28. Bundeswettbewerb Informatik, 1. Runde

Lösungshinweise und Bewertungskriterien

Allgemeines

Zuerst soll an dieser Stelle gesagt sein, dass wir uns sehr darüber gefreut haben, dass einmal mehr so viele Leute sich die Mühe gemacht und die Zeit zur Bearbeitung der Aufgaben genommen haben.

Natürlich waren nicht alle Einsendungen perfekt, und einige eher äußerliche Anforderungen wurden häufiger missachtet. Im Einzelnen:

- Online-Anmelder wurden gebeten, ihre Nummer außen auf den Umschlag zu schreiben. Die meisten haben das gemacht, aber viele leider nicht. Das führt zu Komplikationen und möglicherweise zum Ausbleiben der versprochenen Rückmeldung per E-mail über den Eingang der Einsendung.
- Eine Gruppeneinsendung schicken Sie bitte komplett in einem gemeinsamen Umschlag, wir haben sonst größte Mühe, die Einsendungen richtig zuzuordnen. Wenn mehrere Einsendungen in einen Umschlag gesteckt werden, ist es besonders wichtig, bei der Online-Anmedung bzw. auf den Anmeldebögen die Zusammensetzung der Gruppe anzugeben. Außerdem: Eine Gruppe muss sich auf eine Lösung pro Aufgabe einigen, und Gruppenmitglieder können nicht gleichzeitig auch eine eigene Einsendung schicken.
- Seien Sie mit Ihrem Namen nicht so geizig! Schreiben Sie ihn ruhig häufiger, z. B. auf das erste Blatt jeder Aufgabe und auf Ihre CD.
- Lösungsidee, Programm-Dokumentation und insbesondere auch Programm-Ablaufprotokolle und ausreichend viele Beispiele müssen ausgedruckt sein. Wir können aus Zeit- und Kostengründen keine Ausdrucke machen und auch nicht jedes eingesandte Programm ausführen.
- Noch schlechter als Einsendungen nur auf Datenträgern wären für uns übrigens Einsendungen via E-Mail oder anderen Internet-Wegen, auch wenn das für die Teilnehmer noch so praktisch wäre. Papiereinsendungen sind (zumindest zur Zeit und sicher auch noch in den nächsten Jahren) einfach unumgänglich.

- Beispiele werden als Teile des Programm-Ablaufprotokolls immer erwartet. Zu wenige Beispiele und erst recht die Nichtbearbeitung vorgegebener Beispiele führen zu Punktabzug. Es ist auch nicht ausreichend, Beispiele nur auf CD abzugeben, ins Programm einzubauen oder den Bewertern das Erfinden und Testen von Beispielen zu überlassen. Ohne abgedruckte Beispiele ist die Bewertung einer Lösung in der knappen vorhandenen Zeit nicht möglich. Leider fehlten in vielen Einsendungen die Beispiele, was oft das Erreichen der zweiten Runde verhindert hat.
- Zu einer Einsendung gehören auch lauffähige Programme. Kompilierung von Quellcode ist während der Bewertung nicht möglich. Für die gängigsten Skript-Sprachen stehen Interpreter zur Verfügung. Nutzen Sie aber bitte dennoch jede Möglichkeit, eigenständig ausführbare Programme abzugeben.

So, vielleicht denken Sie ja an diese Anmerkungen, wenn Sie (hoffentlich) im nächsten Jahr wieder mitmachen.

Auch die folgenden eher inhaltlichen Dinge sollten Sie beachten:

- Lösungsideen sollten Lösungsideen sein und keine Bedienungsanleitungen oder Wiederholungen der Aufgabenstellung. Es soll beschrieben werden, welches Problem hinter der Aufgabe steckt und wie dieses Problem grundsätzlich angegangen wird. Ein einfacher Grundsatz: Bezeichner von Programmelementen wie Variablen, Prozeduren etc. dürfen nicht verwendet werden – eine Lösungsidee ist nämlich unabhängig von solchen Realisierungsdetails.
- Auch ein Programmablauf-Protokoll soll keine Bedienungsanleitung sein. Es beschreibt nicht, wie das Programm ablaufen sollte, auch nicht die zum Ablauf nötigen Interaktionen mit dem Programm, sondern protokolliert den tatsächlichen, inneren Ablauf eines Programms. Am besten protokolliert ein Programm seinen Ablauf selbst, z. B. durch Herausschreiben von Eingaben, Zwischenschritten oder -resultaten und Ausgaben.
- Oben wurde schon gesagt, dass Beispiele immer dabei sein sollten, zumindest eines davon in einem Programm-Ablaufprotokoll. Das hat seinen Grund: An den Beispielen ist oft direkt zu sehen, ob bestimmte Punkte korrekt beachtet wurden. Viele meinen nun, wir könnten die Programme ja laufen lassen und selbst auf Beispieldaten ansetzen, und liefern keine Beispiele oder nur Beispieldaten in elektronischer Form. Das können wir aber aus Zeitmangel in der Regel nicht. Außerdem ist nicht immer sicher, dass Programme, die auf dem eigenen PC laufen, auch auf einem anderen Computer ausführbar sind. Generell muss man sich darauf einstellen, dass nur das Papiermaterial angesehen wird!
- Mit den verschiedenen Beispielen sollten Sie wichtige Varianten des Programmablaufs zeigen, also auch Sonderfälle, die Ihre Lösung behandeln kann. Die Konstruktion solcher Testfälle ist eine ganz wesentliche Tätigkeit des Programmentwurfs.

Einige Anmerkungen noch zur Bewertung:

- Pro Aufgabe werden maximal fünf Punkte vergeben, bei Mängeln gibt es entsprechend weniger Punkte. Für die Gesamtbewertung sind die drei am besten bewerteten Aufgabenlösungen maßgeblich, es lassen sich also maximal 15 Punkte erreichen. Einen 1. Preis erreichen Sie mit 14 oder 15 Punkten, einen 2. Preis mit 12 oder 13 Punkten

und eine Anerkennung mit 9 bis 11 Punkten. Die Preisträger sind für die zweite Runde qualifiziert.

- Auf den Bewertungsbögen bedeutet ein Kreuz in einer Zeile, dass die (negative) Aussage in dieser Zeile auf Ihre Einsendung zutrifft. Damit verbunden ist dann in der Regel der Abzug eines oder mehrerer Punkte. Eine Wellenlinie bedeutet „na ja, hätte besser sein können“, führt aber alleine nicht zu Punktabzug. Mehrere Wellenlinien können sich aber zu einem Punktabzug addieren. Gelegentlich sind lobende Anmerkungen der Bewerter mit einem ’+’ versehen.
- Wellenlinien wurden übrigens häufig für die Dokumentation (Lösungsidee, Programm-Dokumentation, Programm-Ablaufprotokoll und kommentierter Programm-Text) verteilt, obwohl Punktabzug auch gerechtfertigt gewesen wäre.
- Aber auch so ließ sich nicht verhindern, dass etliche Teilnehmer nicht weitergekommen sind, die nur drei Aufgaben abgegeben haben in der Hoffnung, dass schon alle richtig sein würden. Das ist ziemlich riskant, da Fehler sich leicht einschleichen.

Zum Schluss:

- Sollte Ihr Name auf der Urkunde falsch geschrieben sein, können Sie gerne eine neue anfordern. Uns passieren durchaus schon mal Tippfehler, und gelegentlich scheitern wir bei Anmeldungen auf Papierformular an der ein oder anderen Handschrift.
- Es ist verständlich, wenn jemand, der nicht weitergekommen ist, über eine Reklamation nachdenkt. Gehen Sie aber bitte davon aus, dass wir kritische Fälle, insbesondere die mit 11 Punkten, schon genau und mit Wohlwollen geprüft haben.

Danksagung

An der Erstellung der Lösungsideen haben mitgewirkt: Thomas Leineweber und Johannes Pieper (Junioraufgabe und Aufgabe 1), Benito van der Zander (Aufgabe 2), Jan Balzer und Marvin Künnemann (Aufgabe 3) und Robert Czechowski (Aufgabe 5).

Die Aufgaben wurden vom Aufgabenausschuss des BWINF entwickelt, aus Vorschlägen von Torben Hagerup (Junioraufgabe), Monika Seiffert (Aufgabe 1), Wolfgang Pohl (Aufgabe 2), Dominic Battré (Aufgabe 3), Arno Pasternak (Aufgabe 4) und Peter Rossmanith (Aufgabe 5).

Junioraufgabe: Zahlendreher

Lösungsidee

Die Vizepräsidentin möchte ein System/Programm haben, mit dem bestimmt werden kann, wann eine Zimmernummer durch Drehen der Nummer nicht mit einer anderen Zimmernummer verwechselt werden kann. Es ist zu beachten, dass laut Aufgabenblatt die Ziffern in „durchsichtige Diamantenstäbchen eingelassen“ sind. Somit gibt es verschiedene mögliche Drehrichtungen mit unterschiedlichen Auswirkungen.

Um zu einer Lösung zu kommen, gibt es zwei mögliche Ansätze. Beim ersten wird versucht Regeln aufzustellen. Der zweite Ansatz arbeitet mit einer Drehtabelle. Damit werden die Zahlen ziffernweise je nach Drehrichtung übersetzt und verglichen.

In beiden Fällen muss man sich überlegen, ob man führende Nullen bei den Zimmernummern zulassen will wie z. B. bei der 100 die sich gedreht als 001 darstellen lässt. Lässt man sie nicht zu, liefert dies ein Kriterium, um eine verdrehte Zimmernummer zu identifizieren.

Schaut man sich zusätzlich die 1 auf dem Aufgabenblatt an, so kann sie auch gedreht gelesen werden, da es sich nicht um eine Segmentanzeige handelt. Sie wechselt deshalb nicht beim drehen von rechts nach links.

