Start Competitive Programming!: Ace the USACO Bronze Competition

Updated and revised

Includes 2023-2024 problems

Zachi Baharav and **Daniel Zingaro**

Deutsche Ausgabe

Start Wettbewerbsprogrammierung!: Meistern Sie den USACO Bronze-Wettbewerb (Deutsche Ausgabe)

Zachi Baharav, Daniel Zingaro, und TranslateAI

Dieses Buch wird verkauft unter http://leanpub.com/start_competitive_programming-de

Diese Version wurde veröffentlicht am 2024-06-09

Inhaltsverzeichnis

Brief an den Schüler

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Brief an die Eltern

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Brief an den ausgebildeten Fachmann

Danksagungen

Teil I. Vorbemerkungen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 1. USACO Bronze

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

1.1. USACO Bronze FAQ

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

1.2. Ein USACO-Problem lösen und einreichen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

1.3. Wie man mit diesem Buch arbeitet

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

1.4. Zusammenfassung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 2. Lösen und Codieren: Wettbewerbsspezifika

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.1. Lesen und Analysieren eines USACO-Problems

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.1.1. Lesen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.1.2. Visualisieren

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.1.3. Algorithmus

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.2. Deinen Algorithmus programmieren

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.2.1. Form und Stil

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.2.2. Coding Patterns

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.3. Debugging

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.3.1. Debugging im Training (wenn Sie die erwartete Lösung haben)

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.3.2. Debugging im Wettbewerb

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.4. Verwendung einer Lösung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

2.5. Zusammenfassung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 3. Komplexitätsanalyse

3.1. Große O Notation

3.2. Zeitkomplexität

Problem 3.1: Exakte Gruppengröße

3.3. Speicherkomplexität

Problem 3.2: Fehlende Zahl

3.4. Zusammenfassung

Teil II. Kerntechniken

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 4. Modellierung und Simulation

4.1. Modellierung eines dynamischen Prozesses

4.1.1. Modellierung von Zeitschritten

Problem 4.1: Rund um den See spazieren

4.1.2. Modellierungsprozess-Schritte

Problem 4.2: Wo ist der König?

4.2. Modellierung eines statischen Prozesses

Problem 4.3: Ein Besuch im Mooseum

4.3. Modellierung eines periodischen Prozesses

Problem 4.4: Das Riesenrad

4.4. Beschleunigung der Simulation

Problem 4.5: Spaziergang zum Opernhaus

4.5. Zusammenfassung

Kapitel 5. Suche und Optimierung

Dieses Kapitel behandelt

- Das Erkennen von Suchproblemen im Kontext von USACO.
- Das Lösen von Suchproblemen mithilfe eines vollständigen Suchalgorithmus.
- Die Auswahl eines Bereichs für die Durchführung der Suche.
- Die Aufzählung des gewählten Bereichs.
- Die Beschleunigung eines vollständigen Suchalgorithmus.
- Das Lösen von Suchproblemen mithilfe eines gierigen Algorithmus.

Bei Suchproblemen, wie der Name schon sagt, suchen wir nach etwas. Suchprobleme sind ein weites und intensives Forschungsgebiet und die Entwicklung von Algorithmen in der Informatik. Sie sind wahrscheinlich mit vielen Anwendungen von Suchalgorithmen vertraut: das Suchen nach einem Wort in einem Dokument, das Sie gerade schreiben; das Suchen nach Phrasen im Web; das Suchen nach dem kürzesten Weg von Punkt A nach Punkt B. Aber Suchprobleme haben noch breitere Anwendungen, von denen viele versteckte Suchen beinhalten. Zum Beispiel identifiziert Ihre Autokorrektur das Wort, das demjenigen, das Sie geschrieben haben, am nächsten kommt. Im Hintergrund durchsucht sie alle möglichen Wörter in ihrem Wörterbuch, bezieht sich auf ihr Wissen darüber, welche Wörter häufiger verwendet werden, und schlägt ein neues Wort vor.

Oft werden Suchprobleme auch als Optimierungsprobleme bezeichnet. Optimierungsprobleme streben danach, das bestmögliche Ergebnis für eine bestimmte Bedingung zu erzielen. Betrachten Sie zum Beispiel das Problem, eine Verkehrsampel zu entwerfen, die den maximalen Verkehrsfluss ermöglicht. Wir können bestimmte Grünphasen für die verschiedenen Richtungen festlegen (nicht gleichzeitig!), dann modellieren, wie viele Autos in jede Richtung fließen. Dann können wir diese zugewiesenen Grünphasen ändern und den neu resultierenden Verkehrsfluss modellieren. In diesem Problem suchen wir: Wir versuchen, die beste Zuweisung der Grünphasen zu finden, die den

maximalen Verkehrsfluss ermöglicht. In solchen Optimierungsproblemen finden wir die Lösung oft mithilfe von Suchalgorithmen.

Suchprobleme sind auf allen Ebenen von USACO sehr häufig. Keine Sorge: Dieses Kapitel behandelt nur die Suchalgorithmen, die auf dem Bronze-Niveau benötigt werden. Sie werden mehr lernen, wenn Sie durch die USACO-Level aufsteigen.

