# TDDI16 – Föreläsning 9

## Sortering i linjär tid, ”intressanta” algoritmer och tillämpningar

Filip Strömbäck

LINKÖPING UNIVERSITY

# Planering

| Vecka | Fö | Lab |
|---|---|---|
| 36 | Komplexitet, Linjära strukturer | ---- |
| 37 | Träd, AVL-träd | 1--- |
| 38 | Hashning, meet-in-the-middle | 12-- |
| 39 | Grafer, kortaste vägen | -23- |
| 40 | Grafer, andra grafalgoritmer | -23- |
| 41 | Sortering | --34 |
| 42 | Repetition | ---4 |

LINKÖPING UNIVERSITY

# Sortering i $\mathcal{O}(n)$?

Från förra föreläsningen:

- Jämförelsebaserade sorteringar begränsas av $\Omega(n \log n)$

Hur sorterar vi då i $\mathcal{O}(n)$?

# Sortering i $\mathcal{O}(n)$?

Från förra föreläsningen:

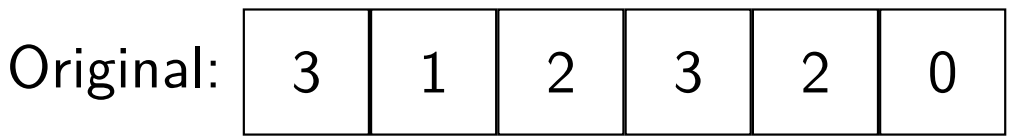
- Jämförelsebaserade sorteringar begränsas av $\Omega(n \log n)$

Hur sorterar vi då i $\mathcal{O}(n)$?

Vi skippar jämförelserna!

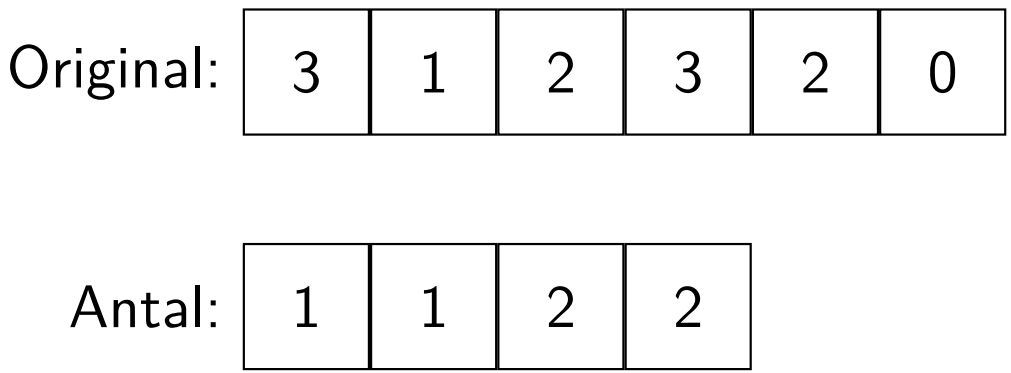
# Count sort – Idé

1. Skapa en array av heltal med ett index för varje möjligt tal i indatat
2. Räkna förekomster av alla heltal i indatan, lagra dem i arrayet
3. Skriv tillbaka talen i ordning utifrån antalet



