24. Bundeswettbewerb Informatik, 2. Runde

Lösungshinweise und Bewertungskriterien

Allgemeines

Es ist immer wieder bewundernswert, wieviel an Ideen und Wissen, an Fleiß und Durchhaltevermögen in den Einsendungen zur zweiten Runde eines Bundeswettbewerbs Informatik steckt. Um aber die Allerbesten für die Endrunde zu bestimmen, müssen wir Ihre Arbeit kritisch begutachten und hohe Anforderungen stellen. Von daher sind Punktabzüge die Regel und Bewertungen über das Soll (5 Punkte) hinaus die Ausnahme, insbesondere bei den schwierigen Aufgaben dieses Jahres. Lassen Sie sich davon nicht entmutigen!

Bewertungsbögen Kein Kreuz in einer Zeile bedeutet, dass die genannte Anforderung den Erwartungen entsprechend erfüllt wurde. Vermerkt wird also in der Regel nur, wenn davon abgewichen wurde – nach oben oder nach unten. Ein Kreuz in der Spalte „+“ bedeutet Zusatzpunkte, ein Kreuz unter „–“ bedeutet Minuspunkte für Fehlendes oder Unzulängliches. Die Schattierung eines Feldes bedeutet, dass die entsprechenden Plus- bzw. Minuspunkte in der Regel nicht vergeben wurden.

Termin der 2. Runde Die 2. Runde liegt zu großen Teilen parallel zum Abitur. Das ist sicher nicht ideal. In der zweiten Jahreshälfte läuft aber die 2. Runde des Mathewettbewerbs, dem wir keine Konkurrenz machen wollen. Also bleibt uns nur die erste Jahreshälfte. Und damit liegt der Abgabetermin der 2. Runde immer in der Zeit der Abiturtermine. Aber: Sie haben etwa vier Monate Bearbeitungszeit für die 2. Runde. Rechtzeitig mit der Bearbeitung der Aufgaben zu beginnen ist der beste Weg, Konflikte mit dem Abitur zu vermeiden.

Dokumentation Eine Dokumentation beginnt nicht mit: „Die Prozedur abc übergibt einen Zeiger p auf ein Feld xy, worauf die Funktion f ...“. Beschreiben Sie zunächst die (implementationsunabhängige) Idee, die Sie zur Lösung der jeweiligen Aufgabe entwickelt haben. Schildern Sie diese Lösungsidee erst grob und gehen dann darauf ein, wie Sie sie in ein abstraktes,

computertaugliches Modell (in Form von Algorithmen und Datenstrukturen) umgesetzt haben. Nutzen Sie dazu auch (halb-)formale Möglichkeiten zur Beschreibung Ihres Modells. Die Implementierung dieses Modells als Programm beschreiben Sie anschließend in der Programmdokumentation, die die wichtigsten Funktionen, Variablen, Klassen, Objekte etc. mit Bezug auf die Lösungsidee dokumentiert und angibt, wo diese Komponenten im Quellcode zu finden sind. Beachten Sie: Wer nicht in der Lage ist, Idee und Modell präzise zu formulieren, bekommt auch keine saubere Umsetzung in welche Programmiersprache auch immer hin.

Lösungshinweise Bei den folgenden Erläuterungen handelt es sich um Vorschläge, nicht um die einzigen Lösungswege, die wir gelten ließen. Wir akzeptieren in der Regel alle Ansätze, die die gestellte Aufgabe vernünftig lösen und entsprechend dokumentiert sind. Einige Dinge gibt es allerdings, die – unabhängig vom gewählten Lösungsweg – auf jeden Fall diskutiert werden müssen. Zu jeder Aufgabe gibt es deshalb einen Abschnitt, indem gesagt wird, worauf bei der Bewertung letztlich geachtet wurde.

1 Museum

Kunstmuseen mit verwinkelten Räumen waren schon immer der Albtraum eines jeden Wachmanns. Nur gut, dass mit Hilfe der Lösung der Aufgabe 1 zumindest überprüft werden kann, ob der gewählte Rundweg geeignet ist, alle Bereiche einzusehen und mögliche Einbrecher dingfest zu machen.

Die Aufgabe teilt sich in zwei Bereiche: Visualisierung und Sichtbarkeitsüberprüfung. Im Bereich der Visualisierung ist es erforderlich, vorgegebene Polygone zu zeichnen. Die Sichtbarkeitsprüfung stellt die eigentliche Herausforderung dar, da hier die Algorithmik zum Bestimmen von Flächenteilen eingesetzt werden muss.

1.1 Lösungsidee Visualisierung

Die Polygone, die die Außenmauer des Museums und die einzelnen abgetrennten Bereiche darstellen, sind als Punktliste gegeben. Nicht festgelegt ist die Skalierung des verwendeten Koordinatensystems. Daher muss zunächst die Bounding-Box (also die extremen Koordinatenwerte) aller Polygone bestimmt werden, um eine Verschiebung und Skalierung ins Bildschirmkoordinatensystem zu ermitteln. Eine Funktion zum Zeichnen von (gefüllten) Polygonen im Bildschirmkoordinatensystem ist in den meisten Grafikbibliotheken vorhanden und kann hier zur Darstellung verwendet werden. Für die Außenmauern und die Löcher wird die selbe Routine verwendet, die Löcher werden lediglich nach den Außenmauern in einer anderen Farbe gezeichnet. Der Weg des Nachtwächters besteht ebenfalls aus einer Menge von Punkten. Nach Transformation ins Bildschirmkoordinatensystem kann der Weg als Polyline dargestellt werden. Auch hierzu stellen die meisten Grafikbibliotheken fertige Funktionen zur Verfügung. Anstelle einer direkten Visualisierung am Bildschirm kann auch der Umweg über

ein Grafikformat und dessen Anzeige gewählt werden – mit Blick auf den Ausdruck von Beispielen ohnehin eine gute Idee.

1.2 Lösungsidee Sichtbarkeitsprüfung

Generell basiert die Sichtbarkeitsprüfung darauf, den gesamten zu überprüfenden Bereich in einzelne (konvexe) Regionen zu zerlegen, die für den Nachtwächter entweder vollständig sichtbar oder vollständig nicht einsehbar sind. Für jede dieser Regionen wird anschließend überprüft, ob sie sichtbar ist, die Region wird entsprechend eingefärbt. Zur Regioneneinteilung gibt es verschiedene Ansätze, die sich grob in zwei Klassen einteilen lassen: a) Einteilung in geometrische Teilobjekte (geometrischer Ansatz) oder b) Einteilung in Berechnungsatome (Pixel; diskretisierender Ansatz). Im folgenden wird zunächst der geometrische Ansatz beschrieben, anschließend wird auf diskretisierende Verfahren eingegangen. Die vielen weiteren Möglichkeiten der Sichtbarkeitsberechnung, bei denen sich geometrische Elemente mit Diskretisierungen insbesondere des Nachtwächterweges, aber auch der Museumsformen oder des Sichtkreises mischen, werden nicht näher behandelt.

1.2.1 Geometrischer Ansatz

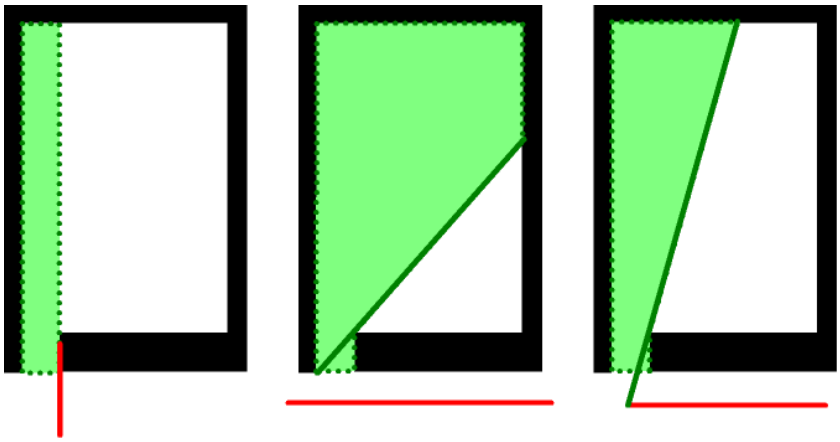


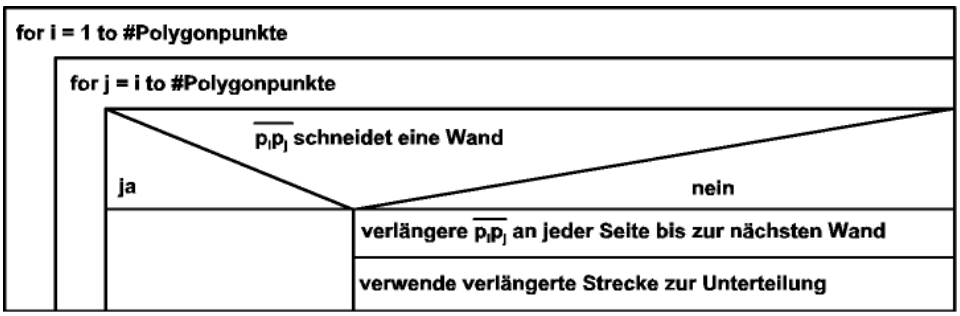
Abbildung 1: Wege des Nachtwächters und sichtbarer Bereich

Beim geometrischen Ansatz wird die Ausstellungsfläche des Museums zunächst in einzelne konvexe Polygonflächen zerlegt (partitioniert). Abbildung 1 zeigt exemplarisch einzelne Sichtbarkeitsverhältnisse für verschiedene Standpunkte und Bewegungssituationen des Nachtwächters. Dabei wird ein Innenpolygon (ein “Loch”) angenommen, das wie ein Raum mit Tür geformt ist; solche Extremfälle zeigen Schwierigkeiten besonders gut auf. An Hand der Bilder ist erkennbar, welche Strecken zur Partitionierung des Raumes (genauer: der Museumsfläche, die von diesem Innenpolygon beinahe eingeschlossen wird) benötigt werden. Die linke Abbildung zeigt, dass alle Wände und ihre Verlängerungen betrachtet werden müssen. Insbesondere

für die Raumaußenkanten sind diese Strecken erforderlich. In einigen Bereichen sind Wände und ihre Verlängerungen aber auch sichtblockierende Kanten. Die mittlere Abbildung zeigt, dass es nicht ausreichend ist, nur Wände und deren Verlängerung zu betrachten. Zum Teil kann auch schräg geschaut werden, der durch eine Wand verdeckte Bereich reduziert sich hier (durchgezogene Linie). Die rechte Abbildung schließlich verdeutlicht, dass auch die Wegpunkte des Nachtwächters betrachtet werden müssen. Geht der Nachtwächter nicht an der gesamten Türöffnung des Raumes vorbei, ist der sichtbare Bereich deutlich geringer. Zusammenfassend ergibt sich, dass sämtliche Polygonpunkte, egal, ob sie zu Außenwänden, Löchern oder zum Nachtwächterpfad gehören, berücksichtigt werden müssen.