Die Kapitelkarte ist in Abbildung 5.1 beschrieben. Der häufigste Suchalgorithmus auf dem Bronze-Niveau ist die vollständige Suche, auch bekannt als komplette Suche oder brute-force Suche, die in Abschnitt 5.1 beschrieben wird. Dieser Algorithmustyp beinhaltet das Durchsuchen aller möglichen Optionen. Zum Beispiel könnte die Rechtschreibprüfung alle möglichen Wörter im Wörterbuch durchsuchen und entscheiden, welches dem falsch geschriebenen Wort am nächsten kommt. Eine vollständige Suche erfordert zwei Hauptentscheidungen. Erstens, welchen "Raum" durchsuchen wir? Zum Beispiel, durchsuchen wir alle Wörter in einem bestimmten britischen oder amerikanischen Wörterbuch? Dieser zu durchsuchende "Raum" wird als Bereich der Suche bezeichnet und in Abschnitt 5.2 behandelt. Zweitens, wie wissen wir, dass wir alle Optionen durchsucht haben? Oder mit anderen Worten, wie ordnen wir die Elemente im Bereich? Im Fall der Autokorrekturfunktion können wir alle Wörter in alphabetischer Reihenfolge durchgehen. Im Fall des kürzesten Weges zwischen zwei Punkten auf der Karte, wo wir viele Straßen berücksichtigen müssen, ist die Antwort nicht so klar. Wir brauchen eine Art Prozess, um eine Reihenfolge für die Durchsuchung aller Elemente festzulegen. Dieser Prozess wird Aufzählung genannt und in Abschnitt 5.3 behandelt.

Abschnitt 5.4 beschreibt Möglichkeiten zur Beschleunigung des Suchalgorithmus. Diese Überlegung ist es wert, auf dem Bronze-Niveau untersucht zu werden, obwohl sie auf den fortgeschrittenen Ebenen von USACO eine zentralere Rolle spielt. Wir schließen in Abschnitt 5.5 mit einer Diskussion eines anderen Suchalgorithmus, des gierigen Algorithmus. Im Gegensatz zu einer vollständigen Suche kann ein gieriger Algorithmus eine Lösung finden, ohne alle Optionen zu überprüfen. Dies kann zu einer erheblichen Reduzierung der Ausführungszeit des Algorithmus führen - könnte jedoch die beste Lösung nicht finden. Wir werden Fälle untersuchen, in denen ein gieriger Algorithmus funktioniert, sowie Fälle beschreiben, in denen er versagt.

Im gesamten Kapitel werden wir auf viele Such- und Optimierungsprobleme stoßen. Eines der Hauptziele dieses Kapitels ist es, Ihnen beizubringen, ein Problem als Suchproblem zu erkennen, eine Fähigkeit, die es viel einfacher macht, einen Algorithmus für eine Lösung zu entwickeln. Achten Sie besonders darauf, wenn wir die Schlüsselbegriffe und Konzepte hervorheben, die darauf hinweisen, dass wir es mit einem Suchproblem zu tun haben.

