

Föreläsning 16 i ADK

Algoritmkonstruktion: sortering i linjär tid

Stefan Nilsson

KTH

```
function COUNTINGSORT( $V[1..n]$ ,  $f : \text{element} \rightarrow [1..k]$ )  
   $C[1..k]$ : Hjälpparray för räkning  
   $\text{res}[1..n]$ : Hjälpparray för lagring av resultatet  
  for  $i \leftarrow 1$  to  $k$  do  $C[i] \leftarrow 0$   
  for  $j \leftarrow 1$  to  $n$  do  $C[f(V[j])]++$   
   $\text{sum} \leftarrow 0$   
  for  $i \leftarrow k$  downto  $1$  do  
     $\text{sum} \leftarrow \text{sum} + C[i]$   
     $C[i] \leftarrow n - \text{sum} + 1$   
  for  $j \leftarrow 1$  to  $n$  do  
     $\text{res}[C[f(v[j])]] \leftarrow v[j]$   
     $C[f(v[j])]++$   
  return  $\text{res}$ 
```

Tidskomplexitet: $\Theta(k + n)$

Exempel

- $V[1..10] = [17, 2, 2, 4711, 2, 17, 16, 4711, 2, 17]$
- $f(2) = 1, f(17) = 2, f(4711) = 3$
- $C[1..3]$ efter räkneslingan $= [4, 4, 2]$
- $C[1..3]$ efter indexomräkningen $[1, 5, 9]$
- $res[1..10] = [2, 2, 2, 2 \mid 17, 17, 17, 17 \mid 4711, 4711]$

Några fall då man kan sortera i $o(n \log n)$

- 1 Bara ett konstant antal olika element ska sorteras:
 - $\Theta(n)$ med räknasortering (counting sort)
- 2 Elementen som ska sorteras är tal som är jämnt fördelade i ett visst intervall:
 - $\Theta(n)$ med bucket sort
- 3 Elementen som ska sorteras är strängar som består av d "siffror"
($v[i] = s_{i,1}s_{i,2} \dots s_{i,d}$)
 - $\Theta(nd)$ med radixsortering
 - **for** $i \leftarrow d$ **downto** 1 **do**
 - | Sortera $v[1..n]$ efter siffra i med en stabil sorteringsalgoritm
 - Om d är konstant får vi linjär tidskomplexitet
 - Om vi räknar antalet siffror i indata få vi linjär tidskomplexitet $\Theta(N)$ där $N = nd$

Radixsorteringsexempel med $d = 3$

Osorterat:	Pass 1:	Pass 2:	Pass 3:
480	480	902	009
973	381	905	381
902		009	
905	902		419
532	532	816	480
652	652	419	
783		532	532
009	973		652
653	783	652	653
419	653	653	
816	905	973	783
381	816	480	816
	009	381	902
	419	783	905
			973