Josephus-Problem

Das **Josephus-Problem** oder die **Josephus-Permutation** ist ein theoretisches Problem aus der Informatik oder Mathematik.

Es werden n nummerierte Objekte im Kreis angeordnet; dann wird, beginnend mit der Nummer s , jedes s -te Objekt entfernt, wobei der Kreis immer wieder geschlossen wird. Die Reihenfolge der entfernten Objekte wird als Josephus-Permutation bezeichnet.

Ziel dieses Problems ist es, bei gegebenem n und s , das letzte Objekt der Permutation zu bestimmen.

1 Geschichte

Das Problem wurde nach dem jüdischen Historiker Flavius Josephus benannt, welcher sich 67 n. Chr. beim Kampf um die galiläische Stadt Jotapata mit 40 weiteren Männern in einer Höhle vor den Römern versteckt hielt. Als das Versteck verraten wurde, sicherten die Römer Josephus freies Geleit zu für den Fall, dass er das Versteck verließ. Seine Gefolgsleute drohten allerdings ihn umzubringen und wollten lieber sterben, als den Römern in die Hände zu fallen. Daraufhin machte Josephus den Vorschlag eines kollektiven Suizids, in dem sich alle im Kreis aufstellen und jeder seinen linken Nachbarn töten sollte. Er stellte sich an die 19. Stelle, blieb damit als Letzter übrig und ergab sich den Römern und zu überleben.

2 Lösung

Wir lösen das Problem für den Fall, dass jedes 2. Element entfernt wird ($k = 2$). Für den allgemeinen Fall $k \neq 2$ folgt eine Lösung unten. Wir leiten die Lösung mittels Rekursion her. Bezeichnen wir mit $f(n)$ das letzte Element der Permutation, wenn anfangs n Elemente vorhanden waren (mit $k = 2$). Im ersten Durchlauf werden alle geradzahligen Elemente entfernt. In der zweiten Runde werden die Elemente entfernt, die an der neuen 2., 4. usw. Position stehen, so als hätte es keine erste Runde gegeben. Wenn die Anfangsanzahl von Elementen gerade ist, dann war das Element x aus der zweiten Runde in der ersten Runde an Position $2x-1$. Das Element $f(2n)$ war ursprünglich auf Position $2f(n)-1$. Das ergibt folgende Rekursionsformel:

$$f(2n) = 2f(n) - 1.$$

Wenn die Elementanzahl anfangs ungerade ist, dann wird in der zweiten Runde das ursprünglich erste Element entfernt, aber auch hier wird dann das neue 2., 4. usw. Element entfernt. In diesem Fall ergibt sich, dass Element x in der ersten Runde an Position $2x + 1$ war. Daraus folgt die Rekursionsformel:

$$f(2n + 1) = 2f(n) + 1.$$

Wenn wir die Werte für n und $f(n)$ tabellarisch darstellen, so sehen wir ein Muster:

Diese Tabelle lässt vermuten, dass $f(n)$ eine aufsteigende Folge ungerader Zahlen ist, welche wieder mit $f(n) = 1$ startet, wenn der Index n eine Zweierpotenz ist. Wenn wir m und l so wählen, dass $n = 2^m + l$ und $0 \leq l < 2^m$, dann folgt $f(n) = 2 \cdot l + 1$. Es ist offensichtlich, dass die Werte aus der Tabelle diese Gleichung erfüllen. Nachfolgend geben wir den Beweis durch vollständige Induktion an.

Behauptung: Wenn $n = 2^m + l$ und $0 \leq l < 2^m$, dann folgt $f(n) = 2l + 1$.

Beweis: Wir verwenden die vollständige Induktion über n . Der Fall $n = 1$ ist wahr. Wir betrachten die Fälle für gerades n und ungerades n getrennt.

Wenn n gerade, dann wähle man l_1 und m_1 so, dass $n/2 = 2^{m_1} + l_1$ und $0 \leq l_1 < 2^{m_1}$. Es gilt $l_1 = l/2$. Wir haben $f(n) = 2f(n/2) - 1 = 2((2l_1) + 1) - 1 = 2l + 1$, bei der die zweite Gleichung aus der Induktionsannahme folgt.