Abbildung 2: Beispielhafte Partitionierung des Raumes nacheinander durch alle drei möglichen Arten von Strecken

Eine beispielhafte Zerlegung zeigt Abbildung 2. Die Unterteilung entsteht, indem alle möglichen Strecken zwischen jeweils zwei Polygonpunkten eingezeichnet und bis zur nächsten Wand verlängert werden. Nicht berücksichtigt sind Strecken, die bereits zwischen den beiden Punkten eine Wand schneiden. Im linken Bild sind lediglich die Verlängerungen von Wänden eingezeichnet, im mittleren Bild zusätzlich die Linien, die durch andere Kombinationen der Wandpunkte entstehen. Im rechten Bild sind alle Linien – auch die vom Endpunkt des Nachtwächter-Rundganges – eingezeichnet.



Struktogramm 1: Auswahl verwendeter Strecken

Dieses Verfahren ist formaler in Struktogramm 1 wiedergegeben. Dabei sind p_0 bis p_n die Polygonpunkte, #Polygonpunkte bedeutet die Anzahl der Polygonpunkte.

Im nächsten Schritt müssen die Schnittpunkte zwischen den so ermittelten Unterteilungslinien bestimmt werden (s. Abbildung 3). Die in Struktogramm 1 gefundenen Strecken werden an den Schnittpunkten in Teilstrecken unterteilt. Diese Teilstrecken sind später die Kanten der

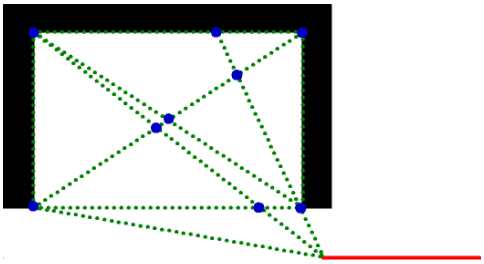
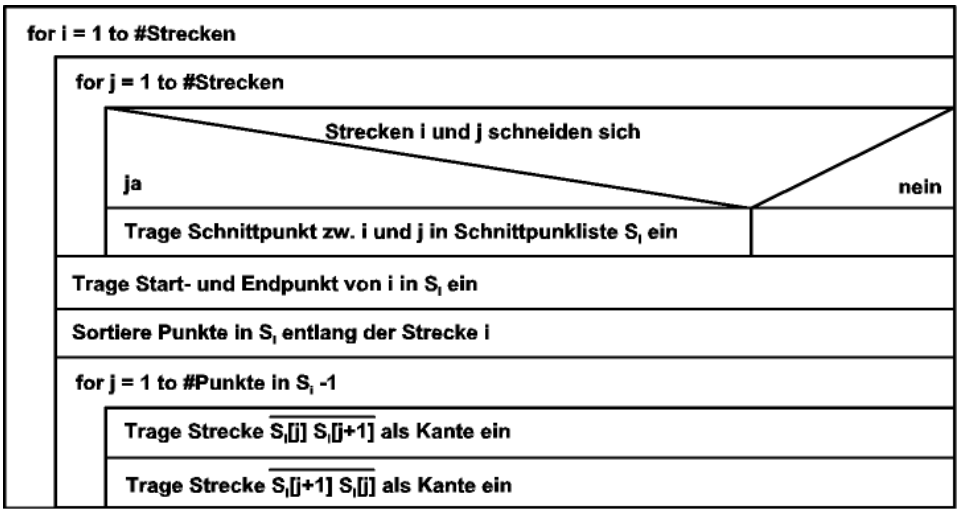


Abbildung 3: Schnittpunkte zwischen den Unterteilungslinien



Struktogramm 2: Unterteilung in einzelne Kanten

konvexen Polygonflächen, in die der Raum aufgeteilt wird. Struktogramm 2 gibt das dazu nötige Verfahren an.

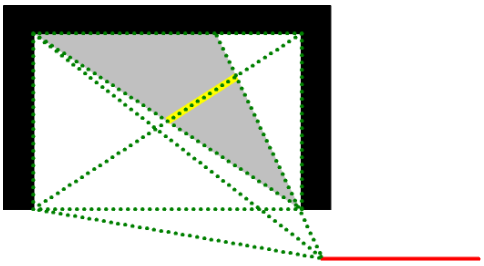


Abbildung 4: Jede Kante gehört zu zwei Polygonflächen

Aus der Kantenmenge muss nun eine Menge von Polygonflächen entstehen. Dabei hilft die Tatsache, dass jede Kante in genau zwei Polygonflächen vorkommt (s. Abbildung 4). Setzt man voraus, dass die Polygonflächen nur in einer Orientierung durchlaufen werden, so gehört jede gerichtete Kante (wie sie in Struktogramm 2 erzeugt wurde) nur zu genau einer Polygonfläche. Die Polygonflächen lassen sich nun erzeugen wie in Abbildung 5 illustriert und in Struktogramm 3 angegeben.

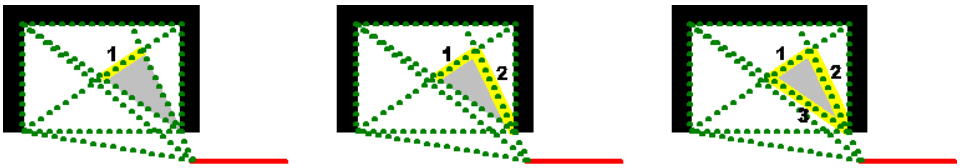


Abbildung 5: Erzeugung einer Polygonfläche durch Kantenumlauf

Jede durch Struktogramm 3 generierte Polygonfläche ist konvex. Da auch die im Museum liegenden Bereiche wie Toiletten, Garderobe etc. von Kanten umgeben waren, sind auch diese Bereiche durch Polygonflächen überdeckt. Dies stört aber nicht weiter, da diese Flächen für den Nachtwächter später von jedem Standpunkt aus hinter einer Wand liegen und richtigerweise als nicht sichtbar klassifiziert werden.

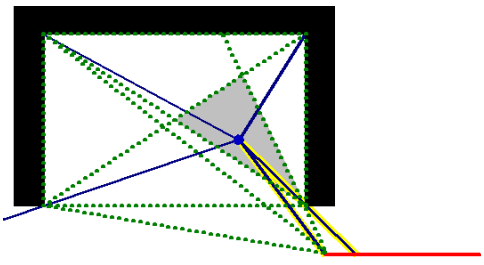
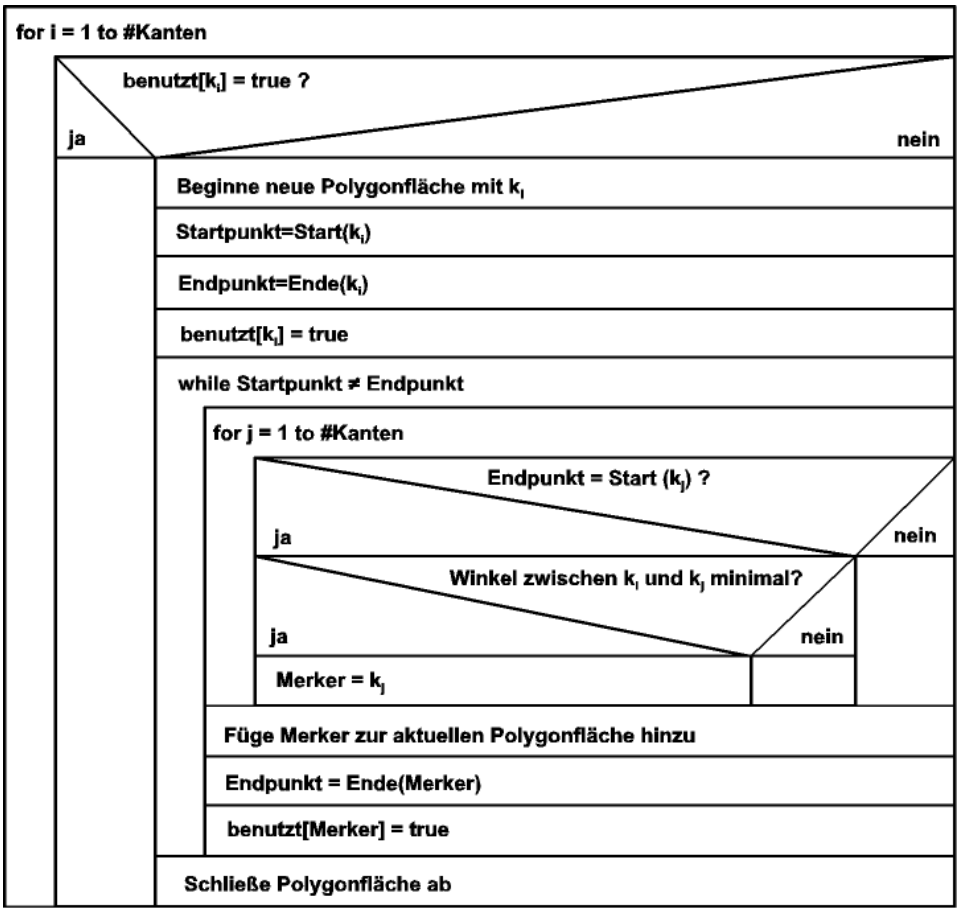
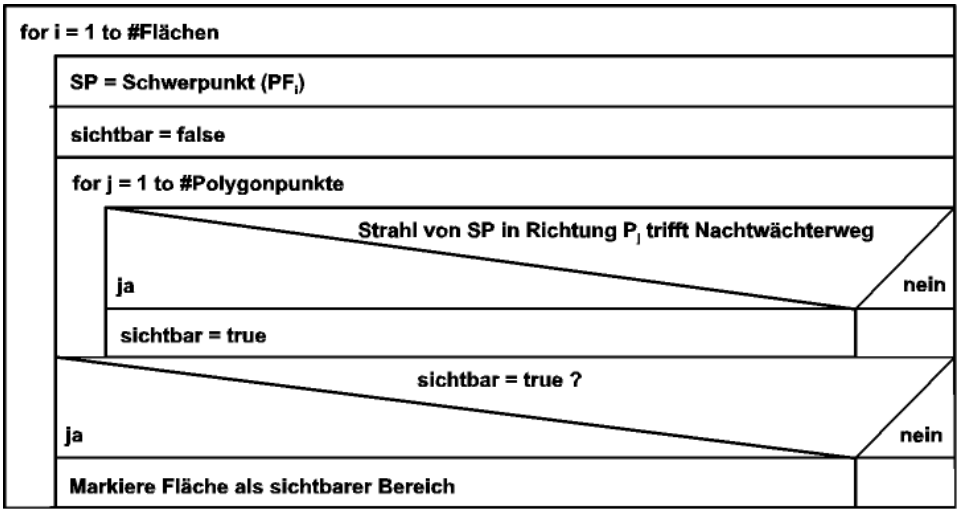


Abbildung 6: Zwei Strahlen erreichen den Weg des Nachtwächters, die Fläche ist sichtbar.