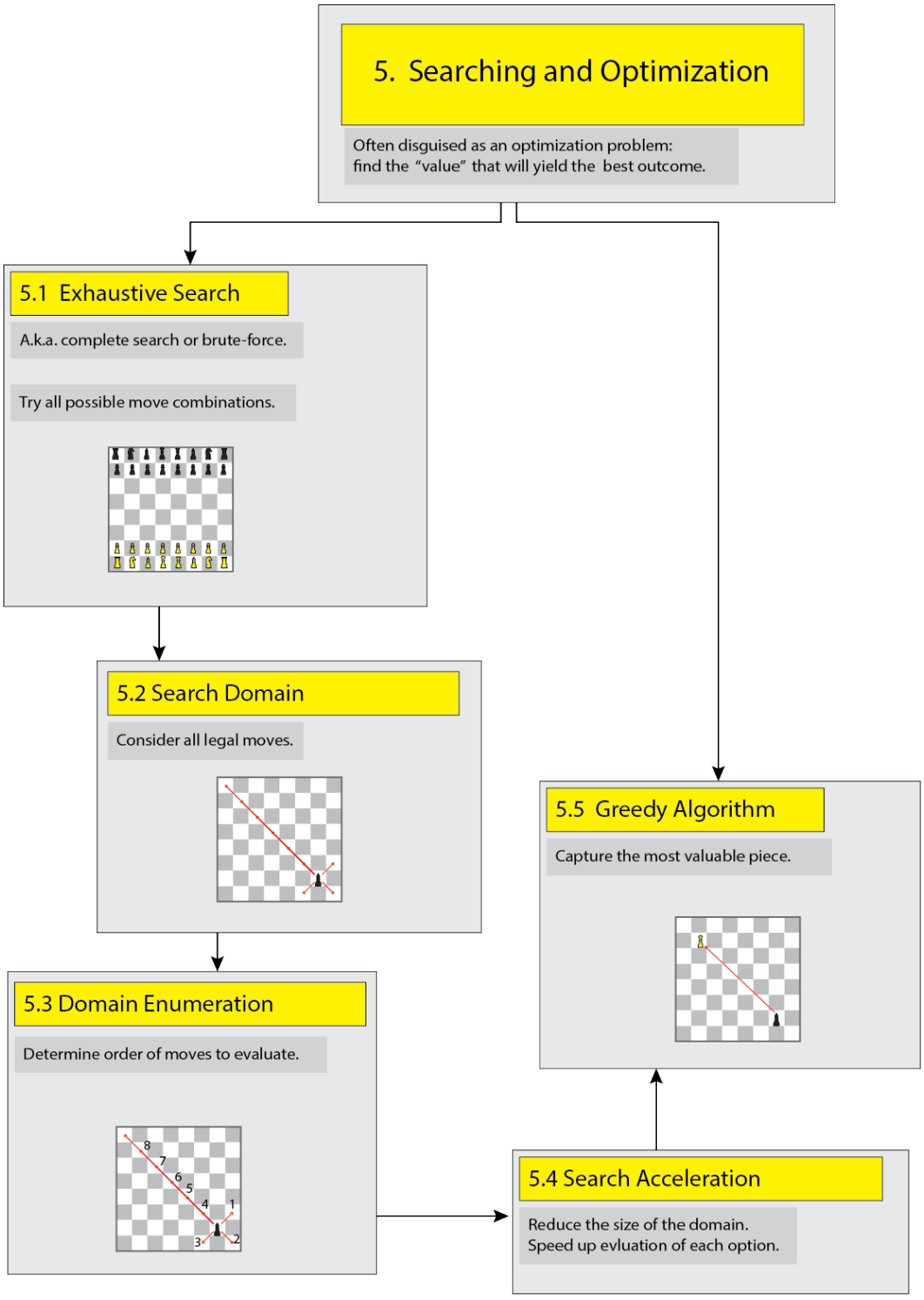

Abbildung 5.1 Kapitelkarte Suche und Optimierung. Wir behandeln zwei Arten von Suchalgorithmen: vollständige Suche und gierige Algorithmen.

5.1. Vollständige Suche

Coach B: Einen schönen Dienstag, alle zusammen. Heute lernen wir über erschöpfende Suchalgorithmen. "Erschöpfende Suche" ist ein sehr treffender Name für diese Methode: Es bedeutet, dass wir alle möglichen Optionen durchsuchen; es deutet auch darauf hin, dass wir, oder zumindest der Computer, nach dieser Suche erschöpft sind. Dies liegt daran, dass er über viele, viele Optionen suchen muss. Unser erstes Problem findet Bessie und ihre Freunde auf Hawaii! Lesen Sie das Problem durch, und wir werden es besprechen.

Problem 5.1: Tiki-Fackeln

Bessie liebt den Waikiki-Strand bei Nacht, wenn Tiki-Fackeln den goldenen Sand erleuchten. Aber es ist teuer, diese Fackeln am Brennen zu halten, und die Leute vom Amt für Naturschutz haben Bessie um Hilfe gebeten. Ihre Aufgabe ist es, eine Fackel zu finden, die entfernt werden kann und die zu minimalen Störungen führt. Diese Fackel darf nicht die erste oder die letzte in der Reihe sein, da diese wichtig sind, um die Gäste zu orientieren.

Bessie hat in ihrem Notizbuch notiert, dass es N Tiki-Fackeln gibt, $2 < N < 10^5$, die entlang des Strandes in einer geraden Linie platziert sind. Ein Standort einer Tiki-Fackel wird durch eine einzelne Zahl angegeben, x_i.

Bestimmen Sie, welche Fackel entfernt werden kann, sodass die maximale Entfernung zwischen zwei benachbarten verbleibenden Fackeln minimal ist.

Eingabeformat

Zwei Zeilen.

Die erste Zeile enthält eine einzelne Zahl, N.

Die zweite Zeile enthält N Ganzzahlen, die die Standorte der Fackeln angeben, $x_1, x_2, \ldots, x_N$.

Es ist gegeben, dass $x_1 < x_2 < x_3 < \ldots < x_N$.

Ausgabeformat

Eine Zahl, den Standort der zu entfernenden Tiki-Fackel. Wenn es mehrere Standorte gibt, die das gleiche Ergebnis liefern, geben Sie einen dieser Standorte aus (irgendeiner reicht aus).

Beispiel-Eingabe

```
6

1 8 10 16 20 23
```

Beispiel-Ausgabe

```
20
```

Wenn wir die Fackel an Standort 20 entfernen, ist die maximale Entfernung zwischen zwei benachbarten Fackeln 7, was der kleinstmögliche Wert ist.

DISKUSSION

Das Team liest das Problem und schaut sich dann verwirrt um.

Coach B: Ich sehe ein paar verwirrte Gesichter. Fangen wir also ganz von vorne an. Das Problem fragt uns, welche Tiki-Fackel wir entfernen sollen, richtig? Und es gibt nur so viele Tiki-Fackeln. Das sagt uns, dass dies ein Suchproblem sein könnte: Wir müssen unter allen Tiki-Fackeln die beste zum Entfernen finden.

Ryan: Danke, Coach B. Ich habe diesen Teil verstanden, aber ich bin immer noch verwirrt, was sie eigentlich wollen. Sie fragen nach einer maximalen Entfernung, aber dann wollen sie sie minimal haben. Lese ich das falsch?

Coach B: Du liest es richtig, Ryan. Dies ist eine sehr häufige Formulierung in Optimierungsproblemen. Bei Optimierungsproblemen suchen wir nach der besten Konfiguration. In unserem Fall suchen wir die beste Tiki-Fackel zum Entfernen. Also versuchen wir, das Problem zu skizzieren. Da das Problem für uns nicht ganz Sinn ergibt, beginnen wir mit den Teilen, die Sinn ergeben. Ich weiß, es ist schwierig, aber versuchen wir, uns mit dem Unkomfortablen anzufreunden! Ryan oder jemand anderes, könnt ihr bitte die Beispiel-Eingabe für uns zeichnen? Das bringt uns weiter.

TIPP: Bleiben Sie nicht bei den Teilen des Problems stecken, die Sie nicht verstehen. Beginnen Sie mit den Dingen, die Sie verstehen, und sehen Sie, ob Sie den Rest herausfinden können.

Visualisieren: Ryan geht zur Tafel, während sich der Rest des Teams anschließt. Während Ryan die Standorte zeichnet, fügt Annie die Tiki-Fackeln hinzu, wie in Abbildung 5.2.

