35. Bundeswettbewerb Informatik 2016/2017

Die Aufgaben der zweiten Runde

Allgemeine Hinweise

Herzlichen Glückwunsch zum Erreichen der zweiten Runde! Hier sind die Aufgaben. Sie sind anspruchsvoll, und ihre Bearbeitung ist aufwändig. Aber die Mühe lohnt sich, denn durch Teilnahme an der zweiten Runde

- wirst du sicher sehr viel lernen;
- kannst du dich für die Endrunde qualifizieren;
- kannst du einen Buchpreis der Verlage O'Reilly oder dpunkt.verlag gewinnen;
- hast du am Ende eine Arbeit fertig gestellt, die du als Besondere Lernleistung in die Abiturwertung einbringen kannst;
- kannst du dich (als jüngerer Teilnehmer) um die Teilnahme an einer Deutschen Schülerakademie bewerben;
- hast du die Chance auf eine Einladung zu den „Forschungstagen Informatik 2017“ des Max-Planck-Instituts für Informatik in Saarbrücken.

Wir wünschen also viel Spaß und viel Erfolg bei der Bearbeitung!

Es gibt drei Aufgaben. **Eine Einsendung darf Bearbeitungen zu höchstens zwei Aufgaben enthalten**, deren Bewertung dann das Gesamtergebnis ausmacht. Sollte eine Einsendung Bearbeitungen zu allen drei Aufgaben enthalten, werden wir zwei davon zufällig auswählen und nur diese bewerten.

An dieser Runde dürfen nur Einzelpersonen teilnehmen, die in der ersten Runde in drei Aufgaben insgesamt mindestens 12 Punkte erreicht oder einem Team angehört haben, dem dieses gelungen ist. Gruppenarbeit ist in der zweiten Runde nicht zulässig.

Einsendeschluss ist der 24. April 2017.

Bearbeitung

Die Bearbeitung einer Aufgabe sollte zunächst eine nachvollziehbare und vollständige Lösung aller Teilaufgaben enthalten. **Pluspunkte** für eine höhere Bewertung kannst du erreichen, wenn du die Aufgabe dort, wo es möglich und sinnvoll ist, eigenständig weiterentwickelst. Sinnvoll sind inhaltliche Erweiterungen und Verbesserungen, etwa von Datenstrukturen und Algorithmen; uninteressant sind aufwändige Tricks, z. B. zur reinen Verschönerung der Benutzungsoberfläche. Begründe für jede Erweiterung, weshalb sie sinnvoll ist und ihre Realisierung eine eigene Schwierigkeit darstellt.

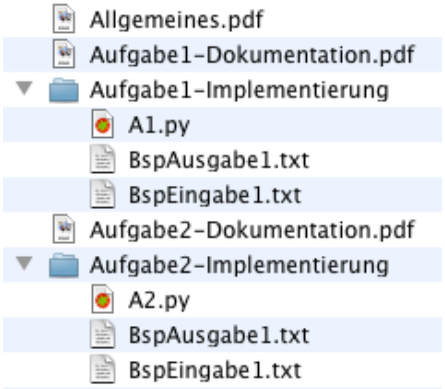
Grundsätzlich gelten die Vorgaben der 1. Runde weiter. Wesentliches Ergebnis der Aufgabenbearbeitung ist also eine **Dokumentation**, in der du den *Lösungsweg* sowie die *Umsetzung* des Lösungswegs in das dazugehörige Programm beschreibst. Die Beschreibung des Lösungswegs kann mit Hilfe (halb-)formaler Notationen präzisiert werden, die Beschreibung der Umsetzung mit Verweisen auf die entsprechenden Quellcode-Elemente. In die Dokumentation gehören auch *Beispiele* (Programmein- und -ausgaben oder Zwischenschritte), die zeigen, wie das Programm sich in unterschiedlichen Situationen verhält. Komplettiert wird die Dokumentation durch *Auszüge aus dem Quelltext*, die alle wichtigen Module, Methoden, Funktionen usw. enthalten.

