

# Lausn á *Reiknirit*

Bergur Snorrason

30. janúar 2023

- ▶ Skoðum aftur skiladæmið *Reiknirit*.

- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .

- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .
- ▶ Síðan koma  $n$  heiltölur  $a_1, a_2, \dots, a_n$ .

- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .
- ▶ Síðan koma  $n$  heiltölur  $a_1, a_2, \dots, a_n$ .
- ▶ Gerum ráð fyrir að við séum með forrit sem gerir eftirfarandi:

- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .
- ▶ Síðan koma  $n$  heiltölur  $a_1, a_2, \dots, a_n$ .
- ▶ Gerum ráð fyrir að við séum með forrit sem gerir eftirfarandi:
  - ▶ Prentar tölurnar.

- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .
- ▶ Síðan koma  $n$  heiltölur  $a_1, a_2, \dots, a_n$ .
- ▶ Gerum ráð fyrir að við séum með forrit sem gerir eftirfarandi:
  - ▶ Prentar tölurnar.
  - ▶ Fjarlægir öll eintök af algengustu tölunni í listan.

- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .
- ▶ Síðan koma  $n$  heiltölur  $a_1, a_2, \dots, a_n$ .
- ▶ Gerum ráð fyrir að við séum með forrit sem gerir eftirfarandi:
  - ▶ Prentar tölurnar.
  - ▶ Fjarlægir öll eintök af algengustu tölunni í listan.
  - ▶ Endurtekur skrefin að ofan þar til listinn er tómur.



- ▶ Skoðum aftur skiladæmið *Reiknirit*.
- ▶ Fyrsta lína inntaksins inniheldur heiltölu  $n$ .
- ▶ Síðan koma  $n$  heiltölur  $a_1, a_2, \dots, a_n$ .
- ▶ Gerum ráð fyrir að við séum með forrit sem gerir eftirfarandi:
  - ▶ Prentar tölurnar.
  - ▶ Fjarlægir öll eintök af algengustu tölunni í listan.
  - ▶ Endurtekur skrefin að ofan þar til listinn er tómur.
- ▶ Dæmið snýst um að finna hversu margar tölur eru prentaðar í heildina.

- ▶ Síðast leystum við þetta dæmi með því að útfæra forritið sem er lýst í dæminu.

- ▶ Síðast leystum við þetta dæmi með því að útfæra forritið sem er lýst í dæminu.
- ▶ Við komumst þó að því að sú lausn var  $\mathcal{O}(n^2)$  sem reyndist of hæg.

- ▶ Síðast leystum við þetta dæmi með því að útfæra forritið sem er lýst í dæminu.
- ▶ Við komumst þó að því að sú lausn var  $\mathcal{O}(n^2)$  sem reyndist of hæg.
- ▶ Tökum eftir að við getum notað svipaða hugmynd og í hægu útfærslunni til að telja hversu oft hver tala kemur fyrir.

- ▶ Síðast leystum við þetta dæmi með því að útfæra forritið sem er lýst í dæminu.
- ▶ Við komumst þó að því að sú lausn var  $\mathcal{O}(n^2)$  sem reyndist of hæg.
- ▶ Tökum eftir að við getum notað svipaða hugmynd og í hægu útfærslunni til að telja hversu oft hver tala kemur fyrir.
- ▶ Tökum einnig eftir að hvert eintak af algengustu tölunni er prentað einu sinni, hvert eintak af næst algengustu tölunni er prentað tvisvar og svo framvegis.

- ▶ Síðast leystum við þetta dæmi með því að útfæra forritið sem er lýst í dæminu.
- ▶ Við komumst þó að því að sú lausn var  $\mathcal{O}(n^2)$  sem reyndist of hæg.
- ▶ Tökum eftir að við getum notað svipaða hugmynd og í hægu útfærslunni til að telja hversu oft hver tala kemur fyrir.
- ▶ Tökum einnig eftir að hvert eintak af algengustu tölunni er prentað einu sinni, hvert eintak af næst algengustu tölunni er prentað tvisvar og svo framvegis.
- ▶ Einnig skiptir talan sjálf ekki máli, heldur eingöngu hversu oft hún kemur fyrir.

- ▶ Látum því  $h_1 \geq h_2 \geq \dots \geq h_k$  þannig að algengast talan kemur  $h_1$  sinni fyrir, næst algengasta talan kemur  $h_2$  sinnum fyrir og svo framvegis.

- ▶ Látum því  $h_1 \geq h_2 \geq \dots \geq h_k$  þannig að algengast talan kemur  $h_1$  sinni fyrir, næst algengasta talan kemur  $h_2$  sinnum fyrir og svo framvegis.
- ▶ Svarið er þá

$$\sum_{i=1}^k i \cdot h_i.$$



- ▶ Látum því  $h_1 \geq h_2 \geq \dots \geq h_k$  þannig að algengast talan kemur  $h_1$  sinni fyrir, næst algengasta talan kemur  $h_2$  sinnum fyrir og svo framvegis.
- ▶ Svarið er þá

$$\sum_{i=1}^k i \cdot h_i.$$

- ▶ Við þurfum þó að passa okkur aðeins.