Aufstellen von Regeln

Zuerst kann man alle arabischen Ziffern in zwei Gruppen unterteilen:

- Als Nichtdrehziffern werden die 4 und die 7 bezeichnet. Im umgedrehten Zustand ist leicht zu erkennen, dass es sich dabei nicht um Ziffern handelt.
- Als Drehziffern bezeichnen wir die 1, 2, 3, 5, 6, 8, 9 und die 0. Diese ergeben bei einer Drehung in mindestens eine Richtung entweder eine andere Ziffer oder wieder sich selbst. Es gibt also drei Sorten von Drehziffern:
  - Paardrehziffern sind Ziffern, die beim Drehen in mindestens eine Richtung eine andere Ziffer ergeben: die 6 und die 9.
  - Selbstdrehziffern sind Ziffern, die beim Drehen in mindestens eine Richtung sich selbst ergeben: die 1, 3, 8 und 0.
  - Verschiedendrehziffern sind Ziffern, die beim Drehen in die eine Richtung sich selbst, bei der Drehung in eine andere Richtung, eine andere Ziffer ergeben: die 2 und 5.

Im zweiten Schritt wird der Aufbau der Zimmernummer betrachtet. Enthält sie eine Ziffer aus der Gruppe der Nichtdrehziffern, so ist auch die Zimmernummer eindeutig. Besteht sie nur aus Drehziffern, so ist die Wahrscheinlichkeit groß, dass es zu Verwechselungen kommen kann.

Es gibt aber für diesen Fall Ausnahmen. Die Ziffer 3 ergibt nur bei einer Drehung an der horizontalen Achse wieder sich selbst. Bei Drehung um die vertikale Achse oder einer Kombination ist klar zu erkennen, dass es sich bei ihr nicht um eine Ziffer handelt. Das bedeutet, wenn eine Zahl eine 3 enthält, muss nur überprüft werden, ob sie nach Drehung um die horizontale Achse eindeutig ist. Dieses ist automatisch der Fall, wenn eine Ziffer des Drehziffernpaars 6 oder 9 enthalten ist. Sollte dieses nicht der Fall sein, so darf sie nicht die Verschiedendrehziffern 2 oder 5 enthalten, da sie sonst zweideutig wäre.

Enthält die Zimmernummer keine 3, so muss die Drehung um die vertikale und oder die z-Achse betrachtet werden. Letztere Drehung kann auch durch Kombination einer Drehung um die horizontale und vertikale Achse erreicht werden. Dabei gibt es Fälle wie die Zimmernummer 88, die auch umgedreht wieder die 88 ergibt. Das gleiche gilt für 906 bei einer Drehung um die z-Achse. Bei diesen Zimmernummern müssen die Ziffern mit dem gleichen Abstand von rechts und von links entweder aus dem selben Drehziffernpaar stammen oder eine Selbstdrehziffer sein. Ist die Anzahl der Ziffern ungerade, so muss in der Mitte eine Selbstdrehziffer zu finden sein, damit die Zimmernummer eindeutig und eine Selbstdrehzahl ist.

Als zusätzlicher Punkt ist zu beachten, dass die 2 und 5 Selbstdrehziffern sind, wenn die Zimmernummer eine 6 oder 9 enthält. Durch diese Ziffern wird eine Verwechselung durch eine Drehung an der vertikalen Achse ausgeschlossen.

Drehtabelle

Eine anderer Ansatz besteht darin, die Zimmernummer ziffernweise so umzuschreiben, wie sie nach den Drehungen um die vertikale, horizontale sowie beide Achsen aussieht.

Dabei wird extra markiert, wenn nach einer Drehung eine Ziffer nicht als Ziffer erkennbar ist. Außerdem muss beim Aufschreiben der Drehung um die vertikale Achse und der Kombination der Drehung um die vertikale und die horizontale Achse die Reihenfolge der Ziffern getauscht werden.

Anschließend untersucht man die drei Ergebnisse. Enthalten alle drei gebildeten Ziffernfolgen mindestens eine Markierung, so ist die Zimmernummer eindeutig. Dieses gilt auch dann, wenn die gebildeten Ziffernfolgen, die keine Markierung enthalten, mit der Zimmernummer übereinstimmen. Ist beides nicht der Fall, so ist die Zimmernummer mehrdeutig.

Umsetzung

Umsetzung durch Prüfung der Regeln

Der Algorithmus, der die übergebene Zahl überprüft, muss die verschiedenen Regeln nacheinander abarbeiten. Diese sind hier aufgeführt:

- Zimmernummer enthält 4 oder 7 $\Longrightarrow$ eindeutig
- Zimmernummer enthält 3:
  - Zimmernummer enthält 6 oder 9 $\Longrightarrow$ eindeutig
  - Zimmernummer enthält keine 2 oder 5 $\Longrightarrow$ eindeutig
  - sonst $\Longrightarrow$ nicht eindeutig
- Zimmernummer enthält 6 oder 9:
  - alle Ziffernpaare mit gleichem Abstand sind 1, 2, 5, 8, 0 oder 6 mit 9 $\Longrightarrow$ eindeutig
  - sonst $\Longrightarrow$ nicht eindeutig
- sonst
  - alle Ziffernpaare mit gleichem Abstand sind 1, 8, 0 $\Longrightarrow$ eindeutig
  - sonst $\Longrightarrow$ nicht eindeutig

Dabei ist zu beachten, dass z. B. bei der Zimmernummer 181 die Ziffer 8 den gleichen Abstand von links und rechts hat und somit auch gegen sich selbst geprüft werden muss.

Nichtzulassung von führenden Nullen Wurde eine Zahl nicht als eindeutig identifiziert und sind führende Nullen bei einer Zimmernummer nicht zugelassen, so müssen noch nacheinander folgende Regeln ausgewertet werden:

- Zimmernummer hat keine 0 am Ende $\Longrightarrow$ nicht eindeutig
- Zimmernummer enthält 6 oder 9 $\Longrightarrow$ eindeutig
- Zimmernummer enthält 2 oder 5 $\Longrightarrow$ nicht eindeutig
- sonst $\Longrightarrow$ eindeutig

Umsetzung durch Drehtabelle

Für die Umsetzung kann folgende Drehtabelle genutzt werden. Dabei ist das x die Markierung für den Fall, dass eine Ziffer nach einer Drehung nicht mehr als Ziffer erkannt werden kann.

Ziffer	horizontale	vertikale	beide (z-Achse)
1	1	1	1
2	5	5	2
3	3	x	x
4	x	x	x
5	2	2	5
6	x	x	9
7	x	x	x
8	8	8	8
9	x	x	6
0	0	0	0

Nichtzulassung von führenden Nullen Enthält eine Zimmernummer eine Null am Ende und es sind keine führenden Nullen zugelassen, so kann man sich bei der Drehtabelle auf die Überprüfung der Drehung um die horizontale Achse beschränken. In den anderen beiden Fällen stünde die Null am Anfang der gedrehten Zimmernummer.

Ergebnisse

An mindestens drei Beispielen muss gezeigt werden, dass der Algorithmus auch funktioniert. Dazu gehören verschiedene eindeutige und nicht eindeutige Zimmernummern, wie z. B. 639 (eindeutig), 659 (eindeutig), 235 (nicht eindeutig), 810 (eindeutig, wenn führende Nullen nicht zugelassen sind), 191 (nicht eindeutig) und 1691 (eindeutig).

Interessant (aber nicht gefordert) ist die Betrachtung der Anzahl der Zimmer, die in eine Abstellkammer umgewandelt werden oder als normales Zimmer bestehen bleiben können. In der unteren Tabelle ist letztere Anzahl aufgeführt. Es ist dabei immer mit der Zimmernummer 1 begonnen worden.

Anzahl Zahlen	Eindeutig (Mit Nullen)	In %	Eindeutig (Ohne Nullen)	In %
10	5	50	6	60
100	53	53	58	58
1000	625	63	658	66
10000	7153	72	7398	74
100000	79501	80	81358	81
1000000	855077	86	869002	87
10000000	8990257	90	9093010	91
100000000	93008209	93	93755814	94

Schaut man sich diese Zahlen an und überschlägt welche Kosten der Umbau macht, so ist auch eine ganz andere Lösung für die Aufgabe möglich: Man sollte der Vizepräsidentin empfehlen, mit den Schlüsselanhängern zum Hersteller zu gehen und die Zimmernummern mit einem Punkt in der unteren rechten Ecke zu ergänzen. Da die Platzierung des Punktes allgemeine Konvention ist, macht dieser alle Zimmernummern eindeutig.

Bewertungskriterien

- In der Aufgabenstellung ist eine Implementation nicht explizit gefordert; ihr Fehlen stellt deshalb keinen Mangel dar.
- Das in der Aufgabenstellung geforderte Verfahren muss nachvollziehbar beschrieben sein. Das gilt insbesondere für einen regelhaften Ansatz, bei einem tabellarischen Ansatz ist das einfacher.
- Alle drei möglichen Drehungen müssen beachtet worden sein, auch wenn die Aufgabenstellung nur auf zwei eingeht.
- In der Aufgabenstellung sind Zimmernummern verschiedener Länge vorhanden. Die Länge der Zimmernummer darf also nicht konstant festgelegt werden.
- Verschieden lange Zimmernummern sprechen dafür, führende Nullen zu verbieten. Das Verfahren muss ein solches Verbot berücksichtigen. Es ist aber auch in Ordnung, wenn führende Nullen zugelassen sind. Es wird sogar hingenommen, dass das Problem übersehen wurde. Eine weitere Besonderheit kann die Ziffer 1 darstellen; falls sie ähnlich wie die 3 als Zahl behandelt wird, die nur bei Drehung um die horizontale Achse erhalten bleibt, muss dies begründet sein.

- Selbstverständlich darf das Verfahren auch nicht aus anderen als den schon genannten Ursachen fehlerhafte Ergebnisse liefern.
- Es müssen mindestens drei Beispiele angegeben werden. Es sollte zu jedem Beispiel geschildert werden (insbesondere bei fehlender Implementation), wie das beschriebene Verfahren zu seiner Entscheidung kommt.

Aufgabe 1: Pizza-Service

Aus der Aufgabenstellung geht nicht klar hervor, wie umfangreich die Lösung zu dieser Aufgabe sein soll. Da der letzte Satz des einführenden Textes „die Pizza“ erwähnt, genügt es, sich auf die Zusammenstellung einer Pizza zu beschränken. Im folgenden wird eine umfassendere Lösung beschrieben, die eine komplette Bestellung von möglicherweise mehreren Pizzen im Lieferservice erlaubt.