Das zweite Ergebnis der Bearbeitung ist die **Implementierung**. Sie besteht aus dem zur Lösung der Aufgabe geschriebenen lauffähigen *Programm* und dem vollständigen *Quelltext*. Außerdem können Beispielein- und -ausgaben oder weiteres hilfreiches Material der Implementierung beigefügt werden.

Die Dokumentation zu einer Aufgabe mit allen oben genannten Bestandteilen muss als ein PDF-Dokument eingereicht werden. Dieses Dokument wird für die Bewertung ausgedruckt. **Es kann sein, dass für die Bewertung nur die Dokumentation herangezogen wird.** Diese sollte also einen lückenlosen und nachvollziehbaren Nachweis des Leistungsumfangs und der Funktionstüchtigkeit der Programme geben – und unbedingt auch Beispiele enthalten! Der Umfang der Dokumentation soll sich in Grenzen halten; eine gute Dokumentation vermittelt kurz und präzise alles Nötige, insbesondere die wesentlichen Ideen beim Lösungsweg. Nötig ist alles, was Interessierte mit guten Informatikkenntnissen, die die Aufgabenstellung kennen, wissen müssen, um den Lösungsweg zu verstehen und seine Umsetzung nachzuvollziehen. Entscheidend für eine gute Bewertung sind zwar richtige (und sauber umgesetzte) Lösungswege, aber die Dokumentation hat schon oft den Ausschlag für oder gegen das Weiterkommen gegeben.

Einsendung

Die Einsendung erfolgt wieder über das BWINF-PMS (pms.bwinf.de). Hochladen kannst du ein max. 40 MB großes ZIP-Archiv (z.B. VornameNachname.zip); seine Inhalte sollten so strukturiert sein wie im Bild rechts. Ein Dokument `Allgemeines.pdf` ist aber nur nötig, wenn du allgemeine, von den Aufgabenbearbeitungen unabhängige Anmerkungen zu deiner Einsendung machen willst. Die Dokumentationen müssen als PDF-Dokumente enthalten sein; Dateien in anderen Formaten werden möglicherweise ignoriert. Die Schriftgröße einer Dokumentation muss mindestens 10 Punkt sein, bei Quelltext mindestens 8 Punkt. Auf jeder Seite einer Dokumentation sollen in der Kopfzeile Verwaltungsnummer, Vorname, Name und Seitennummer stehen. Die Verwaltungsnummer steht auf der Teilnahmebescheinigung der ersten Runde.



Weitere Hinweise

Bei der Bewertung können Programme unter Windows (7 / 8 / 10), Linux, Mac OS X (10.11) und Android ausgeführt werden.

Fragen zu den Aufgaben können per Mail an `kontakt@bundeswettbewerb-informatik.de` oder telefonisch unter 0228–378646 (zu üblichen Arbeitszeiten) gestellt werden. Die Antwort auf E-Mail-Anfragen kann sich leicht verzögern. Informationen zur 2. Runde finden sich auf unseren Webseiten (`www.bundeswettbewerb-informatik.de`). In der Community von `einstieg-informatik.de` werden sicher wieder viele Teilnehmer über die Aufgaben diskutieren – ohne Lösungsideen auszutauschen.

Allen Teilnehmern der zweiten Runde wird bis Mitte Juni 2017 die Bewertung mitgeteilt. Die Besten werden zur Endrunde eingeladen, die im September 2017 vom Hasso-Plattner-Institut in Potsdam ausgerichtet wird. Dort werden die Bundessieger und Preisträger ermittelt und ausgezeichnet. Bundessieger werden in der Regel ohne weiteres Auswahlverfahren in die Förderung der Studienstiftung des deutschen Volkes aufgenommen. Außerdem werden Geld- und Sachpreise vergeben. Der Rechtsweg ist ausgeschlossen.