Struktogramm 3: Generierung der Polygonflächen



Struktogramm 4: Sichtbarkeitsprüfung für eine Fläche

Für alle soweit bestimmten Flächen gilt, dass sie entweder vollständig sichtbar sind oder vollständig nicht im Blickfeld des Nachtwächters liegen. Es reicht also aus, einen ausgezeichneten Punkt im Inneren einer jeden Fläche auf Sichtbarkeit zu prüfen. Dies sei im folgenden der Schwerpunkt der Fläche, der bei einer konvexen Polygonfläche immer innerhalb der Fläche liegt. Von diesem Punkt aus werden die Strahlen zu sämtlichen anderen Punkten der ursprünglichen Polygone betrachtet. Berührt ein solcher Strahl den Weg des Nachtwächters, bevor er eine Wand schneidet, ist der Bereich sichtbar. Abbildung 6 zeigt hierzu ein Beispiel. Den Algorithmus für die Sichtbarkeitsprüfung gibt Struktogramm 4 an.

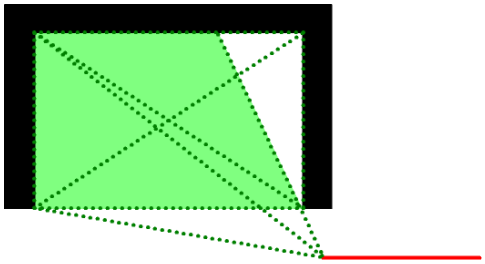


Abbildung 7: Sichtbarer Bereich im Beispiel

Die Ergebnisse von Struktogramm 4 können als geschlossene Polygonflächen eingezeichnet werden. Es ergibt sich ein Diagramm der sichtbaren Fläche; ein Beispiel zeigt Abbildung 7.

1.2.2 Diskretisierender Ansatz

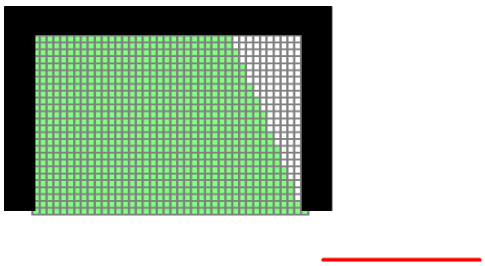


Abbildung 8: Sichtbarer Bereich im diskretisierten Fall

In der Klasse der diskretisierenden Verfahren gibt es verschiedene Abstufungen. So wird zum Teil nur die Fläche diskretisiert, zum Teil auch der Weg des Nachtwächters; sogar die Rundumsicht kann diskretisiert werden, etwa in 360 verschiedene Sichtstrahlen. Generell kann man sagen, dass der algorithmische Aufwand eines diskretisierenden Verfahrens erheblich unter dem des vorgestellten geometrischen Verfahrens liegt. Diskretisiert man lediglich den Freiraum in einzelne Pixel, so ergeben sich unmittelbar konvexe Regionen, für die man jeweils eine Sichtbarkeitsprüfung ausführen kann (vgl. geometrisches Verfahren, Struktogramm 4). Das Ergebnis ist in Abbildung 8 dargestellt.

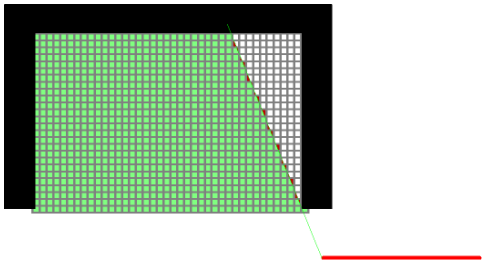


Abbildung 9: Fehlerhaft klassifizierte Bereiche bei einer Diskretisierung.

Bei diesem Vorgehen ergeben sich zwangsläufig Rundungsfehler, wie sie in Abbildung 9 dargestellt sind. Die auftretenden Rundungsfehler sind dabei abhängig von der gewählten Auflösung der Diskretisierung. Je feiner das Raster aufgelöst ist, desto geringer sind auch die auftretenden Rundungsfehler. Entscheidend für die Lösung der Aufgabe ist, dass die Auflösung immer hoch genug gewählt ist, um alle relevanten Details der Umgebung noch hinreichend genau zu erfassen. Um genauer zu sein, sollte die Rasterauflösung nach dem Abtasttheorem mindestens doppelt so hoch sein, wie der minimale Abstand zwischen zwei beliebigen Polygonpunkten in der Umgebung und im Weg des Nachtwächters. Wird dies nicht beachtet, so können Details der Umgebung beim Rastern verschwinden und Aussagen über Teilbereiche sind fehlerhaft.

1.3 Bewertung

Die Aufgabenteile 1 und 2 werfen keine besonderen Probleme auf. Das Einlesen der Formen bzw. eines Rundgangs sollte uneingeschränkt funktionieren, ebenso deren Visualisierung. Bei der Visualisierung ist es schön, wenn die inneren Polygone vom äußeren des Museums abgehoben werden. Das Gesamtbild des Museums darf durchaus unorthodox sein, wenn dies in der Einsendung begründet wird (ein Restaurant kann z.B. eine Außenterrasse haben, so dass sein Polygon nicht vollständig innerhalb des Museums liegt). Das Programm sollte aber sicherstellen, dass der Rundgang oder Weg des Nachtwächters innerhalb des Museums und nicht durch die Löcher verläuft.

Der entscheidende Teil ist die Sichtbarkeitsberechnung. Wie schon geschildert, gibt es für diese Aufgabe verschiedene Lösungswege mit stark unterschiedlichem Aufwand. Dies stellt auch ein zentrales Element in der Bewertung dar. Angesichts der Komplexität der geometrischen Lösung wird diese nicht erwartet. Daher wird eine Lösung, welche die Fläche, den Weg des Nachtwächters und/oder den Sichtkreis diskretisiert (es gibt hier viele „Zwischenlösungen“), nicht grundsätzlich beanstandet. Punktabzüge gibt es aber, wenn die Sichtbarkeitsberechnung unabhängig vom gewählten Lösungsweg fehlerhaft ist. Das Wagnis einer voll geometrischen Lösung wird auf jeden Fall mit Bonuspunkten belohnt.

Je nach Lösungsweg ergeben sich verschiedene weitere wahrscheinliche Fehlerquellen. Im diskreten Fall wird zum einen eine Einsicht in die prinzipbedingten Unzulänglichkeiten einer

solchen Lösung erwartet. Es sollte erkannt worden sein, dass der Auflösung eine entscheidende Bedeutung zukommt und dass es auch bei einer hohen Auflösung immer noch zu Rundungsfehlern kommt. Zum anderen sollte die Auflösung in Abhängigkeit von den Eingabedaten geeignet gewählt sein. Idealerweise bestimmt das Programm die Auflösung anhand der Eingangsdaten automatisch. Eine fest gewählte Auflösung sollte zumindest den Daten gerecht werden; die Auflösung sollte also mindestens doppelt so hoch sein wie der geringste Abstand zwischen zwei Polyederpunkten. Eine solche oder ähnliche plausible Begründung für eine fest gewählte Auflösung wird erwartet.

Im geometrischen Fall wird es vorkommen, dass nicht alle drei Möglichkeiten der Trennungslinien berücksichtigt werden. Fehler dieser Art führen zu Bereichen, die nur teilweise sichtbar sind. Ein weiterer wahrscheinlicher Fehler ist eine fehlerhafte Segmentierung nach korrekter Unterteilung. An dieser Stelle muss man nicht unbedingt die oben beschriebene Zerlegung in Polygonflächen realisieren. Auch eine Triangulierung der Flächen ist möglich. Sollten hier jedoch konkave Flächen als Ergebnis möglich sein, deutet dies auf einen Fehler hin. Zum Schluss muss noch auf den Algorithmus geachtet werden, der zwei Linien auf Schnitt oder Berührung testet. Hier muss zwischen Berührung und Schnitt unterschieden sein! Ein Strahl, der eine Wand nur berührt, geht genauso wie ein Strahl, der keine Wand trifft, weiter. Hingegen ist ein Strahl, der den Weg des Nachtwächters auch nur berührt gleichbedeutend mit einem Strahl, der den Weg schneidet. Beides bedeutet Sichtbarkeit.

Von der Geschwindigkeit der Programme darf erwartet werden, dass die Berechnung für einen einfachen Pfad im Beispielmuseum nicht länger als 10 Minuten dauert. Berechnungsergebnisse müssen noch geeignet angezeigt werden; in einer geeigneten Ergebnisvisualisierung sind das Museum, der Rundgang und der sichtbare Bereich zu erkennen. Zumindest in kleineren Beispielen ist es sinnvoll, Ergebnisse von Berechnungsschritten anzuzeigen, ähnlich wie in den Abbildungen dieses Abschnittes. Wie immer werden mindestens drei verschiedene Beispiele erwartet, die die Funktionsweise des Programms geeignet veranschaulichen.

Als Erweiterungen sind eine automatische Pfadgenerierung und ein eingeschränktes Sichtfeld des Nachtwächters naheliegend. Bei der automatischen Pfadgenerierung muss zum einen auf die Korrektheit (sind wirklich alle Bereiche sichtbar) und zum anderen auf die Effizienz (keine unnützen Schnörkel im Weg) geachtet werden. Ein eingeschränktes Sichtfeld des Wächters ist eine eher geringfügige Erweiterung, da sie bei allen beschriebenen Ansätzen keine große Herausforderung darstellt. Selbstverständlich sind auch weitere Erweiterungen möglich.

2 Trimm-Dich-Pfade

2.1 Lösungsidee

2.1.1 Grundsätzliche Überlegungen

Die entscheidende Frage der Aufgabenstellung lautet im Kern „Wie viele verschiedene Möglichkeiten?“; die naive Art, auf solche Fragen zu antworten, ist das Abzählen der Möglichkeiten und Ausschluss der Duplikate. Doch Möglichkeiten gibt es zu viele: jedes Gerät könnte genommen werden oder nicht, und das macht bei N Geräten 2^N Möglichkeiten, was schon für gar nicht mal allzu lange Trimm-Dich-Pfade so viele sind, dass das Auf- und Abzählen zu lange dauert. Es gilt also, Anzahlen von Möglichkeiten zu berechnen, ohne die Möglichkeiten aufzählen zu müssen. Die richtige Idee dazu lautet: Beim Gang über den Trimm-Dich-Pfad kann ich ein neues Gerät vom Typ t zu allen bisherigen Übungen dazu nehmen oder nicht: die Anzahl der Übungen verdoppelt sich also zunächst. Doch alle Übungen, die beim vorigen Gerät des gleichen Typs durch Hinzunahme dieses Geräts entstanden, entstehen nun noch einmal; hätte ich mir damals diese Zahl gemerkt, könnte ich sie nun einfach abziehen. Das klingt vernünftig – doch ist es auch korrekt?