Teilaufgabe 1

Beschreibe, wie der Vorgang der Bestellannahme mit Hilfe des Systems ablaufen soll.

Die Bestellannahme hat ein gravierendes Problem: Man hat es mit Kunden zu tun, die sich nur selten an festgelegte Vorgänge halten. Deshalb sollte der Vorgang der Bestellannahme so flexibel wie möglich gehalten werden. Dabei muss aber sichergestellt werden, dass in der Pizzeria alle notwendigen Daten aufgenommen werden. Grundsätzlich sind folgende Daten von Bedeutung:

- Daten zu den einzelnen Pizzen: Größe und Belag mit optionaler Menge dieser Pizza.
- Bei externen Bestellungen: Name, Anschrift und Telefonnummer, damit die Pizzen auch ausgeliefert werden können

Da ein Informatiksystem gebaut werden soll, kann davon ausgegangen werden, dass die Bestellung nicht zuerst auf einem Schmierzettel entgegengenommen wird und später erst mit dem System verarbeitet wird. Das System soll direkt während der Bestellannahme mit den entsprechenden Daten versorgt werden.

Der Vorgang kann allgemein wie folgt beschrieben werden:

- Es können jederzeit allgemeine Informationen zur Bestellung eingegeben werden. Wenn es z. B. eine externe Bestellung ist, besteht immer die Möglichkeit, die Kontaktdaten (Name, Anschrift, Telefonnummer für Rückfragen) einzugeben.
- Es muss immer möglich sein, eine weitere Pizza in die Bestellung aufzunehmen. Andererseits sollte auch zu einer vorhandenen Pizza gesagt werden können, dass nun zum Beispiel zwei statt drei Exemplare dieser Sorte bestellt werden.
- Zu einer Pizza kann bei der Bestellannahme jederzeit von groß auf klein umgeschaltet werden.
- Zu einer Pizza können Beläge entfernt und hinzugefügt werden. Es sind hier zwei Varianten vorstellbar. In der einen Variante kann jeder Belag maximal einmal hinzugefügt werden. In der zweiten Variante kann man erlauben, dass die Beläge beliebig oft ausgewählt werden können („doppelt Knoblauch und doppelt extra Käse“).
- Am Ende der Bestellannahme soll eine Zusammenfassung der Bestellung ausgegeben werden, die in die Küche zur Bearbeitung gegeben werden kann. Natürlich muss auch der Preis für die Pizza bzw. die Preise der einzelnen Pizzen und der Endpreis ausgegeben werden.

Teilaufgabe 2

Entwirf für das System eine Benutzungsschnittstelle und stelle sie graphisch dar.

Zusätzlich zu den Anforderungen, die sich aus dem obigen beispielhaften Ablauf ergeben, gibt es noch weitere wünschenswerte Anforderungen an die Benutzungsschnittstelle:

- Bei der Bestellannahme soll im Falle der Frage des Kunden auch schnell gesagt werden können, wieviel ein Belag auf einer großen oder kleinen Pizza mehr kosten würde.
- Zur Information sollte immer mit ausgegeben werden, wie viele Pizzen die Bestellung momentan enthält und wie teuer die Bestellung im Moment ist.
- Natürlich soll man auch eine Bestellung wieder abbrechen und zum Ausgangspunkt zurückkehren können.

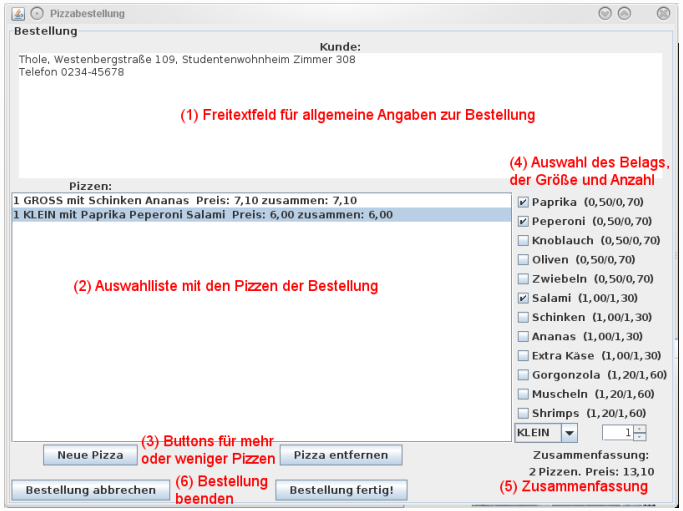


Abbildung 1: Entwurf einer Benutzungsschnittstelle

Aufgrund der gesamten Anforderungen wird die in Abbildung 1 gezeigte Benutzungsschnittstelle vorgeschlagen. Die einzelnen Bestandteile sind:

- Ein Freitextfeld (1), in dem alle möglichen Daten zur Bestellung eingegeben werden können. Bei einer externen Bestellung können dort Anschrift und Telefonnummer eingegeben werden. Bei einer internen Bestellung können Bedienung und Tisch angegeben werden.
- Eine Liste mit den bisher bestellten Pizzen (2). In der Liste kann gesehen werden, wieviele Pizzen welcher Größe mit welchen Belägen bestellt wurden.
- Über die Buttons (3) können Pizzen der Liste hinzugefügt werden oder auch von der Liste entfernt werden.

- Wenn in der Liste eine Pizza ausgewählt ist, werden in den genauen Informationen zur Pizza (4) die Beläge, die Größe und die Anzahl der Exemplare dieser Pizza noch einmal angezeigt. Gleichzeitig kann man direkt diese Informationen durch An- und Abwählen von Belägen oder durch das Ändern von Größe und Anzahl anpassen. Hier wird auch angezeigt, wieviel ein Belag bei einer großen bzw. bei einer kleinen Pizza kostet.
- Eine Zusammenfassung (5) zeigt die Gesamtanzahl der Pizzen und den Gesamtpreis für die aktuelle Bestellung an.
- Mit den unteren Buttons (6) kann die Bestellung angenommen oder abgebrochen werden. Bei Annahme der Bestellung kann dann auf einem Drucker oder auf eine andere Art die Bestellung aus- und weitergegeben werden.

Teilaufgabe 3

Entwickle ein Datenmodell für das System, aus dem hervorgeht, welche Daten verarbeitet werden sollen und wie diese miteinander zusammenhängen.

Für das Datenmodell, das die obige Benutzungsschnittstelle unterstützt, ergeben sich folgende Eigenschaften (Schlüsselworter sind *kursiv* geschrieben):

- Eine *Bestellung* hat einen *Freitext* zur Bestellung.
- Eine *Bestellung* besteht aus einer oder mehreren *Pizzabestellungen*.
- Eine *Pizzabestellung* hat folgende Eigenschaften:
  - eine Menge von *Belägen*;
  - eine *Größe* (groß oder klein);
  - eine *Anzahl*, wie oft diese Pizza gewünscht wird.
- Ein *Belag* hat einen *Preis*, der von der *Größe* der Pizza abhängt.
- Eine *Pizza* hat einen *Grundpreis*, der von der *Größe* der Pizza abhängt.

Daraus ergibt sich das in der Abbildung 2 dargestellte vereinfachte UML-Diagramm für das Datenmodell. Zusätzlich gibt es Methoden zur Berechnung der verschiedenen Preise.

Teilaufgabe 4

Implementiere die Kernfunktionalität des Systems.

Die minimale Kernfunktionalität ist:

- Eingabe der Daten einer Bestellung in einem beliebigen gewählten (und beschriebenen) Format.
- Korrekte Berechnung des Endpreises dieser Bestellung.
- Ausgabe des Endpreises.

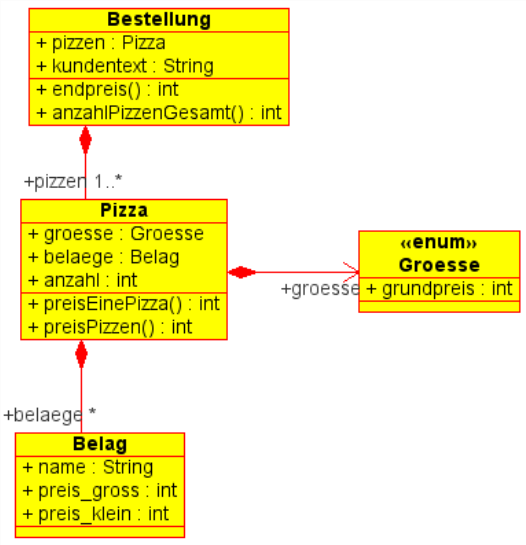


Abbildung 2: Datenmodell (vereinfachtes UML-Diagramm)

Nicht verlangt wird eine Implementierung der grafischen Benutzungsoberfläche. Es kann auch an der Kommandozeile oder mit Ein- und Ausgabedateien gearbeitet werden. Ein festes Verdrahten der Bestellung im Programm ist nicht erlaubt. Die verschiedenen Grundpreise können im Programm festgelegt sein, sollten aber einfach anpassbar sein, d. h. die Werte dürfen nicht mehrfach im Programm verstreut sein.

Bewertungskriterien

- Bei Teilaufgabe 1 ist es wichtig, dass verschiedene Arten der Bestellung abgedeckt sind. Insbesondere sollte berücksichtigt sein, dass der Besteller seine Angaben (etwa zu Größe und Belägen) ändern kann. Die Beschreibung soll von der erst in der nächsten Teilaufgabe entwickelten Benutzungsschnittstelle möglichst unabhängig sein. Formulierungen wie „anklicken“ sind hier also fehl am Platz, werden aber toleriert, wenn die Beschreibung ansonsten allgemein genug ist.
- Bei Teilaufgabe 2 sind prinzipell alle Arten von grafischen Benutzungsschnittstellen erlaubt. Dies kann von einer angepassten Excel-Tabelle bis hin zu ausgefuchsten Fensterbasierten Programmen oder Web-Anwendungen gehen. Wie in der Aufgabenstellung angedeutet, genügt es, wenn die Benutzungsschnittstelle als Zeichnung vorliegt. Der in Teilaufgabe 1 beschriebene Bestellvorgang muss mit der Benutzungsschnittstelle durchführbar sein; dies sollte klar werden, auch aus Beispielen.