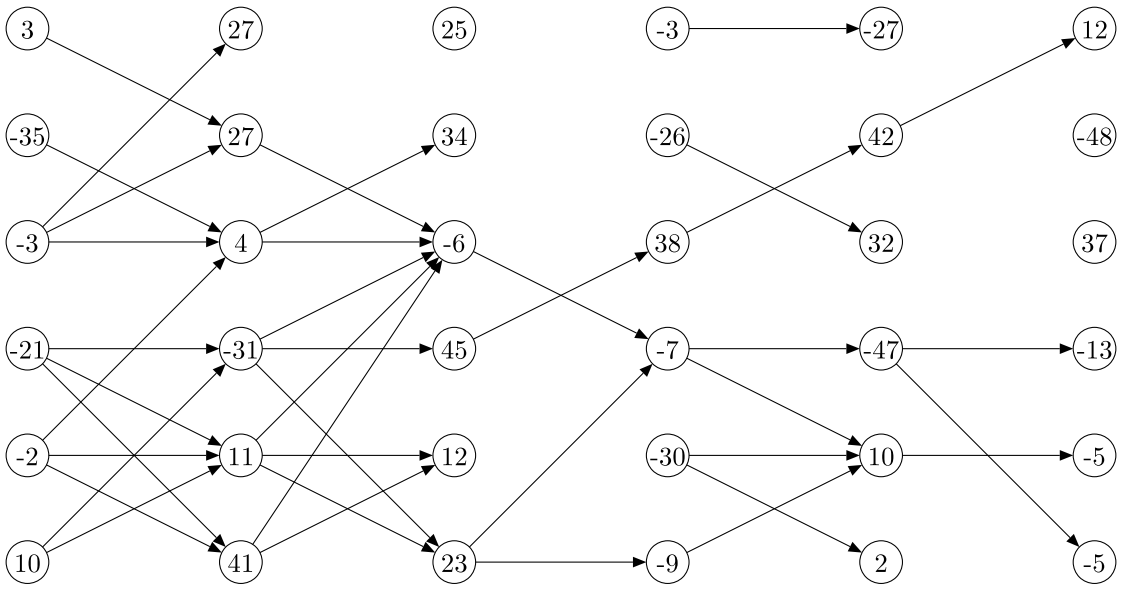
Zum Schluss noch einmal: Viel Spaß und viel Erfolg!

Aufgabe 1: Rosinen picken

Ein großes Firmenkonglomerat wird aufgelöst, und du hast die Aufgabe, aus den vielen Bruchstücken eine möglichst wertvolle Teilmenge auszusuchen. Die Werte der Einzelunternehmen sind unterschiedlich; einige sind sogar Verlustgeschäfte und haben daher einen negativen Wert.

Leider ist es nicht möglich, einfach nur die Unternehmen mit positivem Wert auszuwählen, weil es noch etliche Nebenbedingungen der Form „Nimmst du *A*, musst du auch *B* nehmen“ gibt. In einem Beispielszenario könntest du nur dann die Gummibärenfabrik wählen, wenn du ebenfalls die Zuckerraffinerie und die Altreifenverwertung nimmst – auch wenn diese für sich genommen wenig attraktiv sein sollten. Juristische oder wirtschaftliche Notwendigkeiten, die du nicht beeinflussen kannst, sind für diese Nebenbedingungen verantwortlich.

Graphisch können wir eine solche Situation so darstellen:



Die Kreise repräsentieren die Einzelunternehmen und sind mit deren Wert beschriftet. Ein Pfeil von *A* nach *B* bedeutet: Wenn man *A* auswählt, dann muss auch *B* gewählt werden.

