Sortowanie przez zliczanie

$x_i \in \{1..k\}$

$c[i]$ – $\#i$

- ale żeby było stabilne!

↑ [pary klucz-wartość]

→ zliczać ile jest mniejszych kluczy i wstawiać kolejne elementy od lewej na prawidłowe miejsce

$w[i]$ – ile już wstawiono el. o kluczu i

r – wynik

```
foreach x in xs:
    r[(Σ_{j<x} c[j]) + w[x]] = x
    w[x]++
end
```

Sortowanie kubełkowe

$x_i \in [0,1)$ $\quad i \in 1..n$

---

n kubełków $\quad x \longrightarrow$ kubełek $\lfloor x \cdot n \rfloor$

Każdy kubełek sortujemy z osobna czymś prostym (np. insert sort)

Pesymistycznie: $O(n^2)$

Oczekiwana: $O(n)$

Niech x_i – liczba liczb w i-tym kubełku

$E(x_i) = 1 = n \cdot \frac{1}{n}$

$E(\sum x_i^2) = \sum E(x_i^2) = ? \ldots = O(n)$

$V(x)$?

$E(x^2) \neq (E(x))^2$

Sortowanie leksykograficzne

Dane: $x_i \in \Sigma^*$ $\quad |\Sigma| = k$

1) sortować od pierwszej pozycji, litera ~ kubełek, rekurencyjnie kubełki

2) sortować od ostatniej pozycji

I) $x_i \in \Sigma^d$ (równej długości)

$O(d(n+k))$

II) różnej długości, $|x_i| = d_i$

d_{max}

- posortować ciągi o długości $\geq i+1$
- dołączyć na początek ciągi dł. i
- posortować po i-tej pozycji (stabilnie)

$\sum d_i + k \cdot d_{max}$

↑ nie pasi

- utwórz listy niepuste[1]
  - $r \in$ niepuste[1] gdy r jest na 1-tej pozycji w jakimś ciągu
- niepuste[i] – nienależące

$O(\sum d_i)$ · utwórz listy ciągi[d] $(\in [1..d_{max}]$

· ciągi[d] $= \{x_i \mid |x_i| = d\}$

→ · tworzymy pary (pozycja, litera) ze wszystkich słów [$\# = \sum d_i$]

· pary leksykograficznie $O(d_{max} + k \overset{?}{+} \sum d_i) = O(k + \sum d_i)$

? ? ?

0 0 0