Sicherheitshalber betrachten wir das Problem auch einmal etwas formaler. Zunächst einmal lässt sich beobachten, dass man sich an einer bestimmten Trainingsstation entscheiden kann, entweder sein Training und damit den Trimm-Dich-Pfad zu beenden oder seine Übungen an einem der möglichen Geräte fortzuführen. Ein nächstes mögliches Gerät lässt sich durch genau ein Element der Menge der vorkommenden Buchstaben (im Folgenden als Σ bezeichnet) angeben. Dabei ist zu beachten, dass es keine Rolle spielt, wo sich dieses nächste Gerät konkret befindet, solange es hinter dem aktuellen Gerät vorkommt. Deshalb kann man immer den nächsten entsprechenden Buchstaben im Text auswählen und umgeht damit gleichzeitig das Problem, dass bestimmte Sequenzen doppelt vorkommen (es gibt immer nur genau ein nächstes Gerät eines bestimmten Typs).

Wenn $P_n(c) : \Sigma \to \mathbb{Z}$ nun die Position des nächsten Geräts vom Typ c hinter Position n im Pfad (also im Text) angibt und den Wert -1 hat, falls kein solches Element existiert, dann lässt sich die Anzahl $T(m)$ der möglichen Pfade, die an Station m starten, durch folgende Formel ermitteln:

$$\begin{aligned} T(-1) &= 0 \\ T(m) &= 1 + \sum_{c \in \Sigma} T(P_m(c)) \end{aligned} \tag{1}$$

Da der Pfad natürlich mit jedem Gerät beginnen und außerdem auch komplett leer sein kann, lautet die Formel für die gesamte Anzahl an Möglichkeiten:

$$1 + \sum_{c \in \Sigma} T(P_0(c)) \tag{2}$$

Der Wert von $P_m(c)$ ist immer strikt größer als m oder gleich -1, denn das nächste Gerät vom Typ c steht entweder weiter hinten im Pfad oder existiert nicht. Daher ist garantiert, dass die Berechnung der Formel terminiert.

Weil man nach der Anzahl unterschiedlicher Sequenzen in einer Zeichenkette sucht, sollte das Ergebnis unabhängig davon sein, ob die Zeichenkette vorwärts oder rückwärts verarbeitet wird. Das ist deshalb bemerkenswert, weil der Algorithmus dann noch nicht alle Daten haben muss, um die Berechnung zu beginnen (man sagt, der Algorithmus kann "online" arbeiten).

2.1.2 Mögliche Berechnungsverfahren

Der naive Ansatz an dieser Stelle wäre, einfach zwei Funktionen P und T zu schreiben, die genau der oben beschriebenen Definition folgen. Man kann allerdings schnell erkennen, dass dies zu exponentiell mit der Länge des Textes ansteigenden Laufzeiten führt und deshalb auf gar keinen Fall für den Kafka-Text funktionieren wird. Wäre dies der einzige Algorithmus, dann wäre das Problem „nicht handhabar" (engl. untractable), d.h. unberechenbar im Sinne von „kann nicht zu Lebzeiten berechnet werden". Deshalb muss etwas mehr über das Problem nachgedacht werden.

Hier kommt das Prinzip der dynamischen Programmierung ins Spiel: Mit dynamischer Programmierung wird die Eigenschaft ausgenutzt, dass Ergebnisse von Unterproblemen zwischengespeichert und wiederverwendet werden können. In diesem Fall bedeutet das konkret: Es gibt nur $N+1$ mögliche Werte der Funktion T, wenn N die Länge des Textes ist, wobei einer dieser Werte $T(-1) = 0$ ist. Die anderen N verschiedenen Werte kann man natürlich in einem Array abspeichern, damit sie nicht mehrmals neu berechnet werden müssen. Außerdem werden zur Berechnung von $T(m)$ nur Werte von $T(k)$ für $k > m$ (oder $k = -1$) verwendet. Das bedeutet, dass das T Array also von hinten nach vorne gefüllt und jedesmal auf bereits vorhandene Werte zugegriffen werden kann.

Die Berechnung eines T-Wertes ist jetzt in $|\Sigma|$ Schritten möglich, also nur abhängig von der Größe des Alphabets, wenn man davon ausgeht, dass die Werte von P in konstanter Zeit berechnet werden können. Damit ist die Gesamtlaufzeit des Algorithmus proportional zu $N \cdot |\Sigma|$, was sogar schnell genug für den Kafka-Text ist. Der Speicherverbrauch ist nun proportional zu N.

Es gibt allerdings noch eine weitere Verbesserung, die sowohl die Laufzeit- als auch die Speicherkomplexität verbessert und deshalb auf jeden Fall erwähnenswert ist.

Jedesmal, wenn T berechnet wird, werden genau $|\Sigma|$ weitere Werte der T-Funktion benötigt:

$$T(m) = 1 + \sum_{c \in \Sigma} T(P_m(c))$$

Ruft man sich nocheinmal die Bedeutung der Funktion $P_m(c)$ ins Gedächtnis, nämlich die Position des nächsten Trainingsgeräts vom Typ c hinter Position m, so ist klar, dass sich die

benötigten Werte der T-Funktion nur für das Gerät g ändern, was gerade zuvor betrachtet wurde, also das Gerät an der Stelle $m+1$. Das nächste Vorkommnis von Gerät g hinter der Stelle m ist nämlich $m+1$, war aber vorher (bei Berechnung von rechts) $P_{m+1}(g)$.

Da dies die einzige Änderung in der ganzen Summe ist, kann $T(m)$ aus $T(m+1)$ berechnet werden:

$$\begin{aligned} T(m) &= T(m+1) - T(P_{m+1}(g)) + T(m+1) \\ &= 2 \cdot T(m+1) - T(P_{m+1}(g)) \end{aligned} \tag{3}$$

Mit diesem Schritt ist jetzt die Berechnung eines Wertes von T in konstanter Zeit möglich, sodass nur noch eine Verbesserung der Speicherkomplexität bleibt: es werden immer nur $|\Sigma|$ Werte von T gleichzeitig im Speicher benötigt. Es macht jetzt keinen Sinn mehr, die Werte von T bzgl. der dazugehörigen Position im Text anzugeben, sondern vielmehr bzgl. des Geräts, dass sich an dieser Stelle befindet. An die Stelle von $T(m+1)$ tritt $T(g)$, und $T(P_{m+1}(g))$ wird durch V, den vorhergehenden Wert von $T(g)$ ersetzt. Dieser vorhergehende Wert von $T(g)$ muss also in einer weiteren Variable gespeichert werden, weil $T(g)$ ja schon den neuen Wert beinhaltet.

Alle T-Werte werden mit 0 initialisiert, denn es gibt in einem leeren Pfad keine Parcours, die mit irgendeinem Buchstaben beginnen. Wenn q also das Gerät an der aktuellen Textposition m und g das Gerät an der Textposition $m+1$ ist, dann ist

$$T(q) = 2 \cdot T(g) - V \tag{4}$$

Der Algorithmus hat nun eine Laufzeitkomplexität von $\Theta(N)$ (wenn arithmetische Operationen mit konstanter Laufzeit angenommen werden) und eine Speicherkomplexität von $\Theta(|\Sigma|)$ – und entspricht der Überlegung, die ganz am Anfang schon intuitiv angestellt wurde.

2.1.3 Große Zahlen

Ein weiterer Bestandteil dieser Aufgabe ist es, mit großen Zahlen umzugehen. Die unten angegebene Implementation verwendet dafür ein fertiges Modul. In Bearbeitungen, die Additions- bzw. Multiplikationsarithmetik eigenständig realisieren, sollte dies natürlich auch besprochen werden.

2.2 Programmdokumentation

Diese Lösung verwendet Java, wo die Klasse **BigInteger** die arithmetischen Operationen übernimmt. Die Implementation des oben beschriebenen Algorithmus ist dann relativ übersichtlich. Wie bereits in der Lösungsidee beschrieben, kann der Text in beiden Richtungen verarbeitet werden; diese Implementation arbeitet von links nach rechts. Wichtig ist, dass die

Funktionswerte für T alle mit 0 initialisiert werden (es gibt keine Pfade vor dem ersten Gerät, der leere Pfad wird erst beim Aufsummieren berücksichtigt). $T(g)$ wird in **vorher** gespeichert und V in **letztes**. Damit **T[0]** korrekt auf 1 gesetzt wird, werden **vorher** und **letztes** mit 1 initialisiert.

Listing 1: Quelltext

```
private static BigInteger trimm(String text){
  // T[c] speichert die Anzahl der Moeglichkeiten
  // von dem letzten Zeichen "c" aus
  BigInteger []T = new BigInteger [256];

  // Initialisierung
  for (int i = 0; i < T.length; i++)
    T[i] = BigInteger.ZERO;

  // dynamische Programmierung ueber
  // die Anzahl der Moeglichkeiten
  BigInteger letztes = BigInteger.ONE,
             vorher = BigInteger.ONE;
  for (int pos = 0; pos < text.length(); pos++){
    vorher = vorher.subtract(letztes).add(vorher);
    letztes = T[text.charAt(pos)];
    T[text.charAt(pos)] = vorher;
  }

  // Der Pfad kann mit jedem Buchstaben des Textes
  // beginnen, daher alle Moeglichkeiten aufsummieren.
  BigInteger sum=BigInteger.ZERO;
  for (int i = 0; i < T.length; i++)
    sum = sum.add(T[i]);
  // 1 addieren fuer die leere Uebung
  return sum.add(BigInteger.ONE);
}
```

2.3 Programmablaufdokumentation

Für die Programmablaufdokumentation sollten mindestens die drei gegebenen Beispiele sowie der Kafkatext bearbeitet sein. Die Aufgabenstellung fragt aber auch, wie man sich davon überzeugen kann, dass das Programm korrekt funktioniert. Neben einer überzeugenden Lösungsidee und eventuell einer formalen Begründung für deren Korrektheit (also z.B. eine formale Herleitung wie in Abschnitt 2.1) gehören dazu auch einige Testfälle, deren Antworten bekannt sind, weil sie von Hand berechnet werden können. Beispielsweise gibt es für einen Parcours, der nur aus einer Sorte von Trainingsgeräten besteht, genau $N+1$ verschie-

dene Möglichkeiten; für einen Parcours, der aus N verschiedenen Geräten besteht, gibt es 2^N Möglichkeiten.

Es gibt noch weitere Testmöglichkeiten: ein Beispiel ist der Ergebnisvergleich mit einem „sicheren“ Verfahren, z.B. einer Lösungsbestimmung durch vollständige Suche, wobei dies nur für kleine Eingabefälle möglich ist und eine fehlerfreie Implementation voraussetzt.