- Bei Teilaufgabe 3 ist es wichtig, dass ein Datenmodell beschrieben wird und nicht etwa ein Ablaufmodell. Die Beschreibung muss grundsätzlich ununabhängig von der benutzten Programmiersprache sein. Das Datenmodell muss aber zum Bestellablauf und zur Benutzungsschnittstelle passen. Es wird keine formale Beschreibung des Datenmodells (z.B. als UML- oder ER-Diagramm) verlangt, die Struktur muss aber klar werden.
- Bei Teilaufgabe 4 wird eine einfache Implementierung erwartet, die die wesentlichen Teile des Datenmodells und zumindest die oben erwähnte minimale Funktionalität realisiert. Es wird erwartet, dass die Implementation den vorherigen Beschreibungen entspricht. Außerdem muss eine Auswahl dessen, was zur Kernfunktionalität gehört, begründet werden. Gleichzeitig werden die üblichen Dokumentationen für die Implementierung verlangt (Programmdokumentation, Programmablaufprotokolle/Beispiele, Programmtext). Als Beispiel genügt eine Pizza-Bestellung, wenn daran alle Eigenschaften des Bestellsystems demonstriert werden.

Aufgabe 2: Handytasten

Kostenberechnung

Da in der Aufgabenstellung verlangt wird, dass die „Zahl von Tastendrücken durchschnittlich minimal“ sein soll, ist es sinnvoll, für die Kosten den Erwartungswert der Zahl der Tastendrücke zu nehmen. Dieser Erwartungswert ist für jeden Buchstaben die Wahrscheinlichkeit seines Vorkommens multipliziert mit seiner Position auf der Taste. Der gesamte Erwartungswert ist dann die Summe der Erwartungswerte der einzelnen Buchstaben.

Formuliert man dies, weil’s so schön ist, mathematisch, so erhält man: Sei N die Zahl der Buchstaben (also 26 in unserem Fall) und K die Zahl der Tasten (=8). Außerdem sei für jeden Buchstaben i mit $1 \leq i \leq N$ eine relative Häufigkeit h_i gegeben. Diese relative Häufigkeit – die vorliegenden Eingabedaten geben an, wie oft ein bestimmter Buchstabe unter 100 000 Buchstaben vorkommt – wird im weiteren wie eine Wahrscheinlichkeit verwendet.

Für die Berechnung des Erwartungswertes gibt es nun verschiedene Möglichkeiten:

Position mal Häufigkeit

Für jeden Buchstaben i mit $1 \leq i \leq N$ sei b_i die Position auf der jeweiligen Taste. Da die Buchstaben alphabetisch sortiert sein sollen, muss der erste Buchstabe offensichtlich der erste auf der ersten Taste sein und jeder nachfolgende Buchstabe liegt entweder eine Position weiter auf dieser Taste oder auf der nächsten (übersprungene Tasten können vernachlässigt werden, da sie in einer optimalen Lösung nicht vorkommen). Damit lässt sich b definieren als:

$$b_1 = 1$$

$$b_{i+1} = \begin{cases} b_i + 1 & \text{wenn } b_i \text{ und } b_{i+1} \text{ auf derselben Taste sind} \\ 1 & \text{wenn } b_i \text{ und } b_{i+1} \text{ auf unterschiedlichen Tasten sind} \end{cases}$$

Da alle K Tasten verwendet werden sollen, gibt es genau K Buchstaben mit Position 1:

$$|\{i | b_i = 1\}| = K$$

Der Vorteil dieser Darstellung ist, dass man die beiden Folgen b und h nun als Vektoren auffassen kann, deren Skalarprodukt $\vec{h} \cdot \vec{b}$ den Erwartungswert ergibt:

$$\sum_i b_i \cdot h_i$$

Man kann den Normierungsfaktor $1/\sum_i h_i$ der Summe der relativen Häufigkeiten (100 000 bei den vorgegebenen Eingabedaten) ignorieren, da er vom gesuchten b unabhängig ist.

Eine Variante dieses buchstabenbezogenen Kostenmaßes entsteht, wenn man den Buchstaben zuordnet, auf welcher Taste (anstatt an welcher Position der jeweiligen Taste) sie sind. Für einen Buchstaben $1 \leq i \leq N$ auf Taste j sei $t_i = j$. Seine Position auf der Taste ist dann die

Anzahl der Buchstaben, die sich auf der gleichen Taste vor ihm bzw. auf seiner Position befinden: $b_i = |\{j | t_j = t_i \wedge j \leq i\}|$. Das obige Kostenmaß kann also auch so beschrieben werden (wieder ohne Normalisierung):

$$\sum_{i=1}^{N} |\{j | t_j = t_i \wedge j \leq i\}| \cdot h_i$$

Tastenwahrscheinlichkeiten

Ein tastenbezogenes Kostenmaß ergibt sich, wenn man für jede Taste i mit $1 \leq i \leq K$ angibt, welches der erste Buchstabe k_i auf ihr ist. Für diese k_i gilt, da die Buchstaben alphabetisch sortiert sind und nur einmal vorkommen:

$$k_1 = 1$$

$$1 \leq k_i < k_j \leq N \qquad \text{für } 1 \leq i < j \leq K$$

Auf einer Taste i liegen dann die Buchstaben $k_i, \ldots, k_{i+1} - 1$ (wobei $k_{K+1} = N + 1$ gesetzt ist). Wie häufig die Taste i gedrückt wird (also der Erwartungswert der Taste), ergibt sich dann als

$$\sum_{j=k_i}^{k_{i+1}-1} (j - k_i + 1) \cdot h_j$$

Der gesamte Erwartungswert und damit ein zweites Kostenmaß ist die Summe der Tasten-Erwartungswerte:

$$\sum_{i=1}^{K} \sum_{j=k_i}^{k_{i+1}-1} (j - k_i + 1) \cdot h_j$$

$$= \sum_{i=1}^{N} (i+1) \cdot h_i - \sum_{i=1}^{K} \sum_{j=k_i}^{k_{i+1}-1} k_i \cdot h_j$$

$$= \sum_{i=1}^{N} (i+1) \cdot h_i - \sum_{i=1}^{K} k_i \sum_{j=k_i}^{k_{i+1}-1} h_j$$

Da der erste Term konstant ist, sind die variablen Kosten einer Tastenbelegung:

$$\simeq -\sum_{i=1}^{K} k_i \sum_{j=k_i}^{k_{i+1}-1} h_j$$

Dabei gibt der Ausdruck $\alpha_i = \sum_{j=k_i}^{k_{i+1}-1} h_j$ die Wahrscheinlichkeit an (wieder wird der Faktor $1/100\,000$ ignoriert), dass ein Buchstabe auf Taste i gedrückt werden muss. Eine optimale Belegung muss also $-\sum_{i=1}^{K} k_i \alpha_i$ minimieren (bzw. den Absolutbetrag maximieren).

Berechnung

Alle Möglichkeiten durchrechnen

Wie lässt sich nun für ein gegebenes Kostenmaß eine optimale Belegung berechnen? Zur Beantwortung dieser Frage ist es wichtig zu wissen, wie viele verschiedene Belegungen es überhaupt geben kann. Zunächst stellen wir fest: Die Folge b_i (also eine Belegung) lässt sich vollständig dadurch bestimmen, dass man angibt, welche Buchstaben die ersten auf ihren jeweiligen Tasten sind, also durch die Menge $E = \{i \,|\, b_i = 1\}$ mit $|E| = K$.

Die Anzahl der Belegungen entspricht also der Anzahl der K-elementigen Teilmengen von $\{1, \ldots, N\}$ (das ist die Menge aller Buchstaben): Es gibt genau $\binom{N}{K}$. Auf Grund der alphabetischen Sortierung ist der Buchstabe 1 (also A) immer ein erster Buchstabe. Es muss deswegen $1 \in E$ sein, so dass die Anzahl der verschiedenen Belegungen der Zahl $\binom{N-1}{K-1}$ der Anzahl der $K-1$-elementigen Teilmengen aus den $N-1$ Buchstaben ohne den Buchstaben 1 entspricht.

Da es also „nur“ $\binom{N-1}{K-1} = 480\,700$ mögliche Belegungen gibt, kann man diese durchprobieren, jeweils die Kosten berechnen und die optimale Belegung finden.

Dynamische Programmierung

Insbesondere für andere Konstellationen mit mehr Buchstaben und mehr Tasten wäre ein effizienteres Verfahren sehr wichtig. Es gibt netterweise ein Verfahren mit Dynamischer Programmierung, das die Lösung mit einem Berechnungsaufwand von $O(N^2K)$ findet. Man beachte, dass $N^2K = 4732$ im Handyfall gilt; das ist etwa ein Hundertstel der Anzahl aller Belegungen, die den Berechnungsaufwand beim obigen Verfahren bestimmt.

Zur Lösung mit Dynamischer Programmierung wählt man das oben beschriebene tastenbezogene Kostenmaß. Nun ist zu beobachten, dass in der Kostenformel $-\sum_{i=1}^{K} k_i \sum_{j=k_i}^{k_{i+1}-1} h_j$ die summierten Terme nicht von früheren k_i abhängen. Deswegen kann man mit diesem Maß die Kosten einer Belegung berechnen, die ein Suffix der Buchstaben $1 \ldots N$ (ein Suffix sind hier alle Buchstaben ab einem gegebenen Buchstaben i) einem Suffix der Tasten $1 \ldots K$ zuordnet.

Nun erstellt man eine Tabelle, in der man für jede Taste i und jeden Buchstaben j speichert, wie die optimalen Kosten wären, wenn es weder frühere Tasten noch frühere Buchstaben gäbe. Für ein solches Paar (i, j) berechnet man diese Kosten, indem man für jeden späteren Buchstaben l die Kosten berechnet, die entstehen, wenn l der erste Buchstabe auf der Taste $i+1$ wäre, und davon das Minimum wählt.

