Aufgabe 5: Das ägyptische Grabmal

Team-ID: 00433

Team-Name: Was?

Bearbeiter dieser Aufgabe:
Mathieu de Borman

15. Oktober 2024

Inhaltsverzeichnis

1 Lösungsidee

In dem ersten Schritt wird das Grabmal so lange simuliert bis das Ziel erreicht ist. Anschließend wird das Grabmal rückwärts Simuliert, wobei immer der kürzeste Weg gemerkt wird.

1.1 Kürzeste Zeit

Mit jeder Iteration schreitet die Zeit voran und es wird für jeden Quader überprüft, ob er sich bewegt. Anschließend wird festgehalten, welche Positionen dadurch vom Ausgang aus neu zugänglich sind. Sobald die letzte Position zugänglich ist, wird die Simulation gestoppt.

1.2 Einfachster Weg

Es wird das Gleiche wie bei dem vorherigen Teil simuliert mit der Änderung, dass die Zeit zurückgespult und der Weg vom Grabmal zum Ausgang gesucht wird. Jetzt wird außerdem der kürzeste Weg erfasst, den man braucht, um eine Position vom Grabmal aus zu erreichen.

1.3 Laufzeitanalyse

Die Laufzeitkomplexität hängt von zwei Parametern ab, nämlich der Anzahl der Quader n und der minimal benötigten Zeit m. Das Programm hat eine Laufzeitkomplexität von $O(n \cdot m)$ bei einer Speicherkomplexität von $O(n)$.

2 Umsetzung

Die Lösungsidee ist in JavaScript implementiert.

2.1 Einlesen

Die Eingabe wird über ein Eingabefenster (prompt) ausgelesen. Anschließend werden alle ungültigen Zeichen rausgefiltert (für den Fall, dass keine saubere Eingabe gemacht wurde) und die Gültigen in Zahlen umgewandelt. Danach wird überprüft, ob es ein gültiges Grabmal ist.

2.2 Kürzeste Zeit

Der Weg wird erst im nächsten Abschnitt berechnet, um Zeit zu sparen. Anfangs werden die Quader und die ereichten Felder deklariert. Das erste ist offen und erreicht, da sich das erste Element außerhalb des Grabmals befindet. Anschließend wird über eine For-Schleife das minimale Intervall einer Tür ermittelt. In der While Schleife springt die Zeit immer wieder zu dem Zeitpunkt, zu welchem sich der nächste Quader bewegt. Anschließend wird für jeden Quader ermittelt, ob er sich bewegt. Wenn er sich nicht bewegt, wird sich gemerkt, wann der Quader sich das nächste Mal bewegt. Schließt der Quader sich, so wird gespeichert, dass diese Position nicht mehr erreichbar ist. Öffnet er sich, so wird überprüft, ob diese Öffnung neue Positionen erreichbar macht. Sobald die letzte Position erreicht wurde, wird die While-Schleife verlassen.

2.3 Einfachster Weg

Dies ist das Inverse vom vorherigen Abschnitt (rückwärtslaufende Zeit, Berechnung vom Ausgang zum Eingang hin) mit der Änderung, dass der kürzeste Weg gespeichert wird. Immer wenn sich in der While-Schleife ein Quader öffnet und dadurch neue Positionen errreichbar werden, wird sich gemerkt, wie der Weg bis dorthin war. Es wird also bei jeder Bewegung der Weg zum Ursprungsort sowie die Bewegung zur neuen Position gespeichert. Das wird auch gemacht, wenn eine Position eine längere Wegbeschreibung hat als eine andere direkt erreichbare Position.

2.4 Ausgeben

Die Wegbeschreibung von der ersten Position wird formatiert und in die Konsole ausgegeben.

3 Beispiele

Wir rufen nun das JavaScript-Programm mit den verschiedenen BWINF-Eingabedateien auf. Diese Dateien sind in demselben Ordner wie die Programmdatei. Das Programm wird mit Hilfe des Browsers ausgeführt. In dem Dialogfeld kann man dann den Plan des Grabmals eingeben. Die Ausgabe kann man in der Konsole auslesen. Das Programm terminiert für alle getesteten Eingaben in weniger als 2s auf einem gewöhnlichen PC.

