Aufgabe 3: Eisbudendilemma

Teilnahme-ID: 56860

Bearbeiter/-in dieser Aufgabe:
Christopher Besch

10. April 2021

Inhaltsverzeichnis

1 Ein Wort über die Graphiken

Alle in dieser Dokumentation verwendeten Darstellungen verwenden einheitliche Symbole:

- Der See ist als schwarzer Kreis dargestellt.
- Die Häuser sind verschieden gefärbte Rechtecke, deren Adresse außerhalb des Kreises stehen:
  - Rot: Das Haus stimmt gegen eine Verlegung der Eisdielen.
  - Grün: Es stimmt für eine Verlegung.
  - Andere Farben werden verwendet, um bestimmte Häuser herauszuheben. Die jeweiligen Bedeutungen werden darstellungsspezifisch angegeben.
- Die Adressen sind aufsteigend im Uhrzeigersinn angeordnet. Adresse 0, die Dorfkirche befindet sich oben.
- Blaue Kreuze stellen die Positionen des Test-Arrangements dar und
- blaue Kreise die des Check-Arrangements. In beiden Fällen stehen die Adressen innerhalb des Kreises.

2 Lösungsidee

Das Ziel ist es, ein Arrangement bestehend aus drei Positionen für Eisdielen zu generieren, das in einer Abstimmung durch kein anderes Arrangement abgelöst werden kann. Diese Arrangements werden stabil genannt. Hierzu darf die Eisbudendistanz, die Strecke zwischen einem beliebigen Haus und der nächsten Eisbude, von nicht mehr als der Hälfte der Hauser durch ein anderes Arrangement verkürzt werden. Wäre dies der Fall, würde die Ablösung mehr Ja- als Nein-Stimmen erhalten. Hieraus geht hervor, dass für eine optimale Lösung alle möglichen Arrangements (Diese werden Test-Arrangement genannt.) auf Stabilität überprüft werden müssen. Um die Stabilität zu bestimmen, muss das Test-Arrangement mit

allen möglichen anderen Arrangements (Check-Arrangement genannt) verglichen werden. Wenn auch nur ein einziges Check-Arrangement gefunden wird, das mehr Ja- als Nein-Stimmen erhält, ist das getestete Test-Arrangement instabil. Es lässt sich leicht erkennen, dass dieser Algorithmus, der Durchgang aller möglichen Test-Arrangements, mit einer Laufzeit von $O(n^6)$ nicht verwendbar ist.

Als Versuch der Optimierung werden bevor sie getestet werden alle Arrangement sortiert. Hierzu wird für jedes mögliche Arrangement ein Score berechnet. Dieser entspricht der durchschnittlichen Eisbudendistanz aller Häuser. Nun stellt sich heraus, dass stabile Arrangements einen der niedrigsten Scores aller Arrangements aufweisen. Dies lässt sich damit erklären, dass je kleiner die Eisbudendistanz eines Hauses in einem Test-Arrangement ist, desto weniger Check-Arrangement existieren, die eine noch geringere Eisbudendistanz für das Haus generieren. Wenn die Eisbudendistanz beispielsweise 0 beträgt, existiert kein einziges Check-Arrangement, dem dieses Haus eine Ja-Stimme geben würde, da eine geringere Eisbudendistanz nicht möglich ist und ein Haus bei gleichbleibender Eisbudendistanz immer gegen einen Wechsel stimmt. Dies ist in Abbildung 1 gezeigt. Wenn die Eisbudendistanz den maximalen Wert, dem

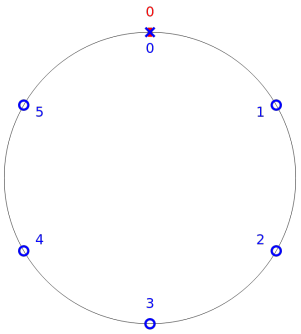
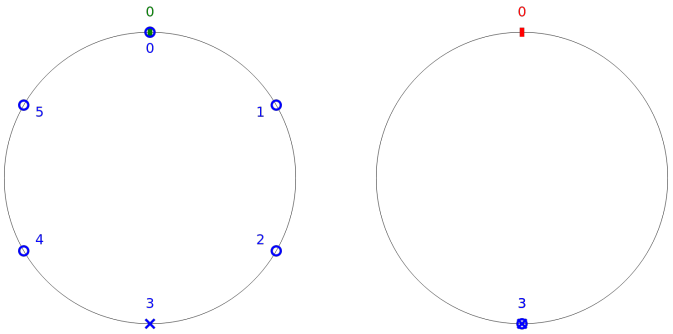


