Besondere Lernleistung

Bundeswettbewerb Informatik 2. Runde

Lucas Schwebler

10. November 2021

Aufgabe 3 - Eisbudendilemma

Aufgabenstellung

- See mit Umfang u und n Häusern
- 3 Eisbuden
- Alle Positionen $\in \mathbb{Z}/u\mathbb{Z}$
- Mehrheitsabstimmung für neue Verteilung der Eisbuden; jedes Haus stimmt dafür, falls sich Abstand zur nächsten Eisbude verkleinert
- Aufgabe: Finde ein **stabile** Verteilung, d.h. es gibt keinen neuen Vorschlag mit Mehrheit

Grundidee

- Bruteforce: Probiere alle $\binom{u}{3} = \mathcal{O}(u^3)$ Eisbudenverteilungen
- Stabilitätstest: Probiere alle $\mathcal{O}(u^3)$ Vorschläge.

Grundidee

- Bruteforce: Probiere alle $\binom{u}{3} = \mathcal{O}(u^3)$ Eisbudenverteilungen
- ~~Stabilitätstest: Probiere alle $\mathcal{O}(u^3)$ Vorschläge.~~

$\Longrightarrow$ Laufzeit $\mathcal{O}(u^6)$ zu langsam!

- Umwandlung in Optimierungsproblem:
  Finde den Vorschlag mit den meisten Stimmen.
- Stimmen $> \frac{n}{2} \implies$ instabil
- Stimmen $\leq \frac{n}{2} \implies$ stabil

Aufgabe 3

Lösungsstrategie

- 3 neue Eisbuden auf 3 Intervalle zwischen alten Eisbuden verteilt.
- Zwei neue Eisbuden zwischen zwei alten:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
H		H	H					H				H	
E	**E**											**E**	E