3.1 Lösungen

3.1.1 grabmal0.txt

Daten während der Ausführung						
Quader →	1		2		3	
Zeit ↓	open	reached	open	reached	open	reached
5	true	true	false	false	false	false
8	true	true	true	true	false	false
10	false	false	true	true	false	false
12	false	false	true	true	true	true

```
Warte 9 Minuten, laufe in den Abschnitt 2,
warte 3 Minuten, laufe zum Grabmal.
```

3.1.2 grabmal1.txt

Daten während der Ausführung										
Quader →	1		2		3		4		5	
Zeit ↓	open	reached	open	reached	open	reached	open	reached	open	reached
7	false	false	false	false	true	false	false	false	false	false
9	false	false	false	false	true	false	true	false	false	false
13	false	false	true	false	true	false	true	false	true	false
14	false	false	true	false	false	false	true	false	true	false
17	true	true	true	true	false	false	true	false	true	false
18	true	true	true	true	false	false	false	false	true	false
21	true	true	true	true	true	true	false	false	true	false
26	true	true	false	false	true	true	false	false	false	false
27	true	true	false	false	true	true	true	true	false	false
28	true	true	false	false	false	false	true	true	false	false
34	false	false	false	false	false	false	true	true	false	false
35	false	false	false	false	true	true	true	true	false	false
36	false	false	false	false	true	true	false	false	false	false
39	false	false	true	true	true	true	false	false	true	false
42	false	false	true	true	false	false	false	false	true	false
45	false	false	true	true	false	false	true	false	true	false
45	false	false	true	true	false	false	true	false	true	false
49	false	false	true	true	true	true	true	true	true	true

```
Warte 25 Minuten, laufe in den Abschnitt 3,
warte 2 Minuten, laufe in den Abschnitt 4,
warte 8 Minuten, laufe in den Abschnitt 3,
warte 6 Minuten, laufe in den Abschnitt 2,
warte 8 Minuten, laufe zum Grabmal.
```

3.1.3 grabmal2.txt

```
Warte 259999 Minuten, laufe in den Abschnitt 3,
warte 20012 Minuten, laufe in den Abschnitt 4,
warte 79988 Minuten, laufe in den Abschnitt 3,
warte 60018 Minuten, laufe in den Abschnitt 2,
warte 70004 Minuten, laufe zum Grabmal.
```

3.1.4 grabmal3.txt

```
Warte 23 Minuten, laufe in den Abschnitt 6,
warte 12 Minuten, laufe zum Grabmal.
```

3.1.5 grabmal4.txt

```
Warte 3314897 Minuten, laufe in den Abschnitt 5,
warte 44606 Minuten, laufe in den Abschnitt 6,
warte 42496 Minuten, laufe in den Abschnitt 8,
warte 30898 Minuten, laufe zum Grabmal.
```

3.1.6 grabmal5.txt

```
Warte 43865195 Minuten, laufe in den Abschnitt 7,
warte 4124 Minuten, laufe in den Abschnitt 9,
warte 2580 Minuten, laufe in den Abschnitt 22,
warte 5076 Minuten, laufe in den Abschnitt 25,
warte 4768 Minuten, laufe in den Abschnitt 32,
warte 2328 Minuten, laufe in den Abschnitt 33,
warte 6840 Minuten, laufe in den Abschnitt 35,
warte 640 Minuten, laufe in den Abschnitt 39,
warte 1632 Minuten, laufe in den Abschnitt 42,
warte 1004 Minuten, laufe in den Abschnitt 47,
warte 3834 Minuten, laufe in den Abschnitt 49,
warte 5226 Minuten, laufe in den Abschnitt 50,
warte 3450 Minuten, laufe in den Abschnitt 53,
warte 4848 Minuten, laufe in den Abschnitt 55,
warte 7494 Minuten, laufe in den Abschnitt 58,
warte 4620 Minuten, laufe in den Abschnitt 63,
warte 4588 Minuten, laufe in den Abschnitt 64,
warte 54 Minuten, laufe in den Abschnitt 69,
warte 4790 Minuten, laufe in den Abschnitt 71,
warte 6316 Minuten, laufe in den Abschnitt 72,
warte 3260 Minuten, laufe in den Abschnitt 74,
warte 7194 Minuten, laufe in den Abschnitt 71,
warte 6718 Minuten, laufe in den Abschnitt 72,
warte 3644 Minuten, laufe in den Abschnitt 74,
warte 1330 Minuten, laufe in den Abschnitt 80,
warte 2366 Minuten, laufe in den Abschnitt 81,
warte 1092 Minuten, laufe in den Abschnitt 82,
warte 979 Minuten, laufe zum Grabmal.
```