2.3.1 Beispiel 1

Listing 2: Eingabe

```
RTRT
```

Listing 3: Ausgabe

```
12
```

2.3.2 Beispiel 2

Listing 4: Eingabe

```
RTRBRTBTTRRBTR
```

Listing 5: Ausgabe

```
3868
```

2.3.3 Beispiel 3

Listing 6: Eingabe

```
RTBRSTBTRBTRRTSSSSSRRTTSTRTTSBBRTBRSBRRBBBSRBRBB
```

Listing 7: Ausgabe

```
307573196245
```

2.3.4 Kafka

Listing 8: Ausgabe

```
213265550539852359962451036087185836707 0
678401950108730781721380951824775328974 8
013314860294155059 79 ... 887617112250276
368163379929808743290477742497190975058 3
386754043415721536620121367007419693193 8
66488
```

2.3.5 Beispiel 4

Listing 9: Eingabe

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Listing 10: Ausgabe

```
41
```

2.3.6 Beispiel 5

Listing 11: Eingabe

```
ABABABABABABABABABABABABABABABABABABAB
```

Listing 12: Ausgabe

```
433494436
```

2.3.7 Beispiel 6

Listing 13: Eingabe

```
RSTUVWXYZ
```

Listing 14: Ausgabe

```
512
```

2.3.8 Beispiel 7

Listing 15: Eingabe

```
AEEEEEEEEEEEEEEEBCCCCCD
```

Listing 16: Ausgabe

```
768
```

2.3.9 Beispiel 8

Ein Beispiel, das per Hand prüfbar ist.

Listing 17: Eingabe

```
ABAC
```

Listing 18: Ausgabe

```
14
```

Ergibt die Zeichenketten:

$$\{\varepsilon, A, B, C, AB, AC, BA, AA, BC, ABA, ABC, AAC, BAC, ABAC\}$$

2.3.10 Beispiel 9

Es ist auch wichtig, dass Grenzfälle getestet werden wie z.B. eine leere Eingabe:

Listing 19: Ausgabe

```
1
```

2.4 Bewertung

Das Wichtigste ist, dass der auf dem Prinzip der Dynamischen Programmierung beruhende Lösungsansatz für diese Aufgabe erkannt wurde. Akzeptabel ist eine Lösung, die die Laufzeit $O(N \cdot |\Sigma|)$ und den Speicherverbrauch $\mathcal{S}(N \cdot |\Sigma|)$ hat, solange sie schnell genug ist, um den Text von Kafka zu verarbeiten und natürlich auch die richtige Antwort liefert. Auf jeden Fall muss die Laufzeit- und Speicherkomplexität des Algorithmus besprochen werden.

Da es hier nicht sehr einfach ist, sich wirklich sinnvolle Erweiterungen zu überlegen, wird eine Lösung, die wirklich eine Laufzeit von $O(N)$ und einen Speicherverbrauch von $\mathcal{S}(|\Sigma|)$ hat, mit Bonuspunkten belohnt.

Wie gesehen gibt es z.B. in Java die Klasse **BigInteger**, die natürlich einen Teil des Problems schon löst. Falls eine solche Klasse Bestandteil einer Sprache ist, ist es legitim, diese zu verwenden. Implementationen eigener Large-Integer Funktionen sollten korrekt sein; für besonders vollständige und gute können Extrapunkte vergeben werden.

Schließlich muss auch auf den letzten Satz der Aufgabenstellung eingegangen werden: Wie kann man sich davon überzeugen, dass das Programm korrekt arbeitet? Wenn dieser Aufgabenteil ausreichend beantwortet worden ist, dann sollte das Testen des Programms (durch die Bewerter) überflüssig sein. Zum einen muss natürlich die Lösungsidee verständlich und korrekt sein, aber besonders wichtig sind hier Testfälle, die, ähnlich wie einige der obigen, per Hand generiert und verifiziert worden sind; denn selbst wenn die Lösungsidee noch so korrekt ist, ist es noch lange nicht offensichtlich, dass die Implementation stimmt. Zu den geforderten vier Beispielen sollten mindestens noch weitere drei kommen. Eine formale Begründung für die Korrektheit des Programms wird belohnt, ebenso eine besonders ausführliche Diskussion und die Umsetzung weiterer Testmöglichkeiten.

Als Erweiterungen sind vor allem Lösungen ähnlicher Probleme denkbar, wie etwa die Bestimmung der Anzahl verschiedener *zusammenhängender* Übungen auf einem Trimm-Dich-Pfad. Die Implementation von Testprozeduren stellt alleine noch keine Erweiterung dar.

3 Wer wird Weltmeister?

Die erste Schwierigkeit der Aufgabe „Wer wird Weltmeister?“ besteht sicherlich in der Identifizierung des sportlichen Wettbewerbs, um den es sich handelt. Die Erwähnung von „Toren“ scheint auf ein Ballspiel zu verweisen, allerdings tragen Tore auch bei Abfahrtsrennen im Skisport eine bedeutende Rolle. Dennoch wird in der folgenden Diskussion häufig die Hypothese verwendet, es handele sich bei dem Wettbewerb um ein Fußballturnier. Selbstverständlich sind jedoch auch andere Interpretationen als gültige Lösungen denkbar – schließlich können mit dem für die Spezifikation des Turnierablaufs verwendeten Format auch Turniere ganz anderer Sportarten mit deutlich anderen Ablaufstrukturen angegeben werden.

Im ersten der folgenden Abschnitte werden die verschiedenen Möglichkeiten für eine Modellierung des Systems „Fußballspiel“ sowie die Alternativen für die Verifikation dieser Modelle behandelt. Darauf aufbauend wird in Abschnitt 3.2 ein Beispiel-Modell entwickelt. Schließlich wird im letzten Teil auf die Bewertung der einzelnen Teilaufgaben eingegangen.

3.1 Lösungsidee allgemein: Modellierung

Allgemein lässt sich die Modellierung des Fußballturniers im vorliegenden Rahmen in drei Komponenten gliedern:

- Definition der Anzahl und Bedeutung der Parameter, die die Fähigkeit einer Mannschaft beschreiben.

- Berechnungsvorschrift für die Ergebnisfunktion, die aus zwei Parametersätzen zweier Mannschaften ein Ergebnis berechnet.
- Vorschrift zur Bestimmung der Werte der Parametersätze aller Mannschaften.

Von wesentlicher Bedeutung ist dabei der Nachweis der Korrektheit des entwickelten Modells.

3.1.1 Parametersätze der Mannschaften

Jeder Mannschaft wird ein Parametersatz zugeordnet, der ihre Fähigkeiten beschreibt. Der Parametersatz kann dabei aus einem oder mehreren Einzelparametern bestehen. Ihre Bedeutung erhalten die Parameter durch die Rechenvorschrift der Ergebnisfunktion.

Im einfachsten Fall ist der Parametersatz *statisch* und verändert sich nicht im Laufe des Turniers. Eine komplexere Modellierung könnte jedoch durchaus veränderliche Parametersätze vorsehen, die dann z. B. Effekte wie Kondition, Motivation oder gesperrte Spieler abbilden könnten.

3.1.2 Ergebnisfunktion

Die Ergebnisfunktion lässt sich formal wie folgt darstellen:

$$E = (E_A, E_B) = S(M_A, M_B) \tag{5}$$

Dabei sind M_A und M_B die Parametersätze der Mannschaften A und B. Das Ergebnis E besteht im einfachsten Fall aus der Anzahl der Tore beider Mannschaften E_A und E_B.[1] Die Ausgabe von *gewonnen/verloren/unentschieden* ist alleine nicht ausreichend, da die Toranzahlen für die Bestimmung der Gruppenrangliste notwendig sind.

Es handelt sich dabei nicht um eine Funktion im streng mathematischen Sinne, schließlich hängt der Funktionswert auch vom Zufall ab. Ein solches Konstrukt wird durch eine Verteilungsfunktion beschrieben, die angibt, welches Torverhältnis wie wahrscheinlich ist. Die Erfahrung zeigt dabei, dass sehr hohe Toranzahlen sehr selten vorkommen, so dass die modellierte Verteilungsfunktion in Richtung hoher Torzahlen sehr schnell abfallen sollte.

Der Erwartungswert der Ergebnisfunktion, also das durchschnittliche Ergebnis bei sehr häufiger Berechnung der Funktion, kann jedoch durchaus als mathematische Funktion betrachtet werden (S_i bezeichnet eine einzelne Berechnung):

$$\langle E \rangle = \bar{S}(M_A, M_B) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} S_i(M_A, M_B) \tag{6}$$

[1] Erweiterungen sind denkbar: Gelbe und Rote Karten, Verletzungen von Spielern, etc.

Offensichtlich muss der Erwartungswert der Ergebnisfunktion symmetrisch in dem Sinne sein, dass eine Vertauschung der Mannschaften A und B zu einer Vertauschung der durchschnittlichen Toranzahlen für beide Mannschaften führt.

3.1.3 Bestimmung der Modellparameter

Zur Vervollständigung des Modells müssen noch die konkreten Werte der Parametersätze M_i aller Mannschaften bestimmt werden. Dabei sind generell zwei Ansätze möglich: Entweder werden die Werte aller Einzelparameter aller Mannschaften *direkt* bestimmt, oder es wird eine *Zuordnungsfunktion* entwickelt (die selbst wiederum einige freie Parameter besitzt), die aus der aktuellen Weltrangliste alle Parametersätze M_i konstruiert.

Beide Verfahren benötigen ein quantitatives Maß zur Bestimmung der Güte eines konkreten Parametersatzes (s. nächster Abschnitt). Existiert dieses, liegt ein Minimierungsproblem vor, das mit Hilfe von Standardverfahren gelöst werden kann. Der wesentliche Unterschied ist jetzt, dass für den Fall der direkten Bestimmung aller Einzelparameter der abzusuchende Parameterraum hochdimensional ist (mindestens 32 Dimensionen bei ebensovielen teilnehmenden Mannschaften), was große Herausforderungen an den Minimierungsalgorithmus stellt und zu langen Rechenzeiten führt. Anders für die Bestimmung mittels Zuordnung: Für die Parametrisierung der Zuordnung sollten einige wenige Parameter ausreichend sein, die sich dann mit weniger Aufwand bestimmen lassen.

3.1.4 Verifikation des Modells

Bei der Entwicklung des Modells bestehen zunächst einmal sehr große Freiheiten und es sind viele unterschiedliche Ansätze denkbar. *Seine Berechtigung erhält ein Modell jedoch ausschließlich dadurch, dass es die Wirklichkeit gut beschreibt.* Deswegen kommt der Überprüfung des gewählten Modells eine zentrale Bedeutung zu.

Zur Verifikation eines Modells mit statischen Parametersätzen bietet es sich an, die Weltrangliste heranzuziehen. Lässt man jede Mannschaft gegen jede andere antreten und berechnet daraus eine neue „Weltrangliste“, so sollte bei einem perfekten Modell genau die tatsächliche Rangliste herauskommen. In der Praxis gestaltet sich das nicht ganz so einfach, da die Berechnung der Liste recht kompliziert und nicht besonders exakt definiert ist, allerdings lässt sich mit einigen vereinfachenden Annahmen ein tauglicher Berechnungsalgorithmus für die Rangliste finden. Dadurch wird die Auswertung praktikabel und zumindest eine ungefähre Validierung des Modells ermöglicht.