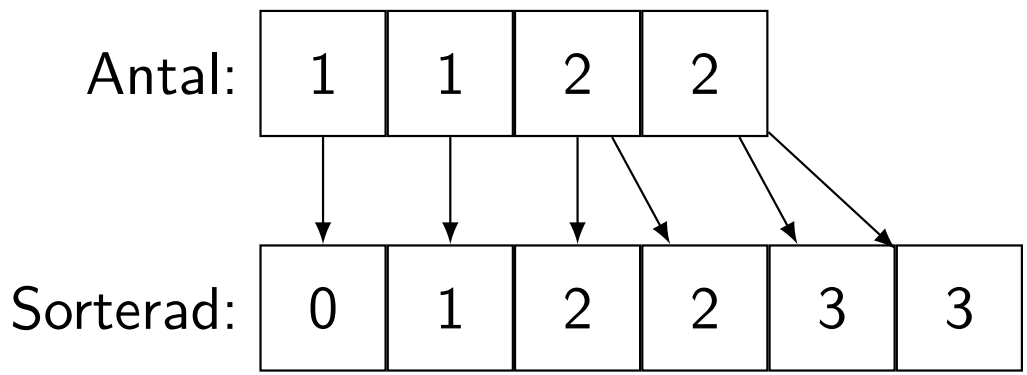
# Count sort – Idé

1. Skapa en array av heltal med ett index för varje möjligt tal i indatat
2. Räkna förekomster av alla heltal i indatan, lagra dem i arrayet
3. Skriv tillbaka talen i ordning utifrån antalet



# Count sort – Idé

1. Skapa en array av heltal med ett index för varje möjligt tal i indatat
2. Räkna förekomster av alla heltal i indatan, lagra dem i arrayet
3. Skriv tillbaka talen i ordning utifrån antalet



# Count sort – Implementation

```
void count_sort(vector<int> &data) {
  vector<int> count(max_number, 0);
  for (int x : data)
    count[x]++;

  size_t pos = 0;
  for (int i = 0; i < int(max_number); i++)
    for (int c = 0; c < count[i]; c++)
      data[pos++] = i;
}
```

# Bucket sort

En grov sortering som första steg.

- Lägg elementen i olika ”lådor” baserat på deras storlek
- Sortera sedan varje låda med någon annan sorteringsalgoritm

# Bucket sort – implementation

```
void bucket_sort(vector<int> &data) {
  vector<vector<int>> buckets(num_buckets);

  const int divide = (max_number / num_buckets);
  for (int x : data)
    buckets[x / divide].push_back(x);

  size_t pos = 0;
  for (vector<int> &x : buckets) {
    sort(x.begin(), x.end());
    copy(x.begin(), x.end(), data.begin() + pos);
    pos += x.size();
  }
}
```

# Radix sort (LSD first) – Idé

Om vi har stora datamängder blir det snabbt mycket minne som krävs...

**Idé:** Vi sorterar flera gånger!

- Sortera efter den minst signifikanta siffran
- Fortsätt sedan med med nästa siffra
- ...

Det är viktigt att sorteringen är **stabil**.

Bucket sort passar bra här!

# Radix sort (LSD first) – Implementation

```
void radix_sort(vector<int> &data) {
  for (int divide = 1;
       divide < 10 * max_number;
       divide *= 10) {

    bucket_sort_digit(data, divide);
  }
}
```