4 Quellcode

```
//Lese die Eingabe aus
let input = (prompt("grabmal.txt") || "0").split(/[\s\n]+/)
let cuboid = []
//Wandle String in Zahlen um und filtere ungültige Zahlen raus
for (let i = 0; i < input.length; i++) {
  let maybe = Number(input[i]) || 0
  if (maybe != 0) {
    cuboid.push(maybe)
  }
}
//Wenn die Eingabe kein Grabmal war, dann stoppe das Programm
if (cuboid.length < 2) {
  throw new Error("Kein Grabmal")
}
const size = cuboid.length
//Alle Türen sind geschlossen
let open = Array(size).fill(false)
open[0] = true
//Keine Position wurde erreicht
```

```
let reached = [...open]
let time = 0
//Finde den Quader mit dem minimalen Intervall
let minIntervall = cuboid[1]
for (let i = 2; i < size; i++) {
  minIntervall = Math.min(minIntervall, cuboid[i])
}
let nextMove = minIntervall
//Solange das Grabmal nicht erreicht wurde
while (!reached[size - 1]) {
  //Springe zum nächsten Zeitpunkt bei dem sich ein Quader bewegt
  time += nextMove
  nextMove = minIntervall
  //Teste für jeden Quader ob er sich bewegt
  for (let i = 1; i < size; i++) {
    if (time % cuboid[i] == 0) {
      open[i] = !open[i]
      reached[i] = false
      //Wenn die position von links erreichbar ist,
      //sind alle positionen rechts davon bis zum nächsten geschlossenen Quader erreichbar
      if (reached[i - 1]) {
        for (let j = i; j < size && open[j] && !reached[j]; j++) {
          reached[j] = true
        }
        //Wenn die position von rechts erreichbar ist,
        //sind alle positionen links davon bis zum nächsten geschlossenen Quader erreichbar
      } else if (reached[i + 1]) {
        for (let j = i; j > 0 && open[j]; j--) {
          reached[j] = true
        }
      }
      //Wenn sich der Quader nicht bewegt, messe seine Zeit bis zur nächsten bewegung
    } else {
      nextMove = Math.min(nextMove, cuboid[i] - time % cuboid[i])
    }
  }
}
//Merke dir den Weg den du bei rückwärtslaufender Zeit nimmst hast
let way = Array(size)
//Vom Grabmal aus wuden nur die positionen erreicht,
//die man direkt vom Grabmal aus erreichen kann
reached = Array(size).fill(false)
for (let i = size - 1; i > 0 && open[i]; i--) {
  reached[i] = true
  way[i] = [time, i]
}
nextMove = 1
//Solange der weg nicht zurück gefunden wurde
while (!reached[1]) {
  //Springe zum nächsten Zeitpunkt in der Zeit zurück bei dem sich ein Quader bewegt
  time -= nextMove
  nextMove = minIntervall
  //Teste für jeden Quader ob er sich bewegt
  for (let i = size - 1; i > 0; i--) {
    if ((time + 1) % cuboid[i] == 0) {
      open[i] = !open[i]
      reached[i] = false
      //wenn die position nun erreichbar und offen ist
```

```
      //suche nach einer direkt erreichbaren position mit den wenigsten weg anweisungen
      if ((reached[i - 1] || reached[i + 1]) && open[i]) {
        let origin = reached[i - 1] ? way[i - 1] : way[i + 1]
        for (let j = i - 1; j < size && reached[j]; j--) {
          if (origin.length > way[j].length) {
            origin = way[j]
          }
        }
        for (let j = i + 1; j < size && reached[j]; j++) {
          if (origin.length > way[j].length) {
            origin = way[j]
          }
        }
        //Alle direkt erreichbaren positionen sind nun erreichbar
        //und haben eine minimale anzahl an weg anweisungen
        reached[i] = true
        way[i] = [...origin, time, i]
        for (let j = i - 1; j > 0 && open[j]; j--) {
          if (!reached[j] || origin.length < way[j].length) {
            reached[j] = true
            way[j] = [...origin, time, j]
          }
        }
        for (let j = i + 1; j < size && open[j]; j++) {
          if (!reached[j] || origin.length < way[j].length) {
            reached[j] = true
            way[j] = [...origin, time, j]
          }
        }
      }
      //Wenn sich der Quader nicht bewegt, messe seine Zeit bis zur nächsten bewegung
    } else {
      nextMove = Math.min(nextMove, (time + 1) % cuboid[i])
    }
  }
}
//Wandle die daten über den kürzesten weg zum Anfang in Anweisungen zum erreichen des Ziels um
//und gebe diese über die console aus
way[1].push(0)
let out = "W"
for (let i = way[1].length - 3; i > 1; i -= 2) {
  out += `arte ${way[1][i] - way[1][i + 2]} Minuten, laufe in den Abschnitt ${way[1][i - 1]},\nw`
}
console.log(`${out}arte ${way[1][0] - way[1][2]} Minuten, laufe zum Grabmal.`)
```