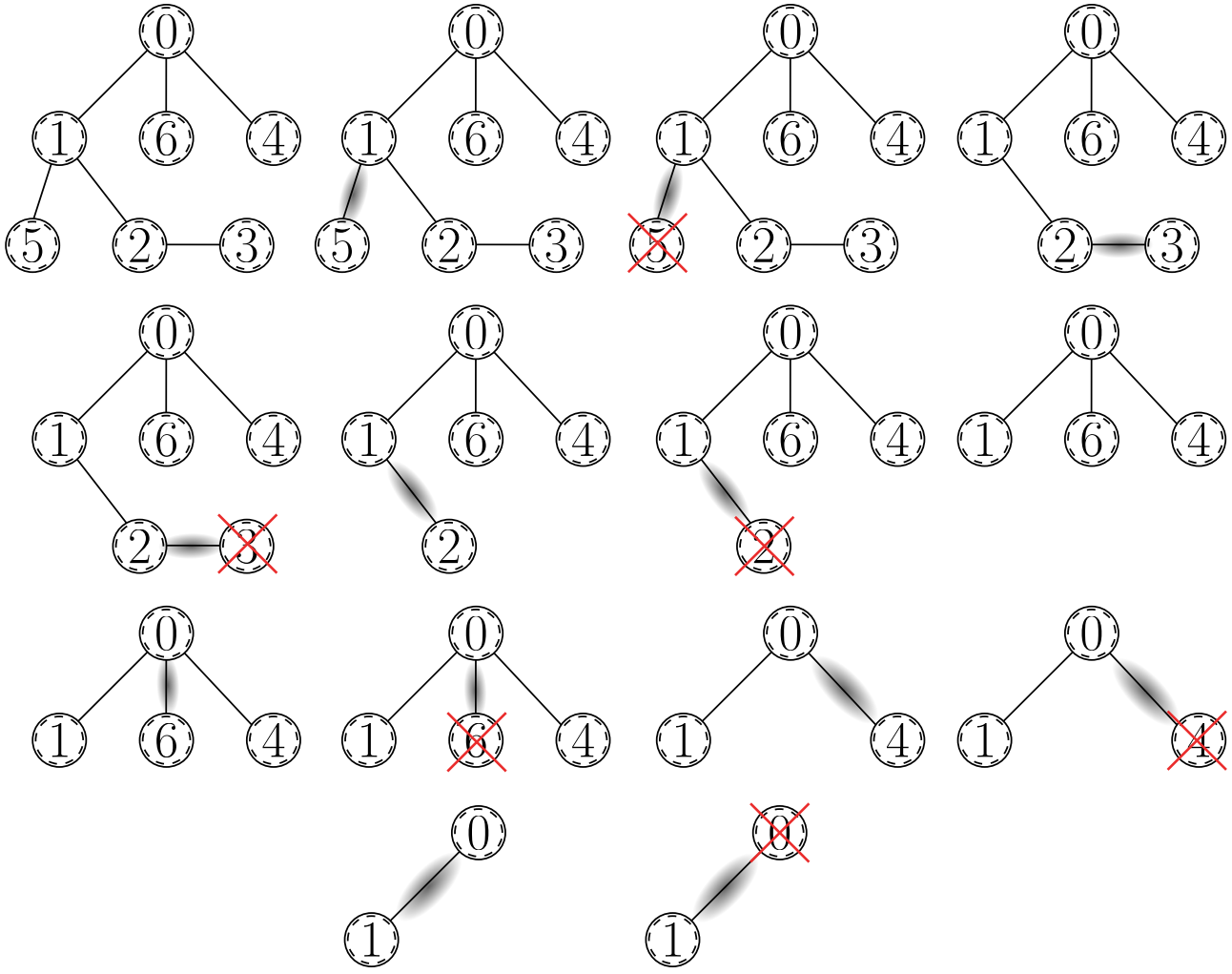
Für C++-Programme kompilieren Sie es zuerst (z. B. mit `g++ -g -O2 -std=gnu++23 -static solution.cpp -o solution.out` “) und führen Sie dann aus:

```
python3 testing_tool.py ./solution.out < sample1.in
```

Beispiel

Beachten Sie, dass das Beispiel in diesem Abschnitt der Einfachheit $N = 7$ hat und daher kein gültiger Testfall ist. Ihr Programm wird diesen Fall voraussichtlich nicht lösen können. Alle Testfälle im Grader haben $N = 1\,000$.

Im Beispiel erhält Ann den folgenden Baum. In der ersten Phase liest Ann den Baum, wählt den Binärstring " `0110` " aus, der an Kathrin gesendet werden soll, und wählt außerdem eine Reihenfolge $[\ell_0, \ell_1, \ldots \ell_{N-2}] = [5, 3, 2, 6, 4, 0]$ in der die Teile aus dem Baum entfernt werden sollen. In der zweiten Phase empfängt Kathrin die Zeichenfolge „ `0110` “, die in der ersten Phase gesendet wurde. Sie erhält dann das Paar $(1, 5)$ und beschließt, den Scheitelpunkt 5 zu entfernen, der tatsächlich das Blatt ist. Für den nächsten Zug erhält sie das Paar $(2, 3)$ und entfernt das Blatt 3 usw. Die folgenden Bilder veranschaulichen die Wechselwirkungen:


0 1 6 4 5 2 3
0 1 6 4 5 2 3
0 1 6 4 5 2 3
0 1 6 4 2 3
0 1 6 4 2 3
0 1 6 4 2
0 1 6 4 2
0 1 6 4
0 1 6 4
0 1 4
0 1
0 1

Grader-Ausgabe	Ihre Ausgabe
1 7	
0 1	
1 2	
2 3	
0 4	
0 6	
1 5	
	0110
	5
	3
	2
	6
	4
	0

Grader-Ausgabe	Ihre Ausgabe
2 7	
0110	
1 5	
	5
2 3	
	3
1 2	
	2
0 6	
	6
0 4	
	4
0 1	
	0