Damit erhält man die Kosten der optimalen Belegung. Die Belegung selbst erhält man, indem man in der Tabelle außer dem Minimum noch den Index des minimalen Elementes speichert. Dann kann man vom Feld $(1,1)$ ausgehen (der erste Buchstabe muss ja auf der ersten Position der ersten Taste stehen: $k_1 = 1$) und findet dort den optimalen ersten Buchstaben für Taste 2, k_2. In Feld $(2, k_2)$ steht dann der optimale erste Buchstabe für die dritte Taste usw.

1: **for** $j = 1$ to N **do**
2: $\quad COSTS[K][j] \leftarrow \left(-j \sum_{l=j}^{N} h_l \right)$
3: **end for**
4: **for** $i = K-1$ to 1 **do**
5: $\quad$ **for** $j = 1$ to N **do**
6: $\quad\quad COSTS[i][j] \leftarrow \min \left\{ COSTS[i+1][l] - \left(j \sum_{m=j}^{l-1} h_m \right) \middle| l > j \right\}$
7: $\quad$ **end for**
8: **end for**

Abbildung 3: Berechnung der optimalen Belegung mit Dynamischer Programmierung

Bewertungskriterien

- In Teil 1 muss ein Kostenmaß klar definiert werden. Dies geschieht am besten mit einer einfachen mathematischen Formel, eine präzise sprachliche Beschreibung genügt aber.
- Wenn die optimale Belegung durch Ausprobieren aller Möglichkeiten bestimmt wird, sollten es nur erlaubte Belegungsmöglichkeiten sein: Buchstaben in alphabetischer Reihenfolge, jede Taste (von 2 bis 9) belegt. Wenn weitere Belegungen (z. B. mit leeren Tasten) in die Suche einbezogen werden, erhöht dies unnötig den Verarbeitungsaufwand.
- Die Ergebnisse sollten bezüglich des gewählten Kostenmaßes optimal sein. Andernfalls sollte zumindest erkannt und begründet sein, warum dem nicht so ist.
- Für die sieben vorgegebenen Beispiele sollen die (korrekten) Ergebnisse in der schriftlichen Einsendung dokumentiert sein. Es genügt jeweils eine Angabe der gefundenen Belegung in einer nachvollziehbaren (nicht zwingend grafischen) Notation.

Lösungen

Deutsch:158780

```
AB
CD
EFG
HIJK
LM
NOPQ
RS
TUVWXYZ
```

Englisch:164682

```
AB
CD
EFG
HIJK
LM
NOPQ
RS
TUVWXYZ
```

Finnisch:154931

```
ABCD
EFGH
IJ
KLM
NOPQ
RS
TUVWX
YZ
```


Französisch:150573

```
AB
CD
EFGH
IJK
LM
NOPQ
RS
TUVWXYZ
```

Italienisch:148640

```
AB
CD
EFGH
IJK
LM
NOPQ
RS
TUVWXYZ
```

Spanisch:154210

```
AB
CD
EFGH
IJK
LM
NOPQ
RS
TUVWXYZ
```

Polnisch:178678

```
ABCD
EFGH
IJ
KLM
NOPQ
RSTUV
WX
YZ
```

Aufgabe 3: Wegfehler

Lösungsidee

Wege einzeichnen Die erste Hürde der Wegfehler-Aufgabe besteht aus der Umrechnung der Breiten- und Längengrade in Pixelangaben für die Karte. Hierbei ist zu beachten, dass die nördliche Breite nach oben wächst und die östliche Länge nach rechts größer wird. Der Kartenausschnitt ist sehr klein, daher ist es nicht notwendig, sich um die Krümmung der Gradlinien bzw. die Art der Projektion[1] zu kümmern. Wir verwenden zur Berechnung der x- und y-Koordinaten aus den eingelesenen Längen- und Breitengraden $(long, lat)$ eines Wegpunktes folgende Formeln:

$$\begin{aligned} latscale &= 50.7820 - 50.7716 \\ longscale &= 6.0918 - 6.0711 \\ x &= \frac{long - 6.0711}{longscale} \cdot imgwidth \\ y &= \frac{50.7820 - lat}{latscale} \cdot imgheight \end{aligned}$$

Der Ursprung $(0,0)$ der Bildkoordinaten x und y liegt bei dieser Variante links oben im Bild. Die Zahlen beziehen sich dabei auf die Angaben zum Kartenausschnitt aus der Aufgabenstellung. *latscale* und *longscale* entsprechen der Höhe und Breite der Karte und dienen zum Skalieren der Koordinaten. Die Wegpunkte können nun eingelesen, umgerechnet und auf der Karte eingezeichnet werden. Schon hier sollten Punkte, die nicht im Kartenausschnitt liegen, erkannt und gelöscht werden. Es bietet sich an, aufeinander folgende Punkte mit direkten Linien zu verbinden, da in unseren GPS-Logs die Auflösung hoch genug ist, um ansehnliche Ergebnisse zu erzielen.

Wege korrigieren Der zweite Teil der Aufgabe – das Erkennen und Korrigieren fehlerhafter Wegpunkte – gestaltet sich etwas schwieriger. Wir benötigen zunächst ein Kriterium, um einen Wegpunkt als „ungewöhnlich“ bzw. Messfehler zu klassifizieren. Üblicherweise berechnet man in der Statistik dazu die so genannte Standardabweichung einer Reihe von Messwerten und geht davon aus, dass eine über zweifache (oder auch dreifache) Abweichung vom Mittelwert aus einem Fehler resultiert. In diesem konkreten Fall bietet es sich an, die Geschwindigkeiten, mit der sich Dominic von Punkt zu Punkt fortbewegt, zu berechnen. Wir verwenden die Einheit „Pixel pro Sekunde“ [2] und berechnen die mittlere Geschwindigkeit μ. Bezeichnet n die Anzahl der Punkte, so sind für $k = 1, \ldots, n$ x_k und y_k die Koordinaten und t_k der Zeitpunkt umgerechnet in Sekunden seit Mitternacht. Mit

$$s_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

bezeichnen wir für $i = 2, \ldots, n$ den Abstand zwischen den Punkten (x_{i-1}, y_{i-1}) und (x_i, y_i). Für den Mittelwert μ der Geschwindigkeiten ergibt sich nun:

[1] Für Interessierte: http://upload.wikimedia.org/wikipedia/commons/f/f8/Netzentwuerfe.png

[2] Es geht hier nur um relative Vergleiche, daher sind solch merkwürdige Einheiten in Ordnung.

$$\begin{aligned} \mu &= \frac{1}{n-1} \cdot \sum_{k=2}^{n} \frac{s_k}{t_k - t_{k-1}} \\ V &= \frac{1}{n-1} \cdot \sum_{k=2}^{n} \left(\frac{s_k}{t_k - t_{k-1}} - \mu \right)^2 \\ \sigma &= \sqrt{V} \end{aligned}$$

Als nächstes wird die Varianz V berechnet, dies ist die durchschnittliche Abweichung vom Mittelwert. Um größere Abweichungen eher zu „bestrafen“, werden alle Differenzen in der Summe quadriert. Die Standardabweichung σ ist als Quadratwurzel der Varianz definiert.

Wegpunkte löschen. Nun werden alle Punkte der Reihe nach betrachtet. Bewegt sich Dominic mit einer Geschwindigkeit, die um mehr als 2σ von der mittleren Geschwindigkeit μ abweicht, auf einen Punkt zu, so wird der Punkt als Messfehler markiert und gelöscht.

Wegpunkte hinzufügen. Hin und wieder erweisen sich Dominics GPS-Logs als unvollständig: Es gibt Zeitabschnitte, für die sich keine Koordinaten finden lassen. Wie können wir ihm helfen und eine realistische Vermutung über den tatsächlich zurückgelegten Weg einzeichnen?

Wir nehmen an, dass Dominic in den fraglichen Zeitabschnitten nicht wesentlich von einer geraden Verbindung zwischen den vorhandenen Punkten am Anfang und am Ende der fehlenden Messreihe abgewichen ist. Die fehlenden Wegpunkte werden in gleichmäßigen Abständen auf der Verbindungslinie verteilt. Eventuell könnte man sich hier am Straßenverlauf orientieren, für eine angemessene Bearbeitung der Aufgabe ist dies jedoch nicht nötig.

Bei allzu großen Messlücken, die mehr als 20 Sekunden umfassen, ist der gegangene Weg schlecht zu rekonstruieren. Deswegen wird in solchen Fällen nur eine gestrichelte Linie gezeichnet, um den groben Verlauf zu verdeutlichen und anzuzeigen, dass hier Daten fehlen.

Weitere Lösungsmöglichkeiten

Die Aufgabe lässt sehr viel Spielraum für weitere spannende Lösungsansätze: Anstelle der strikten Auftrennung von Anzeige des Weges und Fehlererkennung könnte man versuchen, beide Ziele miteinander zu kombinieren. Dabei kann man nach Verfahren suchen, eine gewünschte Form der Wegkurve bestmöglich an die vorhandenen Punkte anzupassen, wobei kleine Abweichungen von den tatsächlichen Koordinaten erlaubt werden.

Weiterhin kann das Ergebnis auch verschönert werden, indem die Punkte nicht durch Linien und damit eher „kantig“, sondern durch Kurven (etwa Splines) verbunden wären.

Um mit fehlenden Koordinaten besonders gut umgehen zu können, kann man auch Verfahren entwickeln, die versuchen, den Wegverlauf in diesem Zeitraum zu erraten und nach bestimmten Kriterien anzupassen, wie zum Beispiel:

- „Glätte“ des Verlaufs, d.h. eher unnatürliche „Ecken“ werden durch eine glatte Kurve eliminiert.

- Anpassung an den Straßenverlauf.
- Anpassung an die Durchschnittsgeschwindigkeit.