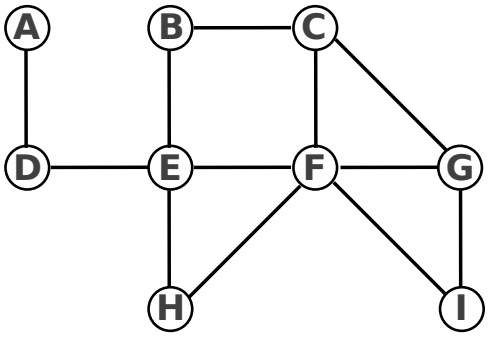
Aufgabe

1. Überlege dir und beschreibe ein Verfahren, mit dem man eine wertvolle Teilmenge finden kann.
2. Erstelle ein entsprechendes Programm und wende es auf die Beispiele an, die du auf der BWINF-Webseite zu dieser Runde findest.
3. Findet dein Verfahren garantiert immer ein bestmögliches Ergebnis oder kommt es diesem Ziel nur nahe? Ist die Laufzeit deines Programms erträglich? Natürlich ist eine lange Laufzeit bei einer so wichtigen Entscheidung akzeptabel, wenn die Zahlen in den Kreisen für Millionen Euros stehen.

Aufgabe 2: Rechtsrum in Rechthausen

Eine Statistik der Polizei von Rechthausen hat gezeigt, dass eine Vielzahl von Autounfällen durch inkorrektes Linksabbiegen verursacht wird. Der Bürgermeister von Rechthausen hat deshalb beschlossen, das Linksabbiegen per Dekret zu untersagen. Autofahrer sollen in Rechthausen nur noch geradeaus fahren oder rechts abbiegen dürfen. Diese Entscheidung führt zu viel Unmut der Bewohner von Rechthausen. Sie befürchten, sich dann in der Stadt nicht mehr zurechtzufinden.

Du möchtest Abhilfe schaffen und ein Navigationssystem erstellen, das nur Routen vorschlägt, bei denen nicht links abgebogen werden muss. Dein System erhält als Eingabe ein Straßenverzeichnis. Dieses besteht aus einer Menge V von *Kreuzungen* mit ihren jeweiligen Koordinaten sowie der Menge E von denjenigen Paaren von Kreuzungen, die direkt durch Straßenabschnitte verbunden sind. Da Rechthausen eine moderne Stadt ist, haben die Straßen keinerlei Kurven. Das Straßennetz von Rechthausen könnte zum Beispiel so aussehen (alles, was in Rechthausen als Kreuzung angesehen wird, ist durch einen Buchstaben gekennzeichnet):



Aufgabe

1. In der üblichen Situation, in der sich zwei Straßen im rechten Winkel kreuzen, ist es klar, was unter Linksabbiegen zu verstehen ist. Formuliere eine allgemeinere Definition des Linksabbiegens. Sie soll auch dann benutzt werden können, wenn die Anzahl d der Richtungen, aus denen man sich einer Kreuzung nähern kann, nicht vier ist oder der Winkel zwischen einer Richtung und der Richtung unmittelbar rechts von ihr nicht immer 90° ist. Am Wichtigsten ist, dass die Definition in jeder Situation angewendet werden kann und eine eindeutige Entscheidung herbeiführt, ob es sich beim Fahren von einem Straßenabschnitt über eine Kreuzung in einen nächsten Straßenabschnitt um Linksabbiegen handelt oder nicht. Versuche außerdem, mit deiner Definition die ursprüngliche Bedeutung des Linksabbiegens als besonders unfallträchtiges Verkehrsmanöver zu erhalten. Deine Definition soll insbesondere in den Fällen $d = 1$ (das Ende einer Sackgasse) und $d = 2$ sinnvolle Antworten liefern.

   Im obigen Beispiel sollten beispielsweise die Routen DADEF sowie GCB und FEBC erlaubt sein, die Routen DEB und IGF dagegen nicht.

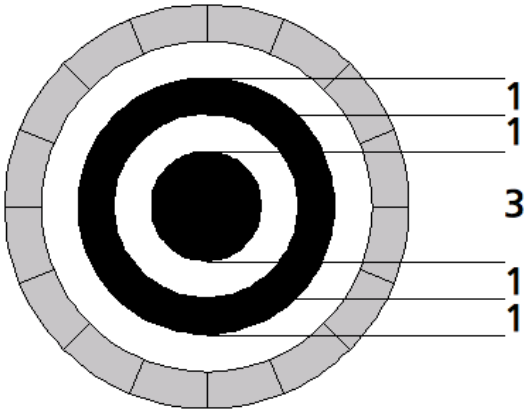
2. Veranschauliche deine Definition durch geeignete Beispiele.