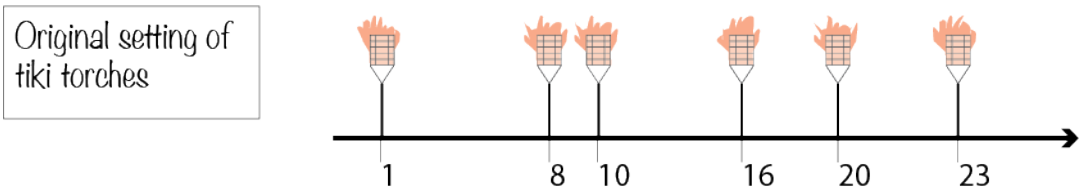

Abbildung 5.2 Die anfängliche Platzierung der Tiki-Fackeln.

Coach B: Großartig. Ich liebe die Tiki-Fackeln. Nun spricht das Problem davon, eine Fackel zu entfernen. Lassen Sie uns eine auswählen, entfernen und sehen, wie es aussieht.

Rachid: Wir können nicht die erste oder die letzte entfernen, also lassen Sie uns die an Standort 8 entfernen.

Rachid zeichnet das Setting neu, wie in Abbildung 5.3, ohne die Fackel an Position 8.

TIPP: Wenn möglich, versuchen Sie nicht zu radieren oder über vorherige Zeichnungen zu schreiben. Dies ermöglicht Ihnen, den Fortschritt Ihrer Arbeit zu sehen und wie sich die Dinge entwickeln. Natürlich sind einige Probleme zu kompliziert, um sie jedes Mal neu zu zeichnen. Finden Sie den richtigen Weg für sich, aber bedenken Sie, dass klare Zeichnungen beim klaren Programmieren helfen.

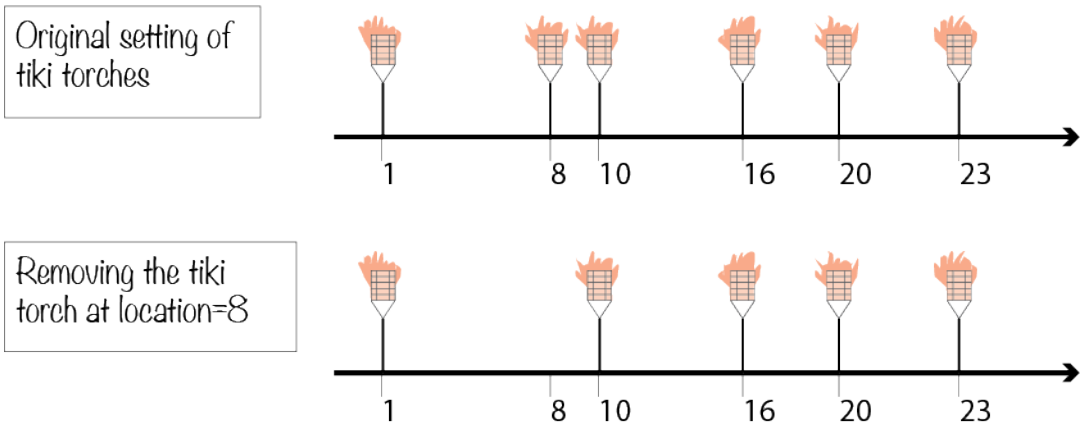

Abbildung 5.3 Entfernen der Tiki-Fackel an Position 8.

Coach B: Sieht perfekt aus. Wir machen Fortschritte. Was ist nun die maximale Entfernung zwischen zwei benachbarten Tiki-Fackeln?

Rachid schreibt die Entfernung zwischen allen benachbarten Fackeln auf, wie in Abbildung 5.4.

Rachid: Die maximale Entfernung beträgt 9, zwischen den Fackeln an den Positionen 1 und 10. Ich betrachte 1 und 10 zusammen, weil wir eine Fackel entfernt haben, die ursprünglich zwischen ihnen war, nämlich die an Position 8.

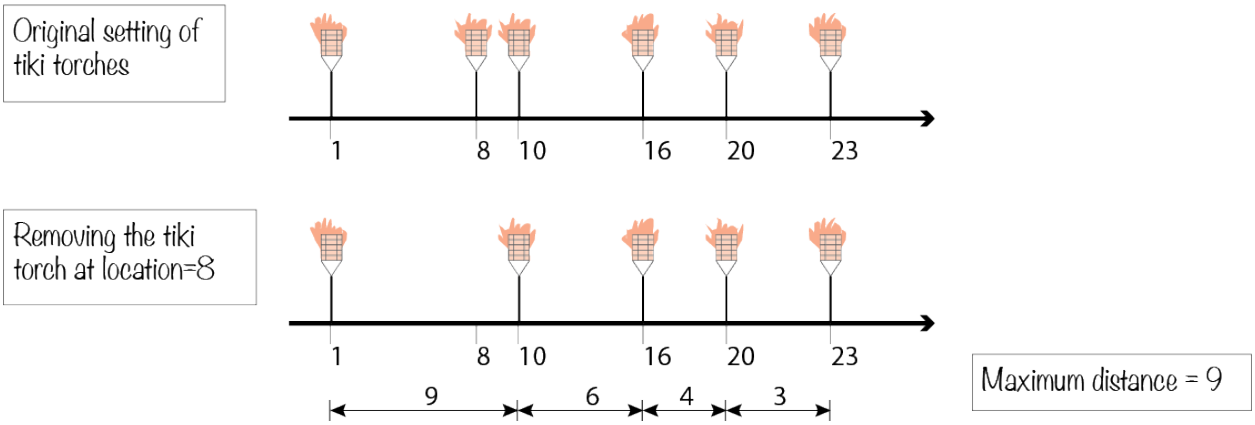

Abbildung 5.4 Nach dem Entfernen einer Tiki-Fackel untersuchen wir die Skizze, um die größte Entfernung zwischen zwei Fackeln in diesem Setting zu finden.