Neben der Fehlerkorrektur durch das Löschen einzelner Wegpunkte gibt es noch die Möglichkeit, Messdaten leicht zu verändern und Wegpunkte zu verschieben. Zur Begradigung des Weges ist es vorstellbar, den Straßenverlauf zu berücksichtigen, da anzunehmen ist, dass Dominic eher auf als neben der Straße läuft. Problematisch können dabei verschiedene Straßenfärbungen und schwarze Beschriftungen werden. Außerdem durchquert er eventuell ab und zu Gebäude, da er als Student die Abkürzungen durch die Aachener Uni zu kennen scheint.

Ablaufprotokolle

Eine Implementierung der Lösungsidee liefert für die GPS-Logs aus dem BWINF-Material die Karten aus Abbildung 4.

Die Ergebnisse für die erste Tour sind sehr zufriedenstellend: Der Weg orientiert sich stark an den Straßen, anscheinend fehlerhafte Punkte wurden gelöscht und plausibel ersetzt.

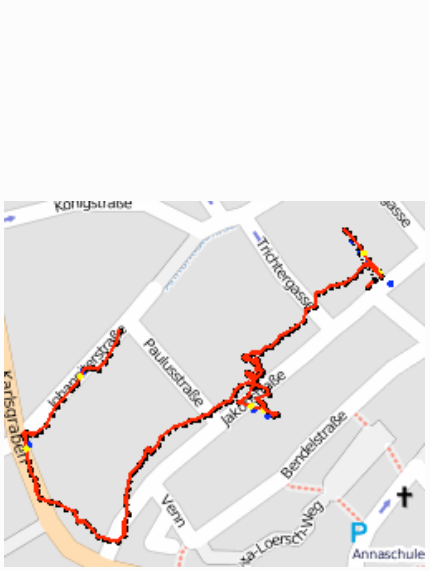
Im zweiten Log gibt es einige Zeitsprünge, es fehlen Messdaten über Zeiträume von 20 Sekunden bis sieben Minuten. In solchen Fällen wurde der vermutete Weg als gestrichelte schwarze Linie gezeichnet. Insbesondere die Punkte nördlich und östlich des Rundgangs erscheinen seltsam und könnten gelöscht werden. Allerdings ist es unklar, wie sich Dominic hier verhalten hat. Die Geschwindigkeit weicht bei den schwarzen Linien nicht wesentlich vom berechneten Mittelwert ab, daher ist es immer noch realistisch, dass er sich an diesen Punkten tatsächlich aufgehalten hat.

Die dritte Tour scheint ein Parcours-Lauf zu sein: Dominic weicht zuweilen stark von der Straße ab, hält sich für längere Zeiten in einem begrenzten Bereich auf, aber behält dabei weitestgehend seine Durchschnittsgeschwindigkeit. Dies sieht zwar merkwürdig aus, entspricht aber am besten den Messdaten.

Bewertungskriterien

Teilaufgabe 1: Einlesen und Darstellen eines gegebenen Weges:

- Die Koordinaten werden korrekt auf den Kartenausschnitt projiziert.
- Die zur Projektion verwendeten Berechnungen sollten dokumentiert und begründet werden.
- Der Verlauf des Weges ist erkennbar. Eine Abbildung der Punkte ist allein nicht ausreichend; die Punkte sollten zumindest direkt durch Linien verbunden werden.



(a) log3.csv (b) log1.csv

(c) log2.csv

Abbildung 4: Korrigierte Wege

Teilaufgabe 2: Erkennen und Korrigieren von Fehlern in den GPS-Logs:

- Es sind verschiedene Kriterien vorstellbar, die einen Punkt als wahrscheinlichen Messfehler identifizieren lassen. Mindestens ein Kriterium muss diskutiert und korrekt umgesetzt werden.
- Die Zeitpunkte, an denen die Position ermittelt wurde, sollten bei der Erkennung von Messfehlern berücksichtigt werden. Der zeitliche Kontext spielt neben dem Abstand der Punkte eine große Rolle. Falls generell nur mit Distanzen gearbeitet wird, muss das begründet sein. Allgemein muss der Umgang mit fehlenden Zeitintervallen (bis zu sieben Minuten in log2.csv) vernünftig begründet sein.
- Ein Kriterium sollte skalierbar und nicht zu sehr auf die Beispieldaten zugeschnitten sein. Beispielsweise ist die Argumentation „Punkt *A* ist weiter als 50 Pixel entfernt und deswegen ein Messfehler“ in der Regel nicht ausreichend. Beliebig allgemein muss die Lösung aber auch nicht sein: Wird etwa der Maßstab der Karte betrachtet und davon ausgegangen, dass Dominic Fußgänger oder Radfahrer ist, können wir dies bei guter Begründung gelten lassen.
- Die Touren aus allen drei Logs im BWINF-Material sind vom Programm auf dem Kartenausschnitt aufgezeichnet. Die Ergebnisse werden diskutiert und bewertet.

Aufgabe 4: EU-WAN

Lösungsidee

Bei dieser Aufgabe ist eines klar: Durchprobieren aller Möglichkeiten ist nicht vernünftig. Theoretisch käme jedes Pixel der Kartengrafik als Senderstandort in Betracht, und davon gibt es zu viele. Und es sind auch dann immer noch zu viele, wenn man die Weiten des Meeres, von denen aus ein Sendemast gar kein Land erreichen kann, außer Betracht lässt.

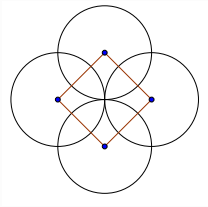
Ein besserer systematischer Ansatz fällt uns leider nicht ein. Es wird also ein heuristischer Ansatz benötigt, der nach einer möglichst klugen Strategie vorgeht. Hier sollen kurz einige erwähnt werden.

Platzieren und Wegstreichen

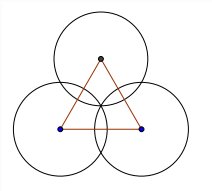
Die Sender werden auf allen möglichen Positionen platziert. Nun kann man diejenigen Sender wieder wegstreichen. die gar kein Land erreichen. Da bleiben aber immer noch zu viele übrig. Klar ist aber auch, dass die allermeisten Landpunkte von sehr vielen Sendern bestrahlt werden. Davon sind viele überflüssig, da alle Punkte, die sie erreichen, auch von anderen Sendern erreicht werden. Doch welche dieser überflüssigen Sender streicht man sinnvollerweise? Ein Ansatz ist, die Sender danach zu sortieren, wie viele Landpunkte sie erreichen, und dann „von oben“ her wegzustreichen, falls sie überflüssig sind. So kann man gültige Konfigurationen von etwa 80 Sendern ermitteln.

Senderaster

Das Übel an diesem Problem ist, dass die Landkarte so überhaupt keine vernünftige Struktur hat. Europa ist leider nicht rechteckig oder ein sonstwie ordentlich geformtes Gebilde. Nun denn, Ordnung lässt sich schaffen. So kann man die Senderpositionen auf einem Raster anordnen, das gröber ist als mit allen Pixeln aus dem vorigen Abschnitt. Es sollte derart beschaffen sein, dass bei Platzierung der Sender auf diesem Raster kein Punkt unbedeckt bleibt. So kann man ein Raster anlegen, in der die Punkte einer Rasterzeile genau um den Durchmesser des Sendegebiets auseinander liegen – aber jede Zeile gegenüber der anderen um den Radius *r* des Sendegebietes versetzt wird:



Noch Sender-sparender ist ein Raster, bei dem die Punkte gleichseitige Dreiecke bilden, deren Umkreisradius genau der Senderadius *r* ist.



Ist ein Raster definiert, werden alle Rasterpunkte als Senderstandorte gewählt, von denen aus ein Sender Landpunkte erreicht. Dabei wird es nur wenige überflüssige Sender geben, die ggf. gestrichen werden können. Platzierung von Sendern entlang Senderastern liefern für den Fall, dass Sender auch außerhalb des EU-Landgebiets liegen dürfen, ordentliche Ergebnisse. Mit dem „Quadratraster“ kommt man mit etwa 80 Sendern aus, beim „Dreiecksraster“ sind es gar nur rund 70. Zur Optimierung dieser Werte kann das Raster geeignet verschoben werden. Ist die Platzierung innerhalb des EU-Landgebiets gefordert, kommen einige benötigte Rasterpunkte aber nicht in Frage. Für sie muss auf dem Land Ersatz gefunden werden. Dabei können Flächen frei werden, die dann von zusätzlichen Sendern abgedeckt werden müssen. Bei Verwendung des Dreiecksrasters können auch hier Werte von gut 70 Sendern erreicht werden.

Allgemeine Heuristiken

Zur Lösung von Optimierungsproblemen wie diesem gibt es einige allgemeine (d.h. auf viele Probleme anwendbare) heuristische Verfahren. Eines davon ist die algorithmische Anwendung von Evolutionsstrategien. Dabei werden ganze Mengen (Populationen) von Lösungen eines Problems erzeugt, Teile einzelner Lösungen verändert (mutiert) und neue Lösungen aus Teilen anderer Lösungen zusammengesetzt (rekombiniert). Dabei darf auch der Zufall eine Rolle spielen. Eine Evolutionsstrategie lässt sich auch für diese Aufgabe entwickeln, wie eine Einsendung zeigt. Sie erreicht eine Abdeckung mit 66 Sendern.

Eine ähnliche allgemeine Heuristik mit Zufallselementen ist die des „Simulierten Abkühlens“ (engl.: Simulated Annealing). Unter Einsatz dieses Prinzips erreicht eine Einsendung eine Abdeckung mit 60 Sendern (auf EU-Gebiet platziert).

Gierig vorgehen

Eine gute Abdeckung wird möglichst wenig Sendegebiet verschenken, also möglichst wenig ins Meer oder in Nicht-EU-Gebiete hineinstrahlen. Von daher ist es sinnvoll, von den Gebietsrändern aus Sender so zu setzen, dass sie möglichst viel Gebiet abdecken. Eine derartige Strategie, die in jedem einzelnen Schritt einen möglichst großen Teil des zu lösenden Problems

Abbildung 5: Lösung mit 60 Sendern, auf EU-Gebiet platziert.

zu erledigen versucht, nennt man auch „gierig“. Gierige Strategien sind aber häufig nicht optimal. In diesem Fall kann es zu ungeeigneten Platzierungen kommen. Deshalb kann dieser Ansatz verbessert werden, indem gelegentlich Sender zufällig gelöscht werden; frei werdendes Gebiet wird dann wieder „gierig“ abgedeckt. So lässt sich eine Abdeckung mit 60 Sendern (nur auf EU-Gebiet platziert) erreichen (s. Abbildung 5).

