Lausn á *Reiknirit*

Bergur Snorrason

24. 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 $a_1, a_2, \ldots, a_n$.

- Skoðum aftur skiladæmið *Reiknirit*.
- Fyrsta lína inntaksins inniheldur heiltölu n.
- Síðan koma n heiltölur $a_1, a_2, \ldots, a_n$.
- 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 $a_1, a_2, \ldots, a_n$.
- 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 $a_1, a_2, \ldots, a_n$.
- 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 $a_1, a_2, \ldots, a_n$.
- 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 $a_1, a_2, \ldots, a_n$.
- 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 $h_1 \geq h_2 \geq \cdots \geq h_k$ þannig að algengast talan kemur h_1 sinni fyrir, næst algengasta talan kemur h_2 sinnum fyrir og svo framvegis.

- Látum því $h_1 \geq h_2 \geq \cdots \geq h_k$ 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 $h_1 \geq h_2 \geq \cdots \geq h_k$ 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 $h_1 \geq h_2 \geq \cdots \geq h_k$ 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 $h_1 = h_2 = \ldots h_k = 1$ (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 $h_1 \geq h_2 \geq \cdots \geq h_k$ 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 $h_1 = h_2 = \ldots h_k = 1$ (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 $h_1 \geq h_2 \geq \cdots \geq h_k$ 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 $h_1 = h_2 = \ldots h_k = 1$ (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`.

```c
#include <stdlib.h>
#include <stdio.h>
typedef long long ll;

#define CMP(E, F) (F <= E) - (E <= F)
int cmp(const void* p1, const void* p2)
{
    return CMP(*(ll*)p2, *(ll*)p1);
}

int main()
{
    ll i, j, k, r = 0, n;
    scanf("%lld", &n);
    ll a[n], b[n];
    for (i = 0; i < n; i++) scanf("%lld", &a[i]);
    qsort(a, n, sizeof *a, cmp);
    i = k = 0;
    while (i < n)
    {
        j = i;
        while (j < n && a[i] == a[j]) j++;
        b[k++] = j - i;
        i = j;
    }
    qsort(b, k, sizeof *b, cmp);
    for (i = 0; i < k; i++) r += (i + 1)*b[i];
    printf("%lld\n", r);
    return 0;
}
```

```c
#include <stdlib.h>
#include <stdio.h>
typedef long long ll;

#define CMP(E, F) (F <= E) - (E <= F)
int cmp(const void* p1, const void* p2)
{
    return CMP(*(ll*)p2, *(ll*)p1);
}

int main()
{
    ll i, j, k, r = 0, n;
    scanf("%lld", &n);
    ll a[n], b[n];
    for (i = 0; i < n; i++) scanf("%lld", &a[i]);
    qsort(a, n, sizeof *a, cmp);

    for (i = j = k = 0; i < n; k++, i = j)
        for (b[k] = 0; j < n && a[i] == a[j]; j++) b[k]++;





    qsort(b, k, sizeof *b, cmp);
    for (i = 0; i < k; i++) r += (i + 1)*b[i];
    printf("%lld\n", r);
    return 0;
}
```

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

- 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 $\mathcal{O}(\quad)$.

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

- 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 $\mathcal{O}(\quad)$.

- 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 $\mathcal{O}(n)$.

- 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 $\mathcal{O}(\qquad\quad)$.

- 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.