Abbildung 1: Das einzige Haus ist zufrieden mit der einzigen Eisdiele und lehnt jegliche Veränderung ab. Die Eisbudendistanz beträgt 0.

halben Umfang des Sees, entspricht, stimmt es für alle Check-Arrangements (Abbildung 3a), abgesehen von denen, die die Eisbudendistanz nicht verändern (Abbildung 3b). Die durchschnittliche Eisbudendi-

Abbildung 2: Unzufriedene Häuser



(a) Das Haus ist maximal unzufrieden, weshalb es für fast jede Verlegung stimmt. Jeder Kreis repräsentiert eine anderes Check-Arrangement, die alle von dem Haus angenommen werden.

(b) Dies ist der einzige Fall, in dem das Haus trotz seiner extremen Unzufriedenheit gegen eine Verlegung stimmt.

stanz lässt sich dementsprechend als „Zufriedenheitsgrad“ des Dorfes interpretieren. Je höher er ist, desto unwahrscheinlicher wird eine Verlegung durchgesetzt.

Allerdings muss dieser Wert nicht zwangsweise mit der Stabilität eines Arrangements übereinstimmen, was beispielsweise in Abbildung 4 gezeigt wird. Hieraus geht hervor, dass die durchschnittliche Eisbu-

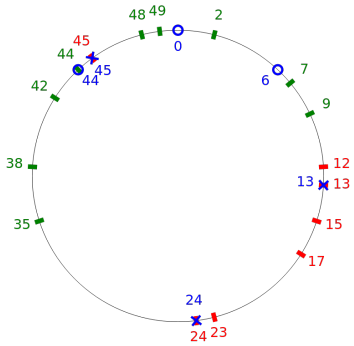


Abbildung 4: Trotz der für dieses Beispiel, *eisbuden3.txt*, minimalen durchschnittlichen Eisbudendistanz von $3,3125$ stimmen mehr Häuser für eine Verlegung.

dendistanz nur eine Näherungslösung liefert. Trotzdem kann sie zur Generierung eines Satzes an Test-Arrangements verwendet werden, die anschließend von dem bereits genannten Algorithmus auf Stabilität überprüft werden.

Um die Stabilität eines Test-Arrangements zu berechnen, müssen alle möglichen Check-Arrangements durchgegangen werden. Es wird nur eine einziges Check-Arrangement gesucht, das das Test-Arrangement schlagen kann. Daher können zwei Optimierungen getroffen werden:

1. Eisdielen sollten nicht übereinander liegen, da bei der Aufsplittung zweier aufeinanderliegender Eisdielen die Eisbudendistanz keines Häuses vergrößert wird.

2. Alle Dopplungen sind unnötig, da die Reihenfolge der Eisdielen für die Stimmen der Häuser irrelevant sind.

Deshalb darf die Bedingung gelten, dass die Adresse der zweiten Eisdiele größer als die der ersten und kleiner als die der dritten ist. Hieraus folgt, dass der See in drei Sektoren unterteilt ist (Abbildung 5).

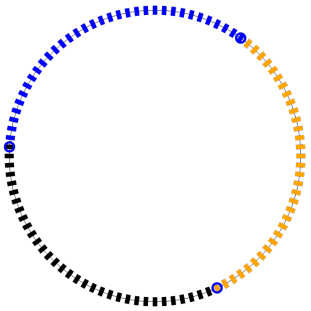


Abbildung 5: Einteilung in Sektoren

Es zeigt sich, dass die Stimme der Häuser innerhalb eines Sektors ausschließlich durch die Größe und Position des Sektors und die in dem Sektor befindlichen Eisdielen des Test-Arrangements determiniert sind.