Bewertungskriterien

Diese Aufgabe ist schwierig zu bewerten. Es gibt keine Teilaufgaben, keine klaren Fehler oder Versäumnisse. Folgende Punkte wurden bei der Bewertung berücksichtigt:

- Ein Programm wird in der Aufgabenstellung nicht ausdrücklich verlangt, nicht einmal die Beschreibung oder Anwendung eines Verfahrens. In der Informatik sollte die Lösung eines Problems aber nicht allein durch Probieren bestimmt werden. Ohne Systematik, ohne ein Verfahren bzw. eine Lösungstrategie lässt sich ein verlässliche Aussage über die Qualität einer gefundenen Lösung nicht treffen. Zumindest eine Strategie zur Platzierung der Sender ist also zu entwickeln. Aber auch die Beschreibung und Anwendung eines Verfahrens genügt alleine nicht. Die erzielten Ergebnisse müssen nachprüfbar sein, und dazu wird eine Implementierung nötig – oder auch (noch besser) eine mathematische Analyse des Verfahrens, die aber nicht erwartet wird.
- Das angewandte Verfahren darf nicht beliebig ineffizient sein. Ein systematisches Ausprobieren war nur selten anzutreffen, da es allzu lange Laufzeiten nach sich zieht.
- Die angewandte Strategie sollte keine offensichtlichen Schwächen enthalten, und wenn doch, sollten sie erkannt und angesprochen werden. Sendegebiete sollten nicht unnötig sich überschneiden oder Nicht-EU-Gebiet überdecken – in diesen Fällen ist Verbesserungspotenzial allzu offensichtlich. Abdeckungen mit 80 Sendern (wie sie etwa einfache Rasterplatzierungen schaffen, wenn Sender auch außerhalb des EU-Gebiets platziert werden) sollten erreichbar sein.
- Beide in der Aufgabenstellung angesprochenen Fälle (Senderstandorte nur auf EU-Gebiet bzw. Standorte auch im Meer und außerhalb der EU) müssen behandelt sein.
- Lösungen sollten auch grafisch angegeben sein; sie sind sonst nicht nachvollziehbar.

Aufgabe 5: Teleskop

Lösungsidee

Öffnet man die Bilder aus dem zur Aufgabe gehörenden Material mit einem Graphikbearbeitungsprogramm, dann kann man folgende Dinge feststellen:

- Das große Bild ist jeweils exakt drei mal so hoch und drei mal so breit wie das kleine Bild ($+/-$ ein Pixel).
- Das niedrig aufgelöste Bild enthält genau den gleichen Bildausschnitt wie das hoch aufgelöste Bild.
- Der fehlerhafte Bereich ist klar zu erkennen, besteht aus einfachen Rundungen und hat klar abgrenzende Kanten.

Weil beide Bilder in verschiedenen Auflösungen vorhanden sind, ist es schwierig, beide zu vergleichen. Wir müssen zunächst das kleine Bild hochskalieren oder das große Bild herunterskalieren. In der Informationstheorie spricht man von „Upsampling“ und „Downsampling“.

Vergrößern Beim Vergrößern bekommt man eine höhere Auflösung, mit der man arbeitet. Dazu muss man nicht vorhandene Bildinformationen aus den vorhandenen interpolieren. Pixelwiederholung bringt praktisch keine höhere Auflösung.

Bilinear Bei dieser Interpolationsmethode berechnen sich die Farbwerte durch die vier benachbarten Punkte, deren Farbwert über eine (genauer gesagt zwei) lineare Funktion gewichtet werden.

Bikubisch Bei dieser Interpolation werden meistens 16 statt nur 4 benachbarte Pixel betrachtet. Außerdem setzt sich die Gewichtungsfunktion nicht aus linearen, sondern aus kubischen Funktionen (Polynome vom Grad 3) zusammen.

Die bikubische Interpolation liefert wesentlich bessere Ergebnisse, ist aber auch umständlicher zu implementieren.

Verkleinern Das Verkleinern ist einfacher zu implementieren als das Vergrößern, allerdings müssen hier vorhandene Informationen verworfen werden.

Mittelwertfilter Dieser Filter, auch Boxfilter genannt, berechnet ein neues Pixel einfach, indem er den Durchschnitt aus den $m \cdot n$ Pixeln bildet, die das neue Pixel überdecken.

Gauß-Filter Im Gegensatz zum Boxfilter spielt hier auch die Entfernung zur Mitte des neuen Pixels eine Rolle. Die Gewichtung ergibt sich über die Gaußkurve (Glockenkurve) aus der Entfernung.

Wenn man beide Bilder auf die gleiche Größe gebracht hat, kann man sie nun Pixel für Pixel vergleichen. (Natürlich kann man die Bilder auch direkt während des Vergleichs skalieren.) Ab einem bestimmten Schwellenwert für die Differenz der Pixelwerte – entweder absolut oder relativ – muss man dann das hochaufgelöste Bild anpassen, um diese Differenz wieder auszugleichen. Da die Bilder gleichmäßig abgedunkelt sind, die Farbwerte der Pixel also mit einem konstanten $k < 1$ multipliziert worden sind, empfiehlt es sich, die relative Differenz zu betrachten, k zu ermitteln und dann die Farbwerte mit dem Kehrwert $\frac{1}{k}$ zu multiplizieren. Da es sich um eine farbneutrale Verdunklung handelt, habe ich die Farbwerte einfach aufaddiert, um eine geringfügig höhere Genauigkeit zu bekommen.

Programm

Meine Implementation verkleinert das große auf die Maße des kleinen Bildes und setzt der Einfachheit halber nur einen Mittelwertfilter dafür ein. Außerdem ist es darauf angewiesen, dass die Größenverhältnisse ganzzahlig sind.

Es liest zunächst die Bilder ein, und vergleicht dann immer ein Pixel aus dem kleinen Bild mit entsprechend vielen aus dem großen, ermittelt eine relative Differenz und entscheidet dann, ob die Pixel aus dem großen Bild angepasst werden und in einem weiteren Bild die entsprechenden Pixel für die Fleckenausgabe markiert werden. Schließlich wird das korrigierte große Bild und das Fleckenbild wieder in eine Datei geschrieben.

Vergleichen der Bilder

Nach dem Einlesen der Bilder werden die Skalierungsverhältnisse in der x-Richtung und in der y-Richtung ermittelt. Da von ganzzahligen Größenverhältnissen ausgegangen wird, werden diese Verhältnisse als erst als `double` ermittelt und dann auf ganze Zahlen gerundet.

Nun wird jedes Pixel des kleinen Bildes betrachtet und die Summe über die einzelnen Farbwerte ermittelt. Für dieses Pixel werden dann die gemäß der Skalierung entsprechenden Pixel aus dem großen Bild betrachtet und für diese auch alle Farbwerte zusammen gezählt. Die Summe aus dem großen Bild wird durch die beiden Skalierungsfaktoren geteilt. Dieser Vorgang entspricht dem Boxfilter bei der Verkleinerung.

Der Vergleich der beiden Summen liefert nun das Verhältnis der Helligkeiten des aktuellen Bildbereichs aus dem großen und dem kleinen Bild. Damit haben wir unseren Faktor k und auch seinen Kehrwert k^{-1} (`faktor`).

Korrigieren des großen Bildes

Wenn k^{-1} jetzt einen bestimmten Schwellenwert überschreitet (Standardmäßig 1,5), können nun die Farbwerte für alle entsprechenden Pixel auf dem großen Bild korrigiert werden, indem alle einzelnen Farbwerte jeweils mit k^{-1} (`faktor`) multipliziert werden. Außerdem werden die entsprechenden Pixel auf dem Fleckenbild dunkel gefärbt, um den Flecken zu markieren.

Ergebnisse

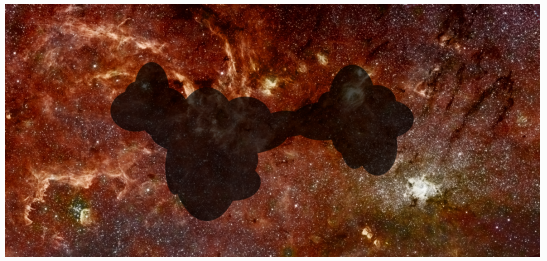


Abbildung 6: Fehlerhafte Aufnahme „galactic-center“

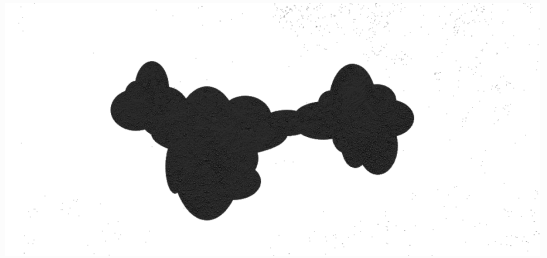


Abbildung 7: Fehlerhafter Bereich in der Aufnahme „galactic-center“

Für das Eingabebeispiel „galactic-center“ (Abb. 6) wird der fehlerhafte Bereich recht klar ermittelt (Abb. 7). Die Aufnahme kann damit ordentlich korrigiert werden (Abb. 8).

Qualität

Wenn man nun die beiden Ausgabedateien genau analysiert, fallen einem zwei Dinge auf:

- Am Rand des Fleckens ist eine konstraststarke Line zu erkennen (Abb. 9). Dies liegt daran, dass immer mehrere Pixel auf einmal betrachtet werden. Liegen diese größtenteils außerhalb des Fleckens, kommt es zu keiner oder nur zu leichter Anpassung, weshalb die Pixel im Flecken zu dunkel bleiben. Liegen die Pixel dagegen größtenteils im Flecken, kommt es zu einer starken Korrektur, was die Pixel außerhalb des Fleckens zu hell werden lässt.
- Das Fleckenbild ist nicht einheitlich grau/weiß, sondern unterliegt leichten Schwankungen, obwohl der Flecken eigentlich vollkommen gleichmäßig verdunkelt ist. Daher kann man davon ausgehen, dass das korrigierte Bild auch an Bereichen, die komplett im Flecken liegen, nicht vollständig dem Original entsprechen sondern leicht abweichen.