- ▶ Látum því  $h_1 \geq h_2 \geq \dots \geq h_k$  þannig að algengast talan kemur  $h_1$  sinni fyrir, næst algengasta talan kemur  $h_2$  sinnum fyrir og svo framvegis.
- ▶ Svarið er þá

$$\sum_{i=1}^k i \cdot h_i.$$

- ▶ Við þurfum þó að passa okkur aðeins.
- ▶ Ef  $h_1 = h_2 = \dots h_k = 1$  (þá er einnig  $k = n$ ) fæst að svarið er

$$\sum_{i=1}^k i \cdot h_i = \sum_{i=1}^n i = \frac{n \cdot (n + 1)}{2}.$$

- ▶ Látum því  $h_1 \geq h_2 \geq \dots \geq h_k$  þannig að algengast talan kemur  $h_1$  sinni fyrir, næst algengasta talan kemur  $h_2$  sinnum fyrir og svo framvegis.
- ▶ Svarið er þá

$$\sum_{i=1}^k i \cdot h_i.$$

- ▶ Við þurfum þó að passa okkur aðeins.
- ▶ Ef  $h_1 = h_2 = \dots h_k = 1$  (þá er einnig  $k = n$ ) fæst að svarið er

$$\sum_{i=1}^k i \cdot h_i = \sum_{i=1}^n i = \frac{n \cdot (n + 1)}{2}.$$

- ▶ Svo, þar sem  $n$  getur verið allt að  $10^6$ , getur svarið okkar orðið of stórt fyrir `int`.

- ▶ Látum því  $h_1 \geq h_2 \geq \dots \geq h_k$  þannig að algengast talan kemur  $h_1$  sinni fyrir, næst algengasta talan kemur  $h_2$  sinnum fyrir og svo framvegis.
- ▶ Svarið er þá

$$\sum_{i=1}^k i \cdot h_i.$$

- ▶ Við þurfum þó að passa okkur aðeins.
- ▶ Ef  $h_1 = h_2 = \dots h_k = 1$  (þá er einnig  $k = n$ ) fæst að svarið er

$$\sum_{i=1}^k i \cdot h_i = \sum_{i=1}^n i = \frac{n \cdot (n + 1)}{2}.$$

- ▶ Svo, þar sem  $n$  getur verið allt að  $10^6$ , getur svarið okkar orðið of stórt fyrir `int`.
- ▶ Við þurfum því að nota `long long`.

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 typedef long long ll;
4
5 #define CMP(E, F) (F <= E) - (E <= F)
6 int cmp(const void* p1, const void* p2)
7 {
8     return CMP(*(ll*)p2, *(ll*)p1);
9 }
10
11 int main()
12 {
13     ll i, j, k, r = 0, n;
14     scanf("%lld", &n);
15     ll a[n], b[n];
16     for (i = 0; i < n; i++) scanf("%lld", &a[i]);
17     qsort(a, n, sizeof *a, cmp);
18     i = k = 0;
19     while (i < n)
20     {
21         j = i;
22         while (j < n && a[i] == a[j]) j++;
23         b[k++] = j - i;
24         i = j;
25     }
26     qsort(b, k, sizeof *b, cmp);
27     for (i = 0; i < k; i++) r += (i + 1)*b[i];
28     printf("%lld\n", r);
29     return 0;
30 }

```

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 typedef long long ll;
4
5 #define CMP(E, F) (F <= E) - (E <= F)
6 int cmp(const void* p1, const void* p2)
7 {
8     return CMP(*(ll*)p2, *(ll*)p1);
9 }
10
11 int main()
12 {
13     ll i, j, k, r = 0, n;
14     scanf("%lld", &n);
15     ll a[n], b[n];
16     for (i = 0; i < n; i++) scanf("%lld", &a[i]);
17     qsort(a, n, sizeof *a, cmp);
18
19     for (i = j = k = 0; i < n; k++, i = j)
20         for (b[k] = 0; j < n && a[i] == a[j]; j++) b[k]++;
21
22
23
24
25
26     qsort(b, k, sizeof *b, cmp);
27     for (i = 0; i < k; i++) r += (i + 1)*b[i];
28     printf("%lld\n", r);
29     return 0;
30 }

```

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(\quad)$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .



- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}( )$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .
- ▶ Síðan röðum við aftur.

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .
- ▶ Síðan röðum við aftur.
- ▶ Að lokum reiknum við summuna í  $\mathcal{O}(n)$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .
- ▶ Síðan röðum við aftur.
- ▶ Að lokum reiknum við summuna í  $\mathcal{O}(n)$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .
- ▶ Síðan röðum við aftur.
- ▶ Að lokum reiknum við summuna í  $\mathcal{O}(n)$ .
- ▶ Tímaflækjan er því í heildina  $\mathcal{O}(\quad)$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .
- ▶ Síðan röðum við aftur.
- ▶ Að lokum reiknum við summuna í  $\mathcal{O}(n)$ .
- ▶ Tímaflækjan er því í heildina  $\mathcal{O}(n \log n)$ .

- ▶ Sjáum við byrjum á að raða, sem er  $\mathcal{O}(n \log n)$ .
- ▶ Við teljum síðan hvað hver tala kemur oft fyrir, sem er  $\mathcal{O}(n)$ .
- ▶ Síðan röðum við aftur.
- ▶ Að lokum reiknum við summuna í  $\mathcal{O}(n)$ .
- ▶ Tímaflækjan er því í heildina  $\mathcal{O}(n \log n)$ .
- ▶ Nú er  $10^{-8} \cdot 10^6 \cdot \log 10^6 \sim 0,2$ , svo þessi lausn er nógu hröð.