Annie: Oh, ich glaube, ich verstehe es. Jetzt müssen wir versuchen, andere Fackeln zu entfernen und sehen, was die maximale Entfernung dann ist. Am Ende nehmen wir das Minimum davon. Ist das richtig?

Coach B: Klingt richtig für mich! Mach weiter, die Tafel gehört dir.

Annie und das Team beginnen, die verschiedenen Fälle zu zeichnen, wie in Abbildung 5.5.

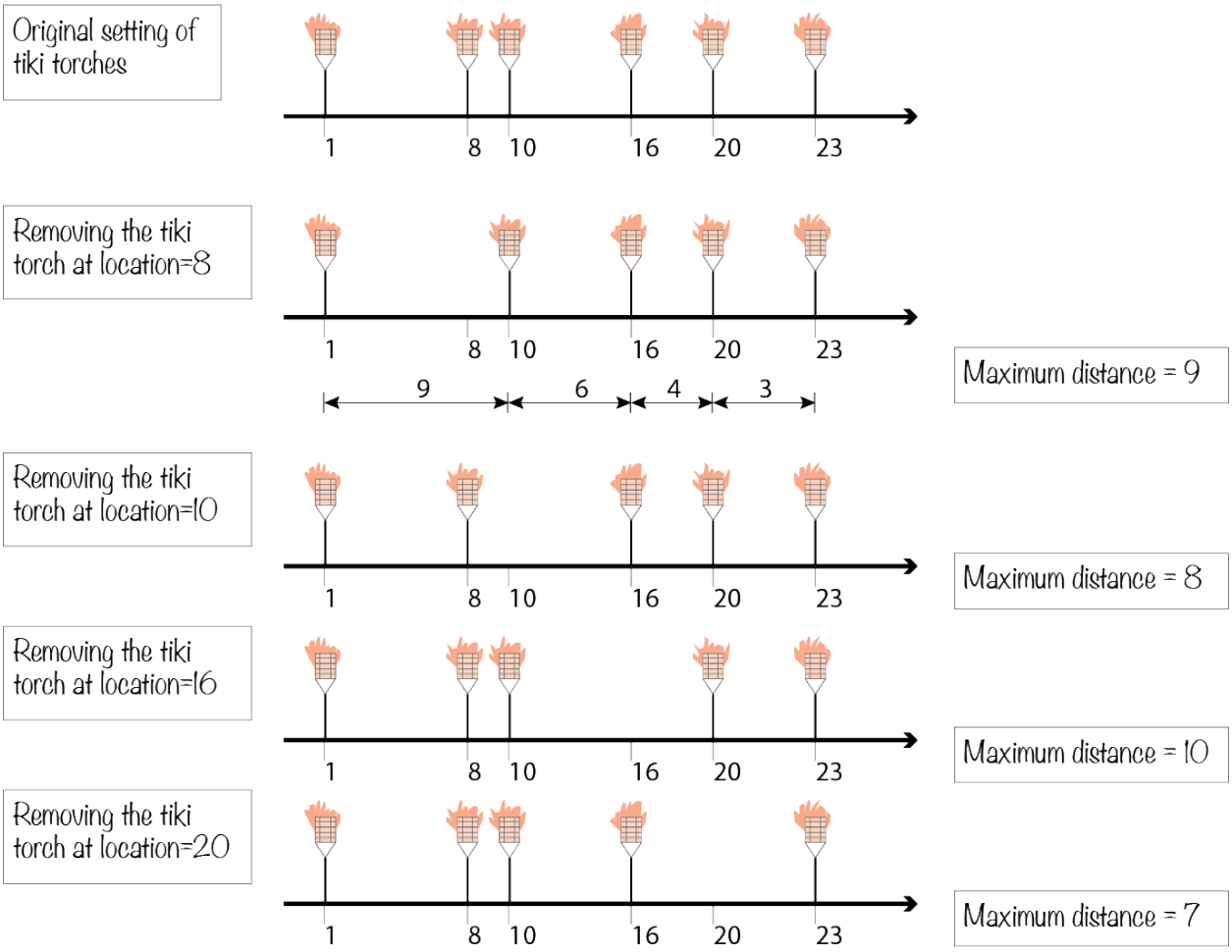

Abbildung 5.5 Untersuchen aller möglichen Tiki-Fackeln zum Entfernen und für jeden Fall die resultierende maximale Entfernung zwischen benachbarten Fackeln angeben.

Mei: Wenn wir das Minimum aus diesen nehmen wollen, beträgt es 7. Das war, als wir die Fackel an Position 20 entfernt haben.

Coach B: Und siehe da, das ist die Antwort, die sie im Problem für das Beispiel-Input haben. Gut gemacht! Ryan, ergibt das jetzt Sinn? Kannst du es in deinen eigenen Worten formulieren?

Ryan: Ich kann es versuchen... Also, hier ist, wie ich es formulieren würde: "Bessie möchte dem Team helfen und eine Fackel entfernen. Das Problem ist, dass jede Fackel, die sie entfernt, einen Abschnitt des Strandes etwas weniger beleuchtet macht. Ihre Aufgabe ist es, eine Fackel so zu entfernen, dass die resultierende Länge des Strandes ohne eine Tiki-Fackel am kürzesten ist. Hilf Bessie zu bestimmen, welche Tiki-Fackel sie entfernen soll."