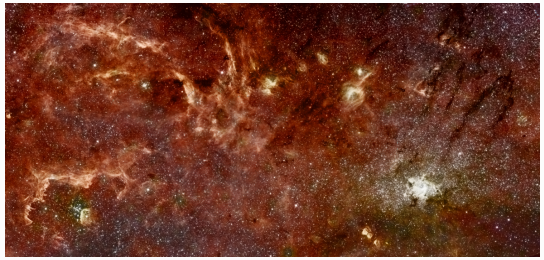


Abbildung 8: Korrigierte Aufnahme „galactic-center“

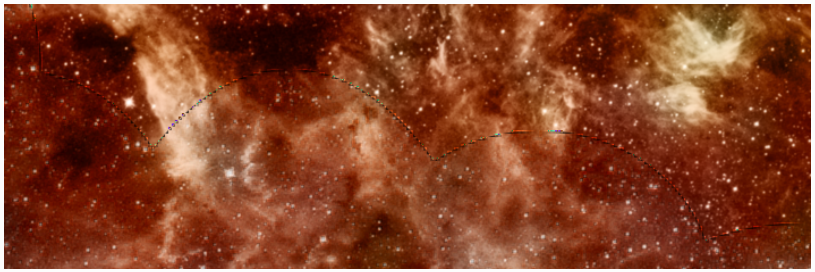


Abbildung 9: Ausschnitt aus der korrigierten Aufnahme „galactic-center“

- Schließlich lässt sich an einigen Punkten noch eine leichte Farbverfälschung erkennen. An dunklen Stellen kommen zum Beispiel sehr niedrige Farbwerte vor, was zu Rundungsfehlern führt, die bei der Korrektur bemerkbar werden.

Bewertungskriterien

- Eine vollständige Lösung leistet zwei Dinge: die Bestimmung des fehlerhaften Bereichs und die Korrektur des Fehlers.
- Bei dieser Aufgabe ist die Verwendung von Funktionsbibliotheken aus dem Grafikbereich naheliegend. Es muss aber begründet werden, warum welche Funktionen verwendet werden und was sie zur Lösung beitragen.
- In den Beispielbildern ist die Verdunklung am Rand „unscharf“. Von daher wird es bei der Fleckenbestimmung zu ähnlichen Effekten kommen. Dies sollten nicht zu stark auffallen (die Randlinie sollte möglichst dünn sein) sowie erkannt und begründet sein.
- Dass der Fehler „mit Hilfe der niedrig aufgelösten Aufnahmen möglichst gut behoben“ werden soll, bedeutet nicht, dass die Daten des kleinen Bildes in das große Bild hineinkopiert werden sollen. Diese Methode wurde häufig angewandt, führt aber zwangsläufig zu Informationsverlust und Unschärfe im Bereich des Fleckens. Besser funktioniert die Zurücknahme der Verdunklung, wie oben beschrieben.
- Zu beiden geforderten Beispielen sollen das rekonstruierte Bild und der Fehlerbereich dokumentiert sein, und zwar sowohl auf Papier als auch in elektronischer Form. Wie der Flecken angegeben wird (etwa als vom restlichen Bild abgetrennter Bereich, durch Umrahmung, . . .), ist nicht entscheidend; wichtig ist, dass er klar erkennbar ist.
- Die Qualität der Bildrekonstruktion muss nicht perfekt sein, darf aber auch keine allzu großen erkennbaren Mängel haben. Auf jeden Fall müssen (Teil 3) die erreichte Qualität besprochen und die Mängel der Lösung erkannt sein.

Aus den Einsendungen: Perlen der Informatik

Allgemeines

Dies erscheint mir zwar unlogisch, jedoch wird es auch durch die Aufgabenstellung untermauert.

Aufgrund der Einfachheit der Aufgabe, der Übersichtlichkeit des Quelltextes und aus Zeitmangel – vorwiegend jedoch aus Zeitmangel – wird auf eine genauere Programmdokumentation verzichtet.

Lassen Sie die Installation nur von Personen mit entsprechender Fachkompetenz durchführen!

Bitte nicht allzu ernst nehmen. Ich habe regelmäßig meine 5 Minuten.

Da die Aufgabe nicht nur aus Text besteht, ist es sinnvoll, eine grafische Oberfläche zu erzeugen.

… und damit einen Algorithmus ins Rollen bringt.

Eigentlich hatten wir hier keine Lösungsidee. Wir haben einfach angefangen zu programmieren, und dann hat alles seinen Lauf genommen. *Eine Art „Extreme Programming“?*

Technisches

Diese Methode enthält verschiedene if-Schleifen.

Die Endlosschleife ist dagegen nicht so einfach berechenbar. […] Trotzdem ist auch diese Komponente linear.

Die while-Schleife ist eine Endlosschleife. Allerdings wird, wenn die letzte Zahl eingelesen wurde, eine Exception ausgelöst, durch die dann die Schleife abgebrochen wird.

Vorsicht, Struktur ist if else if else if else if usw., nicht if if else if …

Ich benutze Brute Force, weil es zu umständlich ist, einen langen Algorithmus zu erfinden, welcher dann erst auch noch bewiesen werden muss.

Zahlendreher

Ist die Zimmernummer eine Abstellkammer, …

Man muss die Ziffern 0-9 in verschiedene Gruppen teilen. Jede Gruppe verhält sich anders, […] so verwirrt die eine, und die andere behebt die Verwirrung wieder.

Pizza-Service

Man muss immer vom DAU ausgehen.

Jeder Pizza wird genau ein Preis zugeordnet, aber ein Preis kann zu mehreren Pizzen gehören.

Die Größe der Pizza ist eine entscheidende.

Der Button drückt optisch aus, dass er nicht betätigt ist.

Die Arbeit in einem Pizza-Service ist so schon schwer und fordert viel geistige Leistung.

Für jede Belagsart sollte ein Spinner die Anzahl bestimmen.

Kassenbong

Nach einer Baumstruktur mittels If-Abfragen werden Checkbox-Parameter verarbeitet, daher, nach Betätigen des Schalters, wird eine Abfrage eingestellt, die voraussetzt, dass eine Checkbox, in diesem ersten Fall die Abfrage nach der Größe der Pizza, angeklickt wurde. *Puh!*

Für den Fall, dass der Kunde ins Restaurant kommt, kriegt er erklärt, wie die Pizzas in dieser Pizzeria gemacht werden.

Da muss der Benutzer nur seinen Hacken rein machen.

Praktisch ist, dass der durchschnittlich verfressene Nutzer nur wenige Klicks machen muss, um seine Wunschpizza zu erhalten.

Diese Pizza-Aufgaben können ja fast kein Zufall sein; verkauft der BWINF etwa die Programme?

Offensichtlich glaubt der BWINF, dass die Teilnehmerzahlen steigen, wenn die Aufgaben mehr im Erlebnisspektrum eines Schülers liegen.

Handytasten

Daraus folgt, dass zu Anfang 27 Tasten existieren.

Im optimalsten *(!)* Fall sind die „Kosten“ gleich der „Häufigkeit“.

Maximale Kosten sind als optimal anzusehen.

Finnisch und Polnisch bilden eine Ausnahme, was durch das relativ häufige Vorkommen von seltenen Buchstaben zu erklären ist.

Wenn man eine Handytaste drücken muss, kostet dies einen Tastendruck.

`klingonisch.txt` *Name einer Eingabedatei*

Die finnische Tastatur sieht genau so aus wie die deutsche. Unserer Meinung nach ist dies relativ unwahrscheinlich.

Wegfehler

Dazu wird der vor-vorherige Punkt am vorherigen Punkt gespiegelt, so dass der eigentliche Punkt auf der Geraden durch die beiden vorherigen Punkte den gleichen Abstand zum vorherigen Punkte hat wie der vor-vorherige.

… lässt mich der Gedanke an ein Koordinatensystem, welches über den Kartenausschnitt gelegt wird, nicht los.

… sollte Dominic schneller als der Weltrekordsprinter laufen, …

Die Höchstgeschwindigkeit zu Fuß setze ich (sehr vorsichtig) auf den aktuellen Weltrekord: 100 Meter in 9,58 Sekunden. Wer weiß, ob Dominic mit Nachnamen „Bolt“ heißt.

Diese Methode ist aber viel zu unrealistisch, da Dominic so durch Häuser gehen würde, was er nur mit einem Panzer könnte.

Die gesuchte Luftlinie ist der Öffnungswinkel im Bogenmaß multipliziert mit dem Erdradius.

EU-WAN

Da Backtracking in ungeahnten Zeitaufwand ausartet, wobei dieses die optimalste *(!)* Lösung finden würde, habe ich beschlossen, die Variante mit einem Algorithmus zu verwenden.

…, da Kreise auf Grund ihrer Form kaum dazu geeignet sind eine Fläche abzudecken.

Aber es gibt so viele Spezialfälle von Küstenlinien und Inseln, dass immer ein Spezialfall denkbar ist, in dem die Konstruktion auf fantastische Weise fehlschlägt.

Wir hatten Lösungsideen mit Algorithmen und Ähnlichen, kamen aber zu dem Schluss, dass ein einfaches Programm sinnvoller ist.

Lösungsansatz: Quadratur des Kreises

Obwohl ich eine Funktion nutze, die Ellipsen zeichnet, habe ich in den Kommentaren immer Kreis geschrieben. Das liegt daran, dass ich die Ellipse gleichseitig zeichnen lasse, sie also ein Kreis ist.

Zur Lösung dieser Aufgabe haben wir uns für einen einfachen und nicht allzu effektiven Algorithmus entschieden.

Teleskop

… Name des befleckten Bildes …

Außerdem könnte sich die Sternwarte auch einen neuen Chip in die Kamera einbauen lassen, denn mit so vielen Pixelfehlern ist das ein Garantiefall.