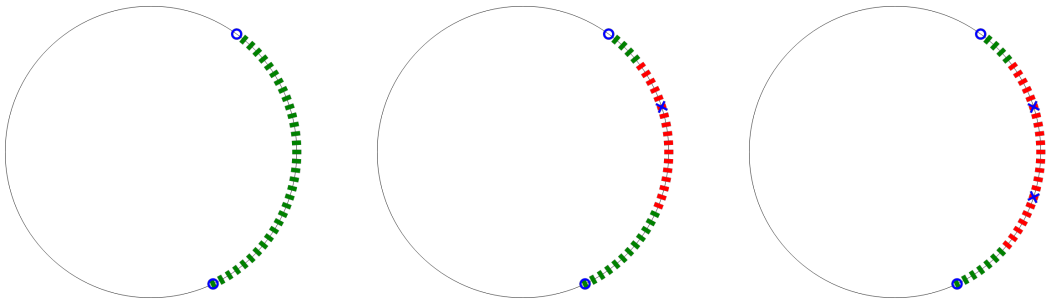
- Alle Test-Eisdielen außerhalb des Sektors sind von den Häusern immer weiter entfernt als die Ränder des Sektors, weshalb sich die ehemalige Eisbudendistanz ohne Test-Eisdielen innerhalb des Sektors nie verkürzen wird.

- Genauso ist für jedes Haus im Sektor eine der Sektorgrenzen immer die nächste Check-Eisdiele. Dementsprechend ist die Position der dritten Check-Eisdiele ebenfalls irrelevant. da immer nur die Eisdielendistanz betrachtet wird.

Aus diesen beiden Punkten geht hervor, dass ausschließlich die Ränder des Sektors und die sich im Sektor befindlichen Test-Eisdielen Einfluss auf die Stimme eines Hauses in dem Sektor hat.

Nun wird sich die Verteilung der Ja- und Nein-Stimmen betrachtet. Es existieren nur drei verschiedene Möglichkeiten für die Anordnung der Test-Eisdielen in einem Sektor, In dem letzten Fall von mehr als

Abbildung 6: Test-Eisdielen Anordnung im Sektor



(a) Keine Test-Eisdiele (b) Eine Test-Eisdiele (c) Zwei oder mehr Test-Eisdielen (nur die Äußersten dargestellt)

zwei Test-Eisdielen im Sektor, müssen nur die Äußersten betrachtet werden, da zwischen diesen beiden keine Check-Eisdiele stehen kann. Dies geht aus der Definition eines Sektors hervor. Somit stimmen immer alle Häuser zwischen diesen gegen eine Verlegung, unabhängig von der Menge and Test-Eisdielen.

Um die Anzahl an Nein- beziehungsweise Ja-stimmenden Häusern zu bestimmen wird müssen zwei verschiedene Methoden verwendet werden:

- Wenn sich keine Test-Eisdiele in dem Sektor befindet, stimmt kein Haus gegen eine Verlegung.
- In allen anderen Fällen befinden sich die einzigen Ja-stimmenden Häuser in den beiden Bereichen zwischen einem Rand un der nächsten Test-Eisdiele. Wenn diese Bereiche in der Mitte geteilt wird, befinden sich alle Ja-stimmenden Häuser in der Hälfte, die näher am Rand ist. Alle Nein-stimmenden Häuser liegen in der anderen Hälfte. Bei einer ungeraden Anzahl an Adressen in diesem Bereich wird der Nein-stimmenden Hälfte eine Adresse mehr zugeteilt, da bei gleichbleibender Eisdielendistanz das Haus Nein stimmen.

Auf diese Weise können effizient die Bereiche berechnet werden, in denen alle sich befindlichen Häuser Nein stimmen.

Es ergibt sich ein weiterer Vorteil: Um alle Position für die Check-Eisdielen durchzugehen, können zuerst zwei Positionen für die ersten beiden Eisdielen festgesetzt werden und daraufhin alle möglichen Positionen für die dritte Eisdiele verwendet werden, die die genannten Bedingungen erfüllt. Die verschiedenen Optionen für die ersten beiden Check-Eisdielen müssen in einem übergeordneten Schritt durchgegangen werden. Somit ist die Größe und Position des ersten Sektors bei vielen Durchläufen gleich. Wenn bereits mindestenz die Hälfte der Häuser sich in diesem Sektor befinden und gegen eine Veränderung stimmen, ist das Check-Arrangement unfähig, das Test-Arrangement abzulösen. Somit kann in diesem Fall die Suche nach Positionen für die dritte Check-Eisdiele übersprungen werden und die Wahl der ersten beiden Check-Eisdielen sofort geändert werden.