Mei: Wow, können wir Ryan nominieren, um USACO-Fragen zu schreiben?

Coach B: Ich denke, die Voraussetzung ist das Bestehen von Bronze. Aber ich stimme zu, das war schön formuliert, Ryan! Und ich denke, das hilft uns auch, die Verwendung der Minimum/Maximum-Formulierung im ursprünglichen Problem zu schätzen: Es ist viel prägnanter.

Das Problem lautete: “Bestimmen Sie, welche Fackel entfernt werden kann, sodass die maximale Entfernung zwischen zwei benachbarten verbleibenden Fackeln minimal ist.” Und wir mussten das als “Betrachte alle möglichen resultierenden größten Entfernungen und wähle die kleinste davon” verstehen. Die Minimum/Maximum-Formulierung hier passt in viele Optimierungsprobleme, wie Sie in den Übungsaufgaben sehen werden, während die Tiki-Fackeln-Formulierung nur in diesem speziellen Fall passt. Aber es hat Spaß gemacht, es neu zu formulieren, also danke nochmal, Ryan!

TIPP: Wenn Sie einen Satz wie diesen sehen: “sodass die maximale Entfernung zwischen zwei benachbarten verbleibenden Fackeln minimal ist,” sagt Ihnen das wahrscheinlich, dass es sich um ein Optimierungsproblem handelt. Speziell dieser Problemtyp wird als Minimax-Problem bezeichnet. Ja, alles in einem Wort, “Minimax.” Eine Kombination aus “Minimum” und “Maximum”.

ALGORITHMUS

Coach B: Nun, irgendwelche Bedenken bezüglich spezieller Fälle, oder sind wir bereit, zum Algorithmus überzugehen?

Mei: Haie sind immer bereit. Ich bin bereit, es zu versuchen.

Coach B: Das ist die Einstellung! Los, Mei!

Mei nimmt den Marker und schreibt Listing 5.1.

Mei: Zuerst durchlaufe ich alle relevanten Tiki-Fackeln. Denken Sie daran, wir müssen die erste und letzte überspringen. Für jede davon habe ich eine Schleife über alle verbleibenden Tiki-Fackeln und berechne die Entfernung zwischen den benachbarten. Ich berechne einfach die Entfernung zum linken Nachbarn und behalte den größten Wert davon bei.

Auflistung 5.1 Tiki-Fackeln

```
int min_max_distance = INT_MAX;
int min_max_location;

for (int tiki_removed = 1; tiki_removed < N - 1; ++tiki_removed) {
    int max_dist = 0;
    int dist = 0;

    for (int i = 1; i < N; ++i) {
        if (i == tiki_removed) continue;
        if (i == tiki_removed + 1) {  // Are we to the right of the removed torch?
            // Yes: Our left neighbor is the previous one.
            dist = tiki_location[i] - tiki_location[i - 2];
        }
        else {
            // No: Distance from the left neighbor.
            dist = tiki_location[i] - tiki_location[i - 1];
```

```
        }
        max_dist = max(max_dist, dist);
    }

    if (max_dist < min_max_distance) {
        min_max_distance = max_dist;
        min_max_location = tiki_removed;
    }
}
```

Rachid: Ich verstehe, warum du die erste Schleife von 1 bis $N-1$ gemacht hast. Das liegt daran, dass du die erste und letzte Fackel vermeiden wolltest. Aber warum überspringst du die Fackel Nummer 0 in der inneren Schleife? Du gehst nur von $i = 1$ bis N.

Mei: Wenn ich die Entfernung berechne, berechne ich sie von der aktuellen Fackel zu ihrem Nachbarn links. Die allererste Fackel hat keinen Nachbarn links, daher überspringe ich sie. Ergibt das Sinn?

Rachid: Oh, ich verstehe. Danke.

Coach B: Sehr gut. Irgendwelche Kommentare?

Das Team scheint mit dem Code zufrieden zu sein.

Coach B: Sehr gut. Eigentlich habe ich noch eine Frage, bevor wir mit diesem Problem weitermachen. Irgendwelche Gedanken dazu, wie die Zeitkomplexität dieses Algorithmus sein könnte?

Stille im Raum. Komplexität ist, nun ja, komplex.

Ryan: Ich kann es versuchen. Wenn die Anzahl der Tiki-Fackeln N ist, dann ist dies unsere Basis, um über die Ordnung des Problems zu sprechen. Nun, wir machen eine verschachtelte Schleife über alle Tiki-Fackeln, das bedeutet, wir gehen über N^2 Fälle. Das bedeutet, unsere Zeitkomplexität ist $O(N^2)$. Ist das... richtig?

Ryan verstummt, unsicher.

Coach B: Sehr gut, Ryan! Das Einzige, was fehlt, ist etwas mehr Selbstbewusstsein in deiner Antwort! Kannst du es mit mehr Selbstbewusstsein sagen?

Ryan spricht lauter.

Ryan: Unsere Zeitkomplexität ist $O(N^2)$!

Das Team lacht.

Coach B: Richtig! Sehr schön. Wir werden es jetzt nicht versuchen, aber ich möchte erwähnen, dass es eine Lösung für dieses Problem mit einer Zeitkomplexität von nur $O(N)$ gibt. Ich lade euch ein, dieses Problem noch einmal zu betrachten, nachdem wir über die Beschleunigung von Suchalgorithmen gesprochen haben.

Mei: Wow, das klingt unmöglich. Kannst du uns wenigstens einen Hinweis geben?