Ein Modell mit variablen Parametersätzen vollständig zu verifizieren ist hingegen kaum sinnvoll möglich. Hier müsste auf Aufzeichnungen alter Weltmeisterschaften und der jeweils vorherigen Weltrangliste zurückgegriffen werden, welche nicht ohne weiteres zugänglich sind. Durch die geringe Anzahl der „Beispieldaten“ (Die Weltrangliste existiert erst seit 1993 und wurde zudem noch 1999 überarbeitet.) wäre die Aussagekraft ohnehin gering. Ein möglicher

Ansatz an dieser Stelle bestünde allerdings in der Verifikation des statischen Anteils des Parametersatzes und schlüssiger Argumentation zur Begründung des nichtstatischen Teils.

3.2 Lösungsidee konkret: Modellentwicklung anhand eines Beispiels

Es wird nun eine Beispiel-Lösung für Teil 1 der Aufgabenstellung entwickelt. Dabei wird der Schwerpunkt auf die generelle Methodik gelegt und das Modell selbst bewusst einfach gehalten.

Angestrebt wird ein Modell, das möglichst wenige freie Parameter enthält und an der realen Weltrangliste verifiziert werden kann. Folglich kommt nur die Verwendung statischer Parametersätze M_i in Frage. Die Bestimmung der Parametersätze M_i erfolgt dabei über eine Zuordung f aus den Weltranglistenpunkten P_i der Mannschaft i (vgl. Abschnitte 3.1.3 und 3.1.4):

$$M_i = f(P_i) \tag{7}$$

Wenn nun der Parametersatz M_i als Funktion von P_i bestimmt ist, kann die Anzahl der unabhängigen Parameter des Parameter-„Satzes“ nicht größer als eins sein. Anstatt eines Parametersatzes M_i wird also jeder Mannschaft über die Funktion f nur ein einzelner Parameter m_i zugeordnet:

$$m_i = f(P_i) \tag{8}$$

Ein völlig äquivalenter Ansatz bestünde darin, m_i mit P_i zu identifizieren und die funktionale Abhängigkeit f zu einem Teil der Ergebnisfunktion zu machen.

3.2.1 Ergebnisfunktion

Für die Schätzfunktion (vgl. Abschnitt 3.1.2) wird ebenfalls eine sehr stark vereinfachende Annahme getätigt: Es wird davon ausgegangen, dass die Verteilungsfunktion für die Anzahl der geschossenen Tore einer Mannschaft ausschließlich vom Parameter *dieser* Mannschaft abhängt. Damit zerfällt Gleichung 5 in zwei unabhängige Teile:

$$E_A = S(M_A) \quad \text{und} \quad E_B = S(M_B) \quad \text{mit} \quad E = (E_A, E_B) \tag{9}$$

Nun wird als exemplarische Schätzfunktion (Verteilungsfunktion für die Toranzahl) die *Gleichverteilung* mit 0 als unterer und m_i als oberer Grenze verwendet.[2]

[2] Konkret wird eine reelle Zufallszahl zwischen 0 und m_i berechnet, die abgerundet die simulierte Toranzahl der Mannschaft i darstellt.

3.2.2 Berechnung der Weltrangliste

Nachdem die Parameter m_i der Mannschaften statisch sind, kann das Modell anhand einer fiktiven Neu-Berechnung der Weltrangliste verifiziert werden (s. Abschnitt 3.1.4).

Für dieses Vorgehen wird allerdings Kenntnis des offiziellen Berechnungsverfahren der Weltrangliste vorausgesetzt. Auf der Website der FIFA[3] findet sich eine lange Beschreibung, ohne dass das Verfahren jedoch in allen Details quantitativ offen gelegt wird. Allerdings lassen sich die angegebenen Beispiele verwenden, um eine gute Näherung für das verwendete Verfahren zu gewinnen.

Die Beschreibung der FIFA geht von einer bereits bestehenden Rangliste aus, die die Punktestände aller Mannschaften enthält. Ein Spiel zwischen zwei Mannschaften führt demnach zu einer Veränderung der bereits bestehenden Punktezahlen der beteiligten Mannschaften.

Die zu vergebenden Punkte einer Partie setzen sich dabei aus Punkten für den Spielausgang (Gewinn, Verlust oder Unentschieden) und aus Punkten für die einzelnen Tore zusammen. Der Stärkeunterschied der beiden Mannschaften wird mit Δ quantifiziert, welches dem Betrag der Differenz der Punktestände beider Mannschaften entspricht.

Dem Gewinner einer Partie werden für den erfolgreichen Ausgang des Spiels 20 Punkte gutgeschrieben, der Verlierer erhält dafür keine Punkte. Bei Unentschieden werden jeder Mannschaft 10 Punkte zugerechnet. Allerdings werden, um die unterschiedlichen Spielstärken der beiden Gegner zu berücksichtigen, zusätzlich dem schwächeren Partner pauschal $\Delta/50$ Punkte gutgeschrieben und dem stärkeren Partner ebensoviele Punkte abgezogen. (Dieser Teil der Berechnung stimmt zu 100% mit den Beispielen der FIFA-Website überein.)

Neben der Wertung für Sieg und Niederlage fließen auch Tore und Gegentore in die Berechnung ein. Für das erste Tor einer Mannschaft werden $2.5 \pm \Delta/750$ gutgeschrieben und für das erste erhaltene Gegentor werden $1.7 \pm \Delta/900$ abgezogen. Das Vorzeichen des $\pm$ ist dabei stets so zu wählen, dass die schwächere Mannschaft begünstigt wird. Um Toren und Gegentoren nicht zuviel Gewicht zu geben, beträgt die Punktzahl für jedes *folgende* Tor bzw. Gegentor einer Mannschaft jeweils nur $3/4$ der Punkte des jeweils vorangegangen Tors bzw. Gegentors. (Diese Überlegungen sind zwar durch die Beschreibung der FIFA-Website motiviert, haben jedoch eine etwas geringere Übereinstimmung mit den Beispielzahlen.)

Nun werden für jede Mannschaft die einzelnen Beiträge addiert und zur bisherigen Punktezahl der Rangliste hinzugefügt. Ergibt sich für eine Mannschaft aus der Verrechnung der Punkte für Tore und Spielausgang ein negatives Saldo, bleibt ihr Punktestand in der Rangliste unverändert.

Damit der Punktestand den *aktuellen* Leistungsstand der Mannschaften widerspiegelt, fallen Ergebnisse älterer Spiele mit der Zeit aus der Rangliste heraus. Die FIFA-Website führt dazu folgendes Schema auf: Spiele des aktuellen Jahres werden voll berücksichtigt, Spiele des vorangegangenen Jahres zu $7/8$, die des Jahres davor zu $6/8$, usw. bis hin zu einer Gewichtung

[3] http://www.fifa.com/de/mens/statistics/rank/procedures/0,2540,3,00.html

von $1/8$ für Spiele von vor 7 Jahren. Der aktuelle Punktestand eines Landes entspricht somit in etwa einem gewichteten gleitenden Durchschnitt der letzten 8 Jahre.

Gemeinsam mit der Angabe, dass jeweils pro Jahr nur maximal 7 Spiele gewertet werden, lässt sich daraus ein Normierungsfaktor ableiten:

$$N = 7 \cdot \left(\frac{8}{8} + \frac{7}{8} + \frac{6}{8} + \frac{5}{8} + \frac{4}{8} + \frac{3}{8} + \frac{2}{8} + \frac{1}{8} \right) = 31,5 \tag{10}$$

Das bedeutet, dass um die simulierte Rangliste mit der aktuellen Weltrangliste vergleichen zu können, die ermittelten Gesamtpunktzahlen der Mannschaften auf jeweils N Begegnungen normiert werden müssen.

Leider gibt es einige Details der Ranglistenberechnung, die für die Simulation nicht quantifiziert werden können: Die FIFA gewichtet die Spiele entsprechend ihrer Bedeutung[4] mit Multiplikatoren ungefähr zwischen 1 und 2 und es existiert ein nicht näher spezifizierter Auswärtsbonus.

Diese unbekannten Komponenten werden dadurch berücksichtigt, dass nach einem Simulationsdurchlauf die simulierte Rangliste auf die tatsächliche Weltrangliste normiert wird. D.h. sie wird linear skaliert, so dass die Summe der Punkte aller Teams aus der simulierten Rangliste mit der entsprechenden Summe aus der Weltrangliste übereinstimmt.[5]

3.2.3 Vergleich von Modell und Wirklichkeit

Wurde nun entsprechend der Beschreibung des vorangegangenen Abschnitts eine Weltrangliste simuliert, stellt sich die Frage der Quantifizierung des Unterschiedes zwischen Modell und Wirklichkeit. Hier bietet es sich an, als Qualitätsmerkmal D die Summe der quadratischen Differenzen zwischen den Punktwerten der Mannschaften in simulierter Rangliste X_i und in tatsächlicher Weltrangliste P_i heranzuziehen.

$$D = \sum_{i=1}^{n} (X_i - P_i)^2 \tag{11}$$

Ein Problem hierbei ist jedoch, dass die simulierte Weltrangliste von einem Lauf des Programms zum nächsten starken Schwankungen unterworfen ist. Schließlich gehen die berechneten Punktwerte der einzelnen Mannschaften auf Zufallsvariablen zurück. Um ausreichende

[4] und der regionalen Herkunft der Teams

[5] Sollte dabei ein Skalierungsfaktor kleiner als 1 oder deutlich größer als 2 zur Anwendung kommen, ist das ein starker Hinweis darauf, dass das bestehende Modell einen fundamentalen Fehler enthält: Es steht offensichtlich im Widerspruch zur Weltranglistenberechnung laut FIFA.

Genauigkeit zu erzielen, muss man den Vergleich von Modell und Wirklichkeit an den Ergebnissen vieler Simulationsläufe ausführen. Das bedeutet, dass die *durchschnittlichen* simulierten Punktwerte,

$$\langle X_i \rangle = \frac{1}{N} \sum_{j=1}^{N} X_i \tag{12}$$

die sich als Mittelwerte der Punktestände N einzelner Simulationen ergeben, mit den Punktewerten der tatsächlichen Weltrangliste verglichen werden:

$$D = \sum_{i=1}^{n} \left(\langle X_i \rangle - P_i \right)^2 \tag{13}$$

Im Zuge der Mittelwertbildung erhält man mit geringem zusätzlichen Aufwand gleichzeitig die Standardabweichung der Punktwerte der einzelnen Mannschaften σ_i:

$$\sigma_i = \frac{1}{\sqrt{N}} \cdot \sqrt{\langle X_i^2 \rangle - \langle X_i \rangle^2} \tag{14}$$

Damit lässt sich nun χ^2 berechnen, das ein besseres Maß für die Abweichung zwischen Simulation und Realität darstellt:

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{\langle X_i \rangle - P_i}{\sigma_i} \right)^2 \tag{15}$$

Im Vergleich zur Berechnung von D wird hier die Differenz $\langle X_i \rangle - P_i$ zusätzlich noch mit der Standardabweichung σ_i gegen-gewichtet, so dass die Abweichung der Simulation von der Realität für eine Mannschaft, deren simulierter Punktestand einer großen Streuung unterliegt, weniger stark ins Gewicht fällt, als für eine Mannschaft, deren simuliertes Punktekonto von Durchlauf zu Durchlauf nur geringen Abweichungen unterliegt.