3 Umsetzung

Die Lösungsidee wird in C++ implementiert. Der Übersichtlichkeit halber ist das Programm in drei Dateien unterteilt,

- **main.cpp** liest die Eingabe und gibt die Ergebnisse aus,
- **utils.h** enthält verwendete Structs und generell anwendbare Funktionen und Makros und
- **calculate_stable.h** enthält die eigentlichen Funktionen zur Bestimmung der stabilen Arrangements.

Einlese der Eingabedatei Als erster Schritt wird in der Funktion **read_file** die Eingabedatei gelesen. Hierbei wird überprüft, ob die Eingabedatei dem gegebenen Format entspricht, wenn nicht wird das Programm abgebrochen. Hierzu wird ein Makro **raise_error()** verwendet, das die Ausführung des Programms abbricht und eine möglichst informative Fehlermeldung zurückgibt.

4 Beispiele

Nun wird das Programm mit allen Beispieldateien ausgeführt.

eisbuden1.txt Mit der Eingabe
20 7
0 2 3 8 12 14 15
gibt das Programm aus, dass es keine stabilen Positionen gibt.

eisbuden2.txt Mit der Eingabe
50 15
3 6 7 9 24 27 36 37 38 39 40 45 46 48 49
gibt das Programm die Menge an stabilen Positionen aus:
45

eisbuden3.txt Mit der Eingabe
50 16
2 7 9 12 13 15 17 23 24 35 38 42 44 45 48 49
gibt das Programm die Menge an stabilen Positionen aus:
2, 3, 4, 5, 6 und 7

eisbuden4.txt Mit der Eingabe
100 19
6 12 23 25 26 28 31 34 36 40 41 52 66 67 71 75 80 91 92
gibt das Programm die Menge an stabilen Positionen aus:
34

eisbuden5.txt Mit der Eingabe
247 24
2 5 37 43 72 74 83 87 93 97 101 110 121 124 126 136 150 161 185 200 201 230 234 241
gibt das Programm die Menge an stabilen Positionen aus:
93, 94, 95, 96 und 97

eisbuden6.txt Mit der Eingabe
437 36
4 12 17 23 58 61 67 76 93 103 145 154 166 170 192 194 209 213 221 225 239 250 281 299 312 323 337 353 383 385 388 395 405 407 412 429
gibt das Programm aus, dass es keine stabilen Positionen gibt.

eisbuden7.txt

4.1 Eigene Beispiele

myeisbuden0.txt Mit der Eingabe
12 4
0 3 6 9
gibt das Programm die Menge an stabilen Positionen aus:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 und 11
Dies ergibt Sinn, da die Häuser gleichmäßig verteilt sind, womit nie eine Mehrheit gegen jegliche Positionen gefunden werden kann.

myeisbuden0.txt Mit der Eingabe
10 0
0, 1, 2, 3, 4, 5, 6, 7, 8 und 9
Dies ergibt Sinn, da keine Häuser vorhanden sind, die für eine Umlegung stimmen könnten.

5 Quellcode

Dies sind die wichtigsten Funktionen:

```
int get_distance(int circumference, int place_a, int place_b)
{
    int direct_distance = std::abs(place_a - place_b);
    // take shortest way, direct or the othe rdirection
    return std::min(direct_distance, circumference - direct_distance);
}

bool vote(int circumference, int house_place, int old_place, int new_place)
{
    // is new place better?
    if (get_distance(circumference, house_place, new_place) <
    get_distance(circumference, house_place, old_place))
        return true;
    return false;
}

bool is_stable(int circumference, std::vector<int> &houses, int test_place)
{
    // would any other place win an election against test_place?
    for (int other_place = 0; other_place < circumference; other_place++)
    {
        int trues = 0;
        for (int house : houses)
            if (vote(circumference, house, test_place, other_place))
                trues++;

        if (trues > houses.size() - trues)
            return false;
    }
    return true;
}

std::vector<int> get_stabel_places(int circumference, std::vector<int> &houses)
{
    // go thorugh all possible places
    std::vector<int> result;
    for (int test_place = 0; test_place < circumference; test_place++)
        if (is_stable(circumference, houses, test_place))
            result.push_back(test_place);
    return result;
}
```