Coach B: Ich möchte euch jetzt wirklich nicht verwirren, also machen wir es so: Ich werde den Code, mit Kommentaren und Erklärungen, auf der Seite des Clubs hinterlassen. Aber dies ist ein guter Punkt, um zu betonen: In Bronze müsst ihr nicht unbedingt den effizientesten Algorithmus finden, um ein Problem zu lösen. Wir werden sehen, dass ihr in einigen Fällen euren Algorithmus beschleunigen müsst, aber das ist nicht immer der Fall. Wenn ihr eine Lösung habt und sie alle Testfälle besteht, solltet ihr zum nächsten Problem übergehen! Also, in unserem Fall habt ihr alle Testfälle bestanden, wir können weitermachen!

Das Team jubelt.

Coach B: Okay. Ich glaube, das beendet unser erstes Suchproblem! Sehr schön. Im Prozess haben wir eine gängige Phrase für Minimum/Maximum in Optimierungsproblemen gelernt. Dann haben wir eine erschöpfende Suche durchgeführt: Wir haben versucht, jede der relevanten Tiki-Fackeln zu entfernen und die beste zum Entfernen gefunden. Und obendrein hat uns Ryan geholfen, die Zeitkomplexität dieses Algorithmus zu analysieren, mit Selbstbewusstsein. Gut gemacht!

Das Team beginnt zu packen, bereit, sich zu verabschieden.

Coach B: Ich werde ein paar Suchprobleme auf die Seite des Clubs stellen. Ich werde auch ein paar Hinweise streuen, wie üblich. Oh, und ich werde auch die $O(N)$ Lösung hinlegen, wenn ihr sehen wollt, wie es gemacht wird. Bis nächste Woche!

TIPP: Wenn ihr zu lange an einem Problem festhängt, könnt ihr immer einen Blick auf die Lösung werfen und sie dann selbst schreiben. Es ist besser, einen großen Hinweis zu bekommen, als entmutigt zu werden. Es ist ein Lernprozess.

EPILOG

Bei erschöpfenden Suchen untersuchen wir alle möglichen Optionen. Das kann zu zeitaufwendig sein, aber auf Bronze-Niveau ist es oft ein gültiger Ansatz. Trotzdem gibt es selbst bei erschöpfenden Suchen Möglichkeiten, Rechenzeit zu sparen. Wir werden später in diesem Kapitel, wenn wir über Beschleunigung sprechen, Wege sehen, wie man Rechenzeit sparen kann.

WORTSCHATZ Ecke: **OPTIMIERUNG** ist der Prozess, etwas in seine beste oder optimale Position zu bringen. Als lustige Anmerkung: Die Wörter "optimieren" und "Optimierung" entstanden aus dem Wort "Optimist." Und Mei hier ist eine Optimistin: eine Person mit einer hoffnungsvollen und positiven Einstellung, die sich auf die besten aller möglichen Optionen konzentriert. Optimisten sehen immer die Sonnenseite und erwarten, dass die besten Dinge passieren. Wie Treibstoffkosten zu sparen, während Waikiki Beach gut beleuchtet und sicher bleibt.

ÜBUNGSAUFGABEN

Hinweise und vollständige Lösungen zu den Aufgaben finden Sie auf der Seite des Clubs: http://www.usacoclub.com

1. USACO 2014 Januar Bronze Problem 1: Skikurs-Design

   http://usaco.org/index.php?page=viewproblem2&cpid=376

   a. Können Sie das Problem als eine Suchfrage formulieren? Wonach suchen Sie?

   b. Hinweis: Wir suchen nach dem Bereich der Hügelhöhen, der keine Änderungen benötigt.

   c. Hinweis: Wenn Sie die niedrigste Hügelhöhe im zulässigen Bereich kennen, können Sie die Kosten des Skikurses berechnen?

   d. Großer Hinweis: Sie werden die Höhe des niedrigsten zulässigen Hügels durchsuchen. Angesichts dessen können Sie die Kosten des Skikurses berechnen. Der niedrigste Hügel, den Sie in Betracht ziehen sollten, ist die niedrigste Hügelhöhe im bereitgestellten Eingabewert, und der höchste Wert, den Sie in Betracht ziehen sollten, ist der höchste Hügel (möglicherweise minus 17).
2. USACO 2016 Open Bronze Problem 1: Diamantensammler

   http://usaco.org/index.php?page=viewproblem2&cpid=639

   a. Können Sie die Ähnlichkeit zum "Skikurs-Design"-Problem (2014 Januar Bronze Problem 1) erkennen?

   b. Hinweis: Wenn Sie die Größe des kleinsten Diamanten, den Sie anzeigen können, kennen, können Sie bestimmen, wie viele Diamanten präsentiert werden?
3. USACO 2019 Dezember Bronze Problem 1: Kuhgymnastik

   http://usaco.org/index.php?page=viewproblem2&cpid=963

   a. Das Anordnen der Eingabedaten in einem zweidimensionalen Array würde die Sache erleichtern.

   b. Dann handelt es sich um eine erschöpfende Suche über alle möglichen Paare.
4. USACO 2019 Dezember Bronze Problem 2: Wo bin ich?

   http://usaco.org/index.php?page=viewproblem2&cpid=964

   a. Suche über Zeichenfolgen.

   b. Eine erschöpfende Suche über alle Teilzeichenfolgen würde zeitlich machbar sein.

