Startaufgabe

Bei dieser Aufgabe wirst du unser Abgabesystem sowie den Aufbau von Aufgaben kennenlernen.

Rechts von diesem Text siehst du das Code-Panel, in das du deine Lösung schreibst und diese auch abschicken kannst. Wähle zuerst unten im Dropdown die Programmiersprache mit der du arbeiten möchtest aus.

Im Code-Panel kannst du deine Lösung schreiben und mit dem Knopf „Abschicken“ zu unserem Server hochladen. Dieser testet deine Abgabe dann automatisiert gegen Testfälle (jeweils mit einem Speicher- und Zeitlimit). Wenn alle Testfälle in einer Teilaufgabe in einer Abgabe gelöst sind, erhälst du die dazugehörigen Punkte.

Die Anzahl Einsendungen, bis du die Punkte erhälst sind dabei egal (aber bitte spamme nicht unser Abgabesystem). Die Teilaufgaben sind bezüglich den Punkten nicht aufbauend. Das heißt, selbst wenn du Teilaufgabe 1 nicht löst, kannst du immer noch Punkte für Teilaufgabe 2 erhalten.

Teilaufgabe 1 (5 Punkte)

Bei dieser Teilaufgabe sollst du die Summe zweier Zahlen berechnen. Gegeben ist eine Funktion `subtask1` mit den Integer-Parametern a und b.

Vervollständige diese Funktion, so dass sie die Summe von `a` und `b` zurückgibt. Mit einem Klick auf ”Abschicken” kannst du diese Lösung einreichen. Das kannst du beliebig oft machen. Wenn die Lösung richtig ist, erhältst du 5 Punkte.

Du kannst annehmen, dass $0 \leq a, b \leq 10^6$ gilt. Solche Einschränkungen helfen dir bei der Implementierung. Hier garantiert sie z.B., dass das Ergebnis in einen 32-Bit Datentypen passt.

Beispiel

Du kannst deine Lösung selber testen, indem du links-unten im Editor den „Test Modus“ aktivierst. Dort kannst du eine Eingabe für das Programm wie folgt spezifizieren:

Eingabe	Ausgabe	Anmerkungen
`1` `5 6`	`11`	Die `1` steht für Teilaufgabe *1*, die Zahlen `5` und `6` sind die Summanden, die zusammen addiert `11` ergeben.

Teilaufgabe 2 (5 Punkte)

Gib die Summe aller Ganzzahlen zwischen 1 und n (inklusive) zurück.

Es gilt: $1 \leq n \leq 30000$. Auch hier hilft dir diese Einschränkung. Da n relativ klein ist, genügt es die n Summanden einzeln aufzusummieren. Formal würde man sagen, dass eine Lösung mit *linearer Laufzeit* ins Zeitlimit passt.

Bonus: Wenn $n \leq 10^9$ wäre, wäre eine lineare Lösung nicht effizient genug. Wie könnte man

in diesem Fall die Summe berechnen?

Beispiel

Eingabe	Ausgabe	Anmerkungen
2 6	21	Die 2 steht für Teilaufgabe 2, die Summe der Zahlen von 1 bis 6 ist 1+2+3+4+5+6 = 21.

Teilaufgabe 3 (5 Punkte)

Gegeben ist eine *aufsteigend sortierte* Liste v mit n Ganzzahlen $v_0, v_1, \dots v_{n-1}$. Deine Aufgabe ist es, für eine Ganzzahl x (ebenfalls zwischen 0 und 10^6) zu beantworten, ob diese Zahl in v enthalten ist.

Wenn x in v enthalten ist, gib `true` zurück, ansonsten gib `false` zurück.

Die Funktion `subtask3` wird q mal mit verschiedenen x aufgerufen. Nur wenn der Rückgabewert für alle q Abfragen stimmt, erhältst du die Punkte für diese Teilaufgabe.

Es gilt:

- $1 \leq n \leq 10^5$
- $0 \leq v_i, x \leq 10^6$
- $v_i < v_{i+1}$
- $1 \leq q \leq 50000$

Auch hier können wir von den Einschränkungen Informationen für unseren Algorithmus ableiten. Eine naheliegende Lösung wäre, mit einer Schleife über die Elemente von v zu prüfen, ob eines davon x entspricht. Das würde etwa n Vergleiche benötigen. Da die Funktion q mal aufgerufen wird, sind das ingesamt nq Vergleiche. Bei den obigen Einschränkungen, wären das ca. $5 \cdot 10^9$, das nichts in Zeitlimit dieser Aufgabe passt. Um eine effizientere Lösung zu entwickeln, müssen wir die Eigenschaft nutzen, dass die Zahlen in v *sortiert* sind.

Der *Binary Search*-Algorithmus nutzt genau diese Eigenschaft aus, um die Laufzeit des Algorithmus zu reduzieren. Lese diesen Artikel durch, und implementiere ein binary search für diese Teilaufgabe. Wie in dieser Aufgabe ersichtlich ist, ist es bei der Informatikolympiade häufig viel mehr relevant, welcher Algorithmus gewählt wird, als wie effizient dein Code implementiert ist.

Beispiel

Eingabe	Ausgabe	Anmerkungen

`3`
`8 2`
`1 4 7 8 12 14 16`
`18`
`16 5`
	`true`
	`false`
		Die `3` steht für Teilaufgabe 3, in der nächsten Zeile stehen n und q. Dann kommen die n Zahlen und q Abfragen, jeweils in einer Zeile.

Nächste Schritte

Jetzt bist du bereit, die nächsten Aufgaben zu probieren. Die besten Teilnehmer:innen qualifizieren sich für das 1. Trainingscamp 2023. Es ist für das Qualifizieren nicht notwendig, jede Aufgabe auf 100 Punkte zu lösen, es lohnt sich daher, sich auf die einfacheren Subtasks zu konzentrieren. Um dir zu helfen, haben wir für jede Teilaufgabe ein „Spicyness“ Rating erstellt. An dem Rating (nur in der sidepanel Ansicht angezeigt, weil emojis sind in LaTeXschwierig) erkennst du, wie schwierig wir die Teilaufgabe einschätzen.

Viel Glück
Das Informatikolympiade Team