3.2.4 Parametrisierung f

Zur Vollständigkeit des Modells fehlt jetzt nur noch eine Parametrisierung f, die die Parameter der Mannschaften m_i aus den Punktwerten P_i der Weltrangliste bestimmt.

Eine sehr einfache Parametrisierung von f mit nur einem einzigen Parameter besteht in einer Skalierung von P_i:

$$m_i = f(P_i) = \frac{P_i}{\alpha} \tag{16}$$

Zur Bestimmung des Parameters α wird χ^2 systematisch für viele unterschiedliche Werte für α ausgerechnet. Aus dieser Rechnung ergibt sich das niedrigste χ^2 für den Parameterwert $\alpha = 170$. Eine mit diesem Parameter simulierte Weltrangliste (Durschnitt aus 10000 Durchläufen) ist in Abbildung 10 gezeigt. An der horizontalen Achse sind die Mannschaften aufgetragen (in der Weltranglisten-Reihenfolge), während an der vertikalen Achse die jeweiligen Punktwerte abzulesen sind.

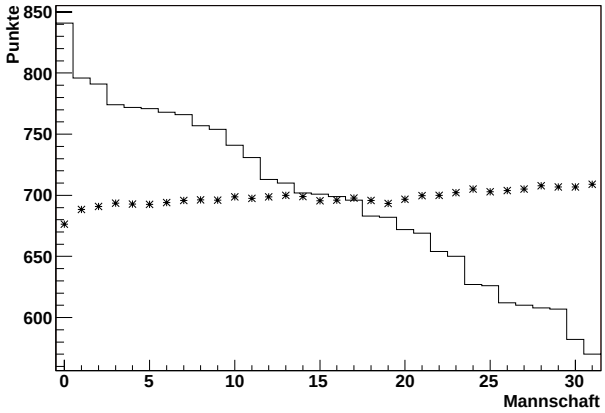


Abbildung 10: Simulierte Rangliste (Sterne) mit dem Parameter $\alpha = 170$ im Vergleich zur tatsächlichen Weltrangliste (durchgezogene Linie).

Leider fehlt hier jegliche Ähnlichkeit zwischen berechneter Rangliste (Sterne) und realer Weltrangliste (durchgezogene Linie). Nicht nur, dass die Punktwerte keine Übereinstimmung zeigen, es hat sich sogar die Hierarchie umgekehrt: Die beste Mannschaft erhält in der Simulation am wenigsten Punkte, wohingegen die schlechteste Mannschaft am besten abschneidet.

Dieses Resultat bedeutet, dass das Modell modifiziert werden muss. Der nächste Versuch erfolgt mit zwei Parametern in f:

$$m_i = f'(P_i) = \frac{P_i - \beta}{\alpha} \tag{17}$$

Die Minimierung von χ^2 führt nun auf die Parameterwerte $\alpha = 100$ und $\beta = 400$. Hier ergibt die Simulation eine Rangliste, die eine deutlich bessere Übereinstimmung mit der tatsächlichen Weltrangliste zeigt (s. Abbildung 11), wenn auch an den beiden Enden der Rangliste noch etwas größere Abweichungen auftreten.

Abbildung 12 zeigt zur Veranschaulichung des Minimierungsverfahrens das errechnete χ^2 für den betrachteten Ausschnitt des zwei-dimensionalen Parameterraums, der von α und β aufgespannt wird.

3.2.5 Ausbau des Modells

Zur Verbesserung des vorgestellten Modells bietet es sich an, zunächst die äußerst unrealistische Annahme zu korrigieren, dass die Anzahl der Tore einer Mannschaft unabhängig von der Stärke des Gegners ist. Realitätsnäher ist es sicherlich, das Ergebnis abhängig von der *Differenz* der Spielstärken zu machen.

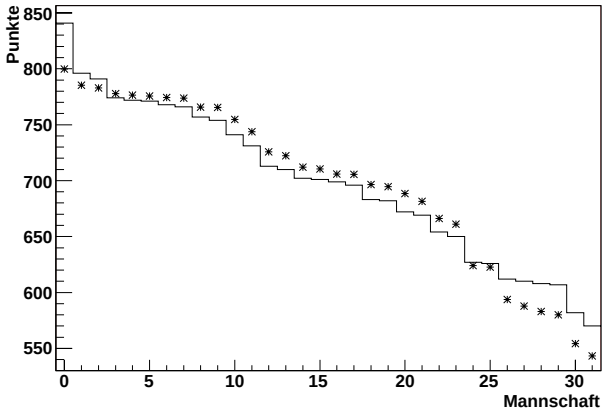


Abbildung 11: Simulierte Rangliste (Sterne) mit zwei Parametern $\alpha = 100$ und $\beta = 400$ im Vergleich zur tatsächlichen Weltrangliste (durchgezogene Linie).

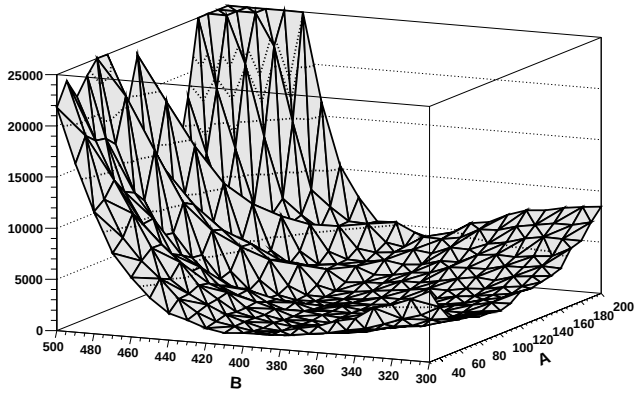


Abbildung 12: Berechnung von χ^2 für einen Ausschnitt des Parameterraums. Das Minimum von χ^2 liegt etwa bei $\alpha = 100$ und $\beta = 400$.

Diese Korrektur läßt sich durch eine Modifikation der Ergebnisfunktion S aus Gleichung 9 erreichen, deren Abhängigkeit vom Parametersatz *einer* Mannschaft durch eine Abhängigkeit von der Differenz der Parametersätze *beider* Mannschaften ersetzt wird:

$$E_A = S(M_A - M_B) \quad \text{und} \quad E_B = S(M_B - M_A) \quad \text{mit} \quad E = (E_A, E_B) \tag{18}$$

Setzt man nun die Parametrisierung aus Gleichung 17 ein, fällt der Parameter α aus der Differenz heraus:

$$E_A = S\left(\frac{P_A - P_B}{\beta}\right) \quad \text{und} \quad E_B = S\left(\frac{P_B - P_A}{\beta}\right) \tag{19}$$

Für eine ausreichende Flexibilität des Modells sollten zumindest zwei Parameter zur Verfügung stehen, deshalb wird ein zusätzlicher Parameter γ eingeführt. Für S wird wieder wie in Ab-

schnitt 3.1.2 beschrieben die Gleichverteilung verwendet.

$$E_A = S\left(\frac{P_A - P_B}{\beta} + \gamma\right) \quad \text{und} \quad E_B = S\left(\frac{P_B - P_A}{\beta} + \gamma\right) \tag{20}$$

Das Ergebnis der Verbesserung des Modells ist in Abbildung 13 zu sehen. Ohne die Anzahl der freien Parameter zu erhöhen wird nur durch realistischere Gestaltung des Modells eine deutlich bessere Übereinstimmung zwischen tatsächlicher Weltrangliste und anhand des Modells berechneter Rangliste erreicht.

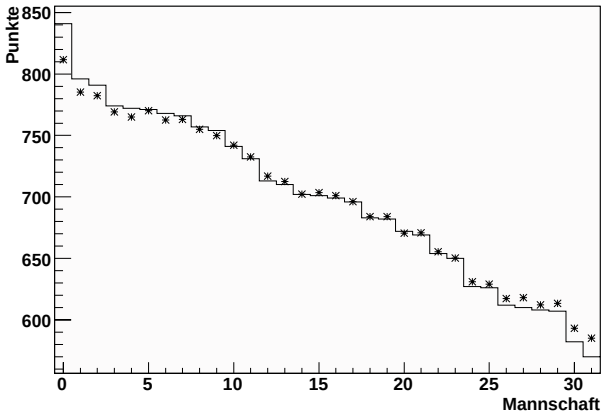


Abbildung 13: Simulierte Rangliste (Sterne) mit zwei Parametern $\beta = 600$ und $\gamma = 135$ im Vergleich zur tatsächlichen Weltrangliste (durchgezogene Linie).

3.2.6 Weitere Ideen zum Ausbau des Modells

Eine verbleibende Unzulänglichkeit des beschriebenen Modells liegt in der Wahrscheinlichkeitsverteilung der Tore. In der Realität ist nicht davon auszugehen, dass die Toranzahl einer Mannschaft einer Gleichverteilung zwischen Null und einer oberen Grenze entspricht. Eher würde ein Ansatz die Wirklichkeit beschreiben, der berücksichtigt, dass die Wahrscheinlichkeit für ein hohes Ergebnis mit der Anzahl der Tore immer mehr abnimmt (ohne jedoch allzu bald die Null zu erreichen).[6]

Natürlich könnte man auch noch die Parametrisierung f – z.B. durch Einfügen quadratischer und höherer Terme in $(P_A - P_B)$ – verfeinern, allerdings wächst dadurch die Anzahl der freien Parameter. Eleganter und vielversprechender wäre es wohl, zunächst weitere Fehler bzw. Ungenauigkeiten der Modellierung zu beheben.

[6] Z.B. wäre hier eine Poisson-Verteilung vorstellbar, die sich auch dahingehend interpretieren ließe, dass eine Partie aus einer gewissen Anzahl von Spielzügen (Ballbesitz einer der Mannschaften) besteht, die nacheinander durchgeführt werden und für die jeweils eine gewisse (relativ geringe) Torwahrscheinlichkeit besteht.