$\Longrightarrow$ Es können alle Stimmen geholt werden

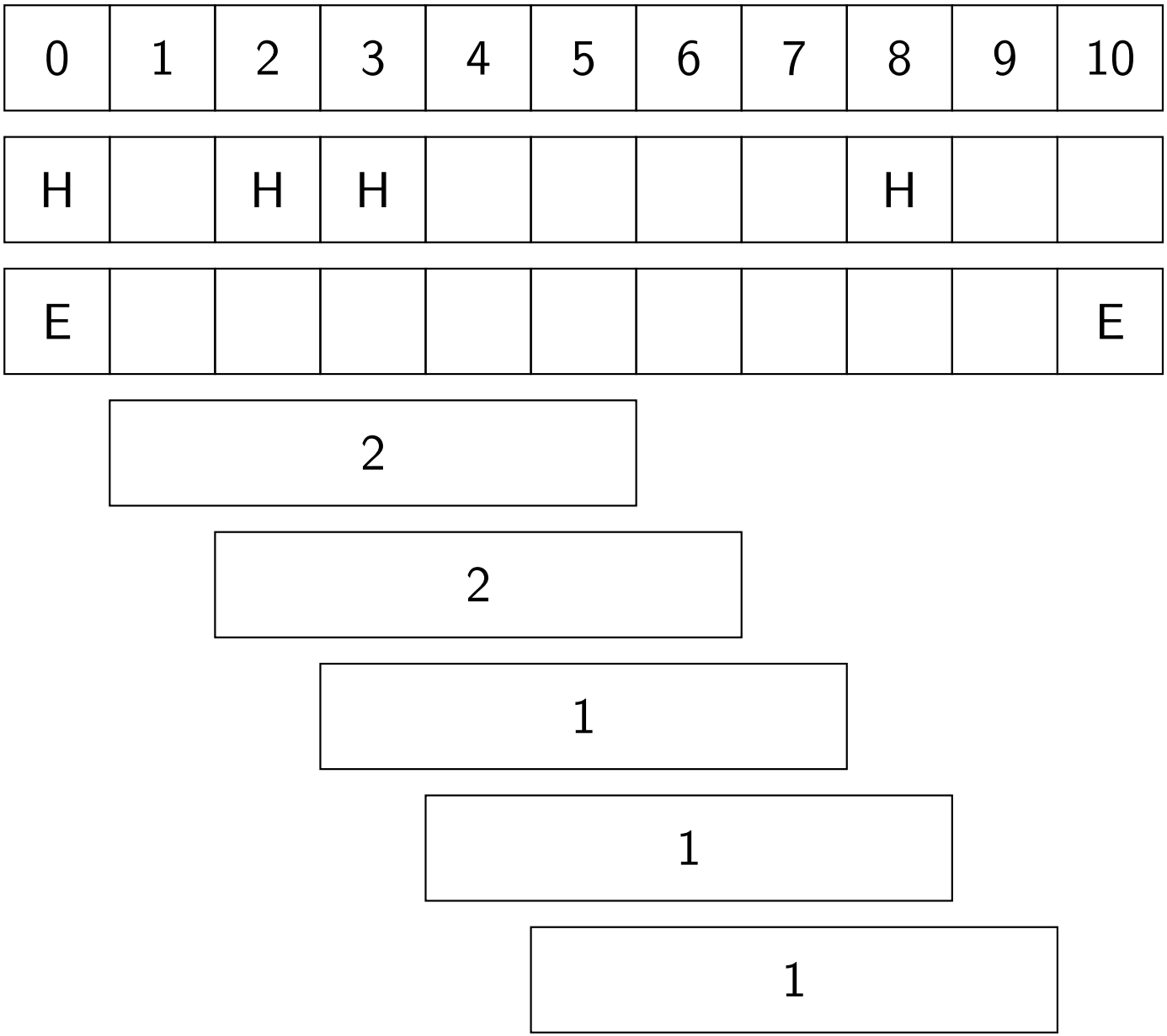
- Mehr als 2 Eisbuden nicht zielführend

Aufgabe 3

Lösungsstrategie

- Eine Eisbude zwischen zwei alten:
- Alle Häuser in einem Teilintervall der Größe $g = \left\lfloor \frac{m}{2} \right\rfloor - 1$ stimmen dafür, wobei m = Intervallgröße (Herleitung: siehe Einsendung)
- Gesucht: Teilintervall der Größe g mit maximaler Häuseranzahl

$\Longrightarrow$ wird vorberechnet


0 1 2 3 4 5 6 7 8 9 10
H H H H
E E
2
2
1
1
1

Aufgabe 3

Vorberechnung

- Präfixsummen:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
H		H	H					H				H	
1	1	2	3	3	3	3	3	4	4	4	4	5	5

$\Longrightarrow$ Anzahl an Häusern in Intervall in $\mathcal{O}(1)$ bestimmen

- z.B. von 2 bis 9: $P_9 - P_1 = 4 - 1 = 3$
- größte Teilintervalle in $\mathcal{O}(u^3)$ berechenbar

Aufgabe 3

maximale Stimmenzahl

- 7 Fälle:
  - Zwischen zwei alten wird je eine neue Eisbude platziert
  - Zwei Eisbuden in einem Zwischenraum, die dritte in einem anderen (6 Möglichkeiten)
- Stabilität nach $\mathcal{O}(u^3)$ Vorberechnung in $\mathcal{O}(1)$ entscheidbar!
- Gesamtlaufzeit $\mathcal{O}(u^3)$

Aufgabe 3

maximale Stimmenzahl

- 7 Fälle:
  - Zwischen zwei alten wird je eine neue Eisbude platziert
  - Zwei Eisbuden in einem Zwischenraum, die dritte in einem anderen (6 Möglichkeiten)
- Stabilität nach $\mathcal{O}(u^3)$ Vorberechnung in $\mathcal{O}(1)$ entscheidbar!
- Gesamtlaufzeit $\mathcal{O}(u^3)$
- Kann auf $\mathcal{O}(u^2)$ oder $\mathcal{O}(n)$ reduziert werden (siehe Einsendung)
  - mindestens eine Eisbude steht bei einem Haus
  - “Rückwärts arbeiten“
  - Monotonietrick

Code - Bruteforce in der Mainfunktion

```
1  int main(){
2      ...
3      // Iteriere über alle Eisbudenverteilungen
4      for(int i = 0; i < u; ++i){
5          for(int j = i+1; j < u; ++j){
6              for(int k = j+1; k < u; ++k){
7                  // Berechne die maxiamle Anzahl an Stimmen eine
8                  int best = maxVotes(i, j, k);
9                  // Stabile Verteilung gefunden, falls keine Me
10                 if(best * 2 <= n)
11                 {
12                     cout << "Stabile Verteilung gefunden!\n";
13                     cout << i << " " << j << " " << k << "\n";
14                     exit(0);
15                 }
16             }
17         }
18     }
19     cout << "Es wurde keine stabile Verteilung gefunden!\n";
20 }
```

Code - maximale Stimmenzahl berechnen

```
1  int maxVotes(int i, int j, int k){
2      // Anzahl an Stimmen, die mit 2 Eisbuden geholt werden kön
3      int s[3];
4      s[0] = getVotes(add(i, 1), sub(j, 1));
5      s[1] = getVotes(add(j, 1), sub(k, 1));
6      s[2] = getVotes(add(k, 1), sub(i, 1));
7      // Anzahl an Stimmen, die mit einer Eisbude geholt werden
8      int m[3];
9      m[0] = maxInterval[i][j];
10     m[1] = maxInterval[j][k];
11     m[2] = maxInterval[k][i];
12     // Teste alle Möglichkeiten, die 3 Eisbuden auf die Interva
13     int best = m[0] + m[1] + m[2]; // Je eine Eisbude pro Inte
14     for(int g = 0; g < 3; ++g){
15         for(int h = 0; h < 3; ++h){
16             // Zwei Eisbuden in einem Intervall und eine in ei
17             if(g != h) best = max(best, s[g] + m[h]);
18         }
19     }
20     return best;
21 }
```

Code - Präfixsummen

```
1  // Bestimmt in O(1) die Anzahl an Häusern / Stimmen zwischen
       den Häusern bei l und r
2  // Die Grenzen sind einschließlich
3  int getVotes(int l, int r){
4      if(r < l) return getVotes(0, r) + getVotes(l, u-1);
5      if(l > 0) return vpref[r] - vpref[l-1];
6      return vpref[r];
7  }
8
9  int main(){
10     ...
11     // Vorberechnung Präfixsumme
12     vpref[0] = votes[0];
13     for(int i = 1; i < u; ++i){
14         vpref[i] = vpref[i-1] + votes[i];
15     }
16     ...
17 }
```