Wenn n ungerade, dann wähle man l_1 und m_1 so, dass $(n - 1)/2 = 2^{m_1} + l_1$ and $0 \leq l_1 < 2^{m_1}$. Es gilt $l_1 = (l-1)/2$. Wir haben $f(n) = 2f((n-1)/2)+1 = 2((2l_1) + 1) + 1 = 2l + 1$, bei der die zweite Gleichung ebenfalls aus der Induktionsannahme folgt. Damit ist die Behauptung bewiesen.

Die eleganteste Form der Lösung folgt aus der binären Repräsentation von n : $f(n)$ kann durch eine 1-Bit-Linksrotation von n selbst ermittelt werden. Wenn wir n binär als $n = b_0 b_1 b_2 b_3 \ldots b_m$ repräsentieren, dann ist die Lösung gegeben als $f(n) = b_1 b_2 b_3 \ldots b_m b_0$. Der Beweis folgt aus der Repräsentation von n als $2^m + l$.

Die dynamische Programmierung ist der einfachste Weg dieses Problem für den allgemeinen Fall zu lösen.

Diese Methode verwendet die Rekursionsformel:

$$f(n, k) = (f(n - 1, k) + k) \mathrm{mod} n \text{ , mit } f(1, k) = 0$$

welche offensichtlich ist, wenn man berücksichtigt, wie sich die Nummer des letzten Elements ändert wenn man von $n - 1$ nach n wechselt. Diese Methode hat eine Laufzeit von $O(n)$, aber für kleine k und große n gibt es einen anderen Ansatz. Dieser zweite Ansatz verwendet auch die dynamische Programmierung, erfordert aber nur eine Laufzeit von $O(k \log n)$. Er entfernt die *k*., 2*k*., ..., $\lfloor n/k \rfloor$. Elemente in einem Schritt und ändert dann die Nummerierung.

3 Implementierung

Der folgende Algorithmus realisiert das Problem rekursiv nach der obigen Rekursionsformel für den Fall *k=2* und besitzt eine Laufzeit von *O(log(n))*.

int **josephus**(int n) { **if**(n == 1) return 1; **if**((n%2) == 0) return 2 * josephus(n / 2) - 1; **if**((n%2) == 1) return 2 * josephus((n - 1) / 2) + 1; }

Gemäß der geschlossenen Formel *f(n)=2*l + 1* lässt sich der folgende nicht-rekursive Algorithmus angeben. Seine Laufzeit liegt in *O(1)*.

int **josephus**(int n) { m = floor(log(n)); l = n - 2^m; j = 2*l + 1; return j; }

4 Literatur

- Paul Yiu: *Recreational Mathematics*, Florida Atlantic University: Department of Mathematics *(Als PDF in englischer Sprache verfügbar; 868 kB)*
- Walter William Rouse Ball, Harold Scott Macdonald Coxeter: *Mathematical Recreations and Essays*, Dover, 1987. Seiten 32-36, ISBN 0-486-25357-0
- Ronald L. Graham, Donald E. Knuth und Oren Patashnik: *Concrete Mathematics: A Foundation for Computer Science*, Massachusetts, 1994. Seiten 8-16, ISBN 978-0-201-55802-9

5 Text- und Bildquellen, Autoren und Lizenzen

5.1 Text

- **Josephus-Problem** *Quelle:* https://de.wikipedia.org/wiki/Josephus-Problem?oldid=159676327 *Autoren:* Kku, Stefan Birkner, Jpp, P. Birken, AHZ, RedBot, GMH, Matthias Kupfer, Vrumfondel, TXiKiBoT, Erdbeerquetscher, Loveless, Achsenzeit, Poisend-Ivy, BOTarate, Inkowik, DumZiBoT, LaaknorBot, Luckas-bot, GrouchoBot, Xqbot, ArthurBot, E anthes, Dymit, Quartl, MorbZ-Bot, Serols, Baird's Tapir, EmausBot, Gartenkralle deluxe, KLBot2 und Anonyme: 9

5.2 Bilder

5.3 Inhaltslizenz

- Creative Commons Attribution-Share Alike 3.0