3.2.7 Berücksichtigung externer Faktoren

Teil 3 der Aufgabenstellung fordert die Berücksichtigung externer Faktoren, nach denen die tatsächliche Spielstärke einer Mannschaft von der durch die Platzierung in der Weltrangliste repräsentierten abweichen kann. Es bietet sich an, solche Faktoren als äquivalente Bonus- oder Maluspunkte $\pm x_i$ zu berechnen, die dann bei der Bestimmung der Mannschaftsparameter m_i zu den zugrundeliegenden Weltranglistenpunkten addiert werden:

$$m_i = f(P_i + x_i) \tag{21}$$

Nachteil des Gastgebers bei der Weltranglistenberechnung Der Gastgeber der WM ist automatisch qualifiziert und führt deshalb in der Vorbereitungsphase der WM keine Qualifikationsspiele sondern nur Freundschaftsspiele durch. Nachdem jedoch Qualifikationsspiele durch die FIFA mit dem Faktor 1,5 gegenüber Freundschaftsspielen übergewichtet sind, hat der Gastgeber einen Nachteil bei der Berechnung der Weltrangliste, er wird also gegebenenfalls schlechter dargestellt als er tatsächlich spielt.

Dieser Effekt lässt sich grob abschätzen. Unter der Annahme, dass für nicht automatisch qualifizierte Mannschaften etwa drei Viertel der Begegnungen in den zwei Jahren vor der WM Qualifikationsspiele sind, ergibt sich ein Anteil r der für die Weltrangliste berücksichtigten Partien, der von Qualifikationsspielen herrührt:

$$r = \frac{3/4 \cdot 7 \cdot \left(\frac{7+8}{8}\right)}{N} \approx 0,31 \tag{22}$$

Dieser Anteil der Partien geht für eine Mannschaft, die sich qualifizieren muss, mit Faktor 1,5 in den Punktestand ein. Für den Gastgeber wird dieser Anteil der Partien jedoch nicht gewichtet (Faktor 1,0), da es sich ja um Freundschaftsspiele handelt.

Der durchschnittliche *ungewichtete* Punktesaldo p (nach Addition von Beiträgen für Spielausgang und Torzahlen aber vor der Multiplikation der Gewichtungsfaktoren), den der Gastgeber pro Begegnung (gemittelt über die Gesamtheit aller Spiele) erzielen konnte, ergibt sich somit zu

$$p = \frac{P}{N \cdot (r + (1-r) \cdot 1,5)} \approx \frac{P}{42}, \tag{23}$$

wenn man davon ausgeht, dass die Spiele, die nicht im Zusammenhang mit der aktuellen WM stehen, im Mittel mit Faktor 1,5 gewichtet sind.

Die korrigierte Punktzahl des Gastgebers P' erhält man nun, indem der Anteil von P, der auf Qualifikationsspiele zurückgeht, mit Faktor 1,5 „nachgewichtet“ wird:

$$P' = P - N \cdot r \cdot p + 1,5 \cdot N \cdot r \cdot p \tag{24}$$

Für den Gastgeber Deutschland mit $P = 710$ Weltranglistenpunkten würde sich durch diese Korrektur der Punktwert um etwa 80 Punkte auf 790 Punkte erhöhen, womit der Anschluss an

die Spitzengruppe des Turniers (zumindest mathematisch) hergestellt wäre. Allerdings bleibt zu berücksichtigen, dass sich die Gegner in den Freundschaftsspielen – hätte es sich tatsächlich um Qualifikationsspiele gehandelt – wohl deutlich mehr angestrengt hätten. Somit wird sich die tatsächliche Korrektur zwischen 0 und 80 Punkten bewegen.

Heimvorteil des Gastgebers Leider gibt es keinen mathematischen Ansatz, den Heimvorteil des Gastgebers herzuleiten. Was bleibt ist eine Abschätzung aus dem Bauch heraus: Es ist wohl nicht davon auszugehen, dass eine Mannschaft wie die deutsche durch das Spiel auf der heimatlichen Scholle derart beflügelt wird, dass sie sich der brasilianischen als ebenbürtig erweisen würde. Entsprechend ist der Heimvorteil wohl bei deutlich weniger als der Punktdifferenz der beiden Länder von 130 Punkten anzusiedeln.

3.3 Bewertung

3.3.1 Teilaufgabe 1

Dies ist sicherlich der aufwändigste Aufgabenteil. Für volle Punktzahl sollte ein vollständiges Modell entwickelt, diskutiert und verifiziert werden.

Die Mindestanforderung an das Modell ist eine gewisse Realitätsnähe. Im Mittel muss eine bessere Mannschaft gegen eine schlechtere öfter gewinnen als verlieren. Eine bessere Mannschaft muss gegen eine etwas schlechtere Mannschaft weniger oft gewinnen als gegen eine deutlich schlechtere Mannschaft. Die Erwartungswerte für Toranzahlen sollten im realistischen Rahmen liegen: nicht deutlich kleiner als 1 und nicht deutlich größer als 5.

Zur Diskussion des Modells gehören plausible Angaben über durchschnittliche Toranzahlen und Statistiken der Ergebnisse, jeweils für die Begegnung zweier in etwa gleich starker und zweier unterschiedlich starker Mannschaften.

Die Überprüfung des Modells anhand der Weltrangliste ist erforderlich. Wenn das Modell nicht-statische, also sich im Verlauf des Turniers verändernde Parameter enthält, sollte zumindest der statische Teil anhand der Weltrangliste verifiziert und dann die Übertragung der Resultate auf den nicht-statischen Teil diskutiert werden.

Es sollte eine persönliche Einschätzung darüber gegeben werden, wie gut das entwickelte Modell die Wirklichkeit beschreibt. Ein Vergleich verschiedener Modelle ist interessant, geht aber über das erwartete Maßhinaus.

3.3.2 Teilaufgabe 2

Hier gibt es keine besonderen Schwierigkeiten. Ein System, das ein Turnier gemäß der Eingaben korrekt und vollständig simuliert, ist Voraussetzung für die Lösung der Teilaufgabe 4. Es muss u.a. den Turnierplan und die Gruppenregeln richtig umsetzen. Es wird erwartet, dass die

Flexibilität des Formats zur Beschreibung des Turnierplans erkannt wird und das Simulationssystem auch Turniere durchspielen kann, deren Struktur vom gegebenen Beispiel abweicht.

3.3.3 Teilaufgabe 3

Bei dieser Teilaufgabe kann der Phantasie freier Lauf gelassen werden. Plausibilitätsüberlegungen für zusätzlich eingeführte Parameter dürfen aber nicht fehlen. Mindestens sollten die von der Aufgabenstellung vorgegebenen Faktoren „Heimvorteil“ und „Nachteil bei der Weltranglistenberechnung“ (oder zwei andere von ähnlicher Komplexität) besprochen werden.

3.3.4 Teilaufgabe 4

Aus der mehrfachen Simulation des Turniers ergeben sich einige Statistiken; sinnvoll sind so oder ähnlich:

- Wahrscheinlichkeiten der einzelnen Mannschaften den Turniersieg zu erreichen
- Wahrscheinlichkeiten der einzelnen Mannschaften eine bestimmte Runde zu erreichen
- Wahrscheinlichkeiten bestimmter Paarungen (einer bestimmten Runde und/oder zu einem bestimmten Spieltermin und/oder an einem bestimmten Austragungsort)

Beispiele sollten auf alle berechneten Statistiken zurückgreifen und auch auf verschiedenen Simulationsläufen beruhen. Besondere Mühe kann hier auch besonders belohnt werden.

3.3.5 Zusatzpunkte/Erweiterungen

Zusatzpunkte gibt es, wenn eine Teilaufgabe besonders gut gelöst oder ein Teilaspekt besonders gut herausgearbeitet wurde oder wenn über die Aufgabenstellung hinausgehende Themen ausführlich betrachtet wurden. Einige Beispiele:

- Berücksichtigung der Eigenheiten verschiedener Sportarten;
- besonders aufwändige Diskussion der Berechnung der FIFA-Weltrangliste;
- ausführliche Betrachtung des systematischen (nicht-statistischen) Fehlers des entwickelten Modells, also seiner inhärenten (Un-)Genauigkeit;
- umfangreiche Suche im Parameterraum nach der besten Parameterkombination;
- sinnvoll durchgeführter quantitativer Vergleich mehrerer unterschiedlicher Modelle;
- aufwändige nicht-statische Modelle (Berücksichtigung von Kondition, Motivation, Wind und Wetter, etc.).

Perlen der Informatik – aus den Einsendungen

Allgemeines

Dies kleine Gedicht, so einfach und schlicht
ist der Anfang der folgenden Seiten
und wünscht allen Gescheiten
viel Spaß beim Durcharbeiten.

Für den interessierten Leser werden folgende Werke als weiterführende / ergänzende Literatur empfohlen.

Bis ich zu dieser Erkenntnis gelangte, hatte ich unzählige Fehlschläge. Die schönsten möchte ich im Folgenden zur allgemeinen Erheiterung auflisten.

Dadurch wird die Anzahl der unterstützten Betriebssysteme leider erheblich eingeschränkt; es stehen nur noch UNIX / Linux, Microsoft Windows und Mac OS X zur Verfügung.

Aufgabe 1

Phytagoras

Trifft der Strahl eine Ecke, kriegt die Funktion Panik und erklärt den Ausnahmezustand.

Aus Zeitgründen konnte keine weitere Klasse ’Ray’ implementiert werden. Dies sei dem Leser als Übung auf Grundlage der vorhandenen Klassen überlassen.

Dies ist eindeutig ein Besucher: er geht desinteressiert am Rafael Roth Learning Center vorbei, auch den Holocaust-Turm guckt er sich nicht an, im Garten des Exils wird ihm dann etwas schwindelig (vielleicht auch vom ständigen 360-Grad-Drehen) und er taumelt zurück zum Ausgang.

Nun wandert der Nachtwächter aber über unendlich viele Punkte, und wir haben nicht genug Zeit, den Algorithmus mit allen auszuführen.

Aufgabe 2

Alternativer Aufgabentitel: Die sportliche Kafkalake

Noch nie habe ich über ein so kurzes Programm so lange nachgedacht.

Algorithmen mit quadratischem Speicherverbrauch oder ähnlichen Späßen kann man sich also schon einmal abschminken.

Auf die mathematische Konstruktion möchte ich aus informatischen Gründen nicht weiter eingehen.

Aufgabe 3

```
Wer wird Weltmeister:
Bayern München    39,5%
```

Mit all diesem kann der Verlauf und das Ergebnis beliebig angepasst, ja beeinflusst werden; und ganz ohne die kroatische Wettmafia.

Wunder-von-Bern-Effekt

```
if (wunder) { // Ein Wunder kehrt das Spielergebnis um
```

Nachdem mein Programm die WM mit diesen Werten simuliert hat, ist mir aufgefallen, dass die Ergebnisse zum Teil recht hoch ausgefallen sind. Ich habe deshalb die Anzahl der (virtuellen) Spielminuten auf 60 gesenkt.

Die Chance nahe beieinander liegender Mannschaften kommen jedoch oft auf annähernd 50:50, so z.B. bei Deutschland und Costa Rica, obwohl Deutschland eigentlich besser abschneiden sollte. *Wie wahr, wie wahr . . .*