5.2. Suchbereich

Problem 5.2: Bessie sucht Muscheln am Strand

5.3. Bereichsaufzählung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Problem 5.3: Vulkane überqueren

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

5.4. Beschleunigung der Suche

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Problem 5.4: Luaus und Leis

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

5.5. Gierige Algorithmen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Problem 5.5: Kajakfahren

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Beispielproblem: Das Rucksackproblem (wir verwenden ein Gepäckstück)

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

5.6. Zusammenfassung

- **Suchprobleme** können schwer zu identifizieren sein. Sie treten in vielen Formen auf und werden oft als Optimierungsprobleme präsentiert. Bei Optimierungsproblemen suchen wir nach einem Parameter eines Prozesses, um das beste Ergebnis zu erzielen.
- Um ein Suchproblem zu identifizieren, stellen Sie sich die folgenden Fragen:
  - Könnten Sie verschiedene Werte ausprobieren und sehen, welcher am besten funktioniert? Wenn das möglich erscheint, dann können Sie vielleicht über all diese Werte suchen.
  - Würde ein Orakel Ihnen erlauben, das Problem zu lösen? Das heißt, wenn jemand erscheinen würde, der Ihnen auf magische Weise den Wert des Parameters offenbart, könnten Sie dann bewerten, wie gut dieser Wert ist? Wenn ja, dann können Sie eine erschöpfende Suche aufbauen, die alle möglichen Werte des Orakels durchläuft.
  - Was ist die erste Entscheidung, die Sie treffen müssten, um das Problem zu lösen? Zum Beispiel, die schwerste Kuh zu nehmen. Wenn Sie diese Art von Entscheidung immer wieder treffen würden, würde das Sie zur Lösung führen? Wenn ja, dann ist vielleicht ein gieriger Algorithmus möglich.
- Auf Bronze-Niveau lösen wir Suchprobleme mit zwei Haupttypen von Algorithmen: erschöpfende Suchen und gierige Algorithmen.
- **Erschöpfende Suchen** bewerten alle möglichen Optionen und wählen die beste aus.
  - Bestimmen Sie den Bereich des Problems. Dies sind die Werte, über die Sie suchen werden.
  - Zählen Sie den Bereich auf. Wie werden Sie den Bereich Element für Element durchlaufen?
- **Beschleunigung erschöpfender Suchen**. Dies tun wir auf zwei Arten:
  - Wählen Sie einen kleineren Bereich. Auf diese Weise müssen Sie weniger Optionen untersuchen.
  - Beschleunigen Sie die Bewertung jeder Option.
- **Gierige Algorithmen** basieren darauf, bei jedem Schritt einfache und schnelle Entscheidungen zu treffen.
  - Sie sind in der Regel sehr schnell.
  - Sie garantieren nicht unbedingt eine optimale Lösung (sie funktionieren nur bei einigen Problemen!).
  - Sie können ein besseres Ergebnis mit einem gierigen Algorithmus erzielen, wenn Sie einen neuen entwerfen, der eine andere gierige Entscheidung verwendet.

Kapitel 6. Geometriekonzepte

6.1. Eine Dimension: Linien

6.1.1. Ort, Länge und Entfernung

Problem 6.1: Gehen oder Busfahren?

6.1.2. Zwei Liniensegmente

Problem 6.2: Golden Gate Brückenpatrouille

6.2. Zwei Dimensionen: Rechtecke

6.2.1. Lage und Fläche

Problem 6.3: Um den Zaun herum

6.2.2. Zwei Rechtecke

Problem 6.4: Zwei Decken für das Picknick

6.3. Über neunzig Grad hinaus

6.3.1. Kreise

Problem 6.5: Sitze in der Arena

6.3.2. Allgemeine Formen

Problem 6.6: Pfad um den See

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

6.4. Zusammenfassung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 7. Zeichenketten

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

7.1. Zeichenketten als Sequenzen von Zeichen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

7.1.1. Darstellung von Zeichen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

7.1.2. Probleme mit Zeichen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Problem 7.1: Doppeltüren

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

7.2. Zeichenketten als Wörter

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Problem 7.2: Nach Alter ordnen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

7.3. Zeichenketten als Objekte

7.3.1. String-Algorithmen

Problem 7.3: Bestes Armband

7.3.2. Lexikographische Ordnung

7.4. Zusammenfassung

Kapitel 8. Ad-hoc-Probleme und fortgeschrittene Techniken

8.1. Die Vorwärts-Rückwärts-Technik

Problem 8.1: Doppeltüren-Reparatur

8.2. Fokussierung auf wichtige Ereignisse

Problem 8.2: Haie und Mondfische

8.3. Bäume

Problem 8.3: Das Restaurant am Ende des Bauernhofs

8.4. Dictionaries und Dynamische Arrays

8.5. Zusammenfassung

Teil III. Wettbewerbstag und darüber hinaus

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 9. Wettbewerbstag

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

9.1. Eine Woche davor

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

9.2. Der Wettbewerb

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

9.3. Nach dem Wettbewerb

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

9.4. Zusammenfassung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Kapitel 10. Jenseits von USACO Bronze

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

10.1. Silber und darüber hinaus

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

10.2. Dein erstes USACO-Silber-Problem lösen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

10.3. Zusammenfassung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Teil IV. Anhang

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Anhang A. Liste aller USACO Bronze Probleme

USACO Probleme

Saison 2012-2013

Saison 2013-2014

2014-2015 Saison

2015-2016 Saison

2016-2017 Season

2017-2018 Season

Saison 2018-2019

Saison 2019-2020

2020-2021 Saison

2021-2022 Saison

2022-2023 Saison

2023-2024 Saison

Codeforces Probleme

CSES Probleme

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

Anhang B. Übung über USACO hinaus

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

B.1. Online-Leitfäden und Live-Coaching

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

B.2. Online-Übung und -Wettbewerbe

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.

B.3. BÜCHER

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/start_competitive_programming-de gekauft werden.