3. Schreibe ein Programm, das einen kürzesten Weg ohne Linksabbiegen von einer Startkreuzung S zu einer Zielkreuzung T auf beliebigen Straßen findet, falls T von S aus überhaupt erreicht werden kann. Dabei sollst du die Weglänge auf zwei verschiedene Arten messen: zum einen nach der Anzahl, zum anderen nach der Gesamtlänge der auf dem Weg durchfahrenen Straßenabschnitte. Im obigen Beispiel: Was ist ein kürzester Weg ohne Linksabbiegen von A nach C? Und von H nach A?
4. Erweitere dein Programm so, dass es für ein gegebenes Straßenverzeichnis ermittelt, ob jede Kreuzung von jeder anderen Kreuzung aus ohne Linksabbiegen erreicht werden kann.
5. Erweitere dein Programm so, dass es ein Paar von Start- und Zielkreuzung ermittelt, für das das Verbot des Linksabbiegens die Weglänge um den größtmöglichen Faktor erhöht. Wende auch hier beide Weglängenmaße an.
6. Wende dein Programm auf die Beispiele an, die du auf der BWINF-Webseite zu dieser Runde findest.

Aufgabe 3: Kreis-Code

Damit Computer mit Text umgehen können, müssen die Zeichen geeignet codiert werden. Für die Codierung gibt es beispielsweise Binärcodes wie den ASCII-Code. Um Zeichen auch in Bildern darzustellen, so dass sie leicht mit dem Computer gelesen werden können, gibt es grafische Codes. Bekannt sind z.B. Strichcodes und QR-Codes, mit denen Ziffernfolgen und Texte codiert werden können.

Beim Erkennen dieser Codes muss das Lesegerät zum Code ausgerichtet werden. Grafische Codes, die auch in Fotos einfach lesbar sein sollen, müssen aber unabhängig von ihrer Orientierung erkannt werden können. Daher führen wir den Kreis-Code ein: Ein Zeichen dieses Codes besteht aus einem schwarzen Kreis und vier mit dem Kreis konzentrischen weißen und schwarzen Ringen. Der Durchmesser des Kreises ist dreimal so groß wie die Breite eines Ringes.



Der äußerste Ring (in der Abbildung grau gezeichnet) ist in 16 Segmente geteilt, die schwarz oder weiß sein können. Das heißt, der Ring entspricht, von oben im Uhrzeigersinn gelesen, einer Folge von 16 Bit. Mit diesen 16 Bit lassen sich Textzeichen codieren, z.B. bedeutet 0000 1110 0110 1101 den Buchstaben A. Damit der Kreis-Code aus jeder Blickrichtung gedeutet werden kann, sind die Codezeichen so gewählt, dass keines durch Rotation eines anderen hervorgeht; d.h. 1001 1011 0100 0011 bedeutet auch den Buchstaben A.

In dieser Aufgabe geht es darum, Kreis-Code in Bildern zu erkennen und zu decodieren.

Aufgabe

1. Schreibe ein Programm, das für ein vorgegebenes Schwarz-Weiß-Bild mit mehreren Kreis-Codezeichen deren Mittelpunkte bestimmt.
2. Erweitere dein Programm so, dass es für jedes Kreis-Codezeichen im Bild dessen Bedeutung ausgibt. Eine Tabelle mit der Bedeutung der Codezeichen findest du auf der BWINF-Webseite zu dieser Runde.
3. Auf Fotos kann die Abbildung von Codezeichen verzerrt, unscharf, verschattet und undeutlich sein. Erweitere dein Programm so, dass es trotz solcher Probleme Kreis-Code in Fotos decodieren kann. Wende es auf die Beispielfotos an, die du auf der BWINF-Webseite zu dieser Runde findest.