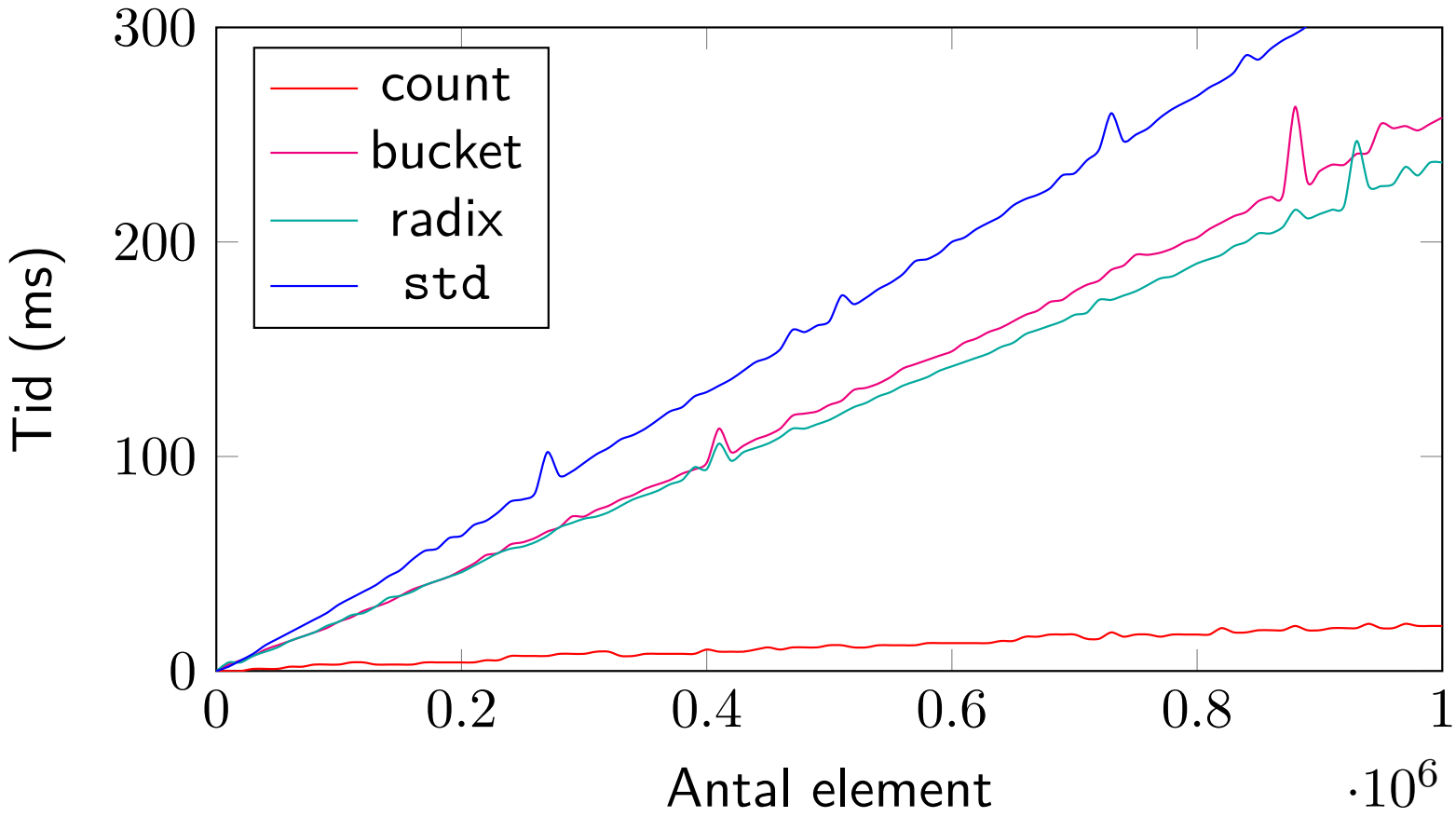
# Radix sort – modifierad bucket sort

```
void bucket_sort_digit(vector<int> &data, int div) {
  vector<vector<int>> buckets(10);

  for (int x : data)
    buckets[(x / div) % 10].push_back(x);

  size_t pos = 0;
  for (vector<int> &x : buckets) {
    copy(x.begin(), x.end(), data.begin() + pos);
    pos += x.size();
  }
}
```

# Radix sort (MSD first)

- Vi kan också börja med den mest signifikanta siffran!
- I så fall sorterar vi de olika ”partitionerna” separat i stället för samtidigt som i tidigare fallet

# Jämförelse – Element mellan 0 och 10000



# Sound of sorting

Det finns många visualiseringar av diverse sorteringsalgoritmer på internet, exempelvis:

- 15 algoritmer, en och en:
  `https://www.youtube.com/watch?v=kPRA0W1kECg`
- 6 olika algoritmer samtidigt:
  `https://www.youtube.com/watch?v=ZZuD6iUe3Pc`

LINKÖPING UNIVERSITY

# Vilken algoritm ska jag välja?

Jag vill sortera en kortlek. Vilken algoritm är bäst?

Jag vill sortera ett relativt litet antal rockar som hänger i galgar efter storlek. Vilken algoritm är bäst?

# Yatzy

Givet värdet på fem tärningar, skriv ut vilken poäng slaget ger på de olika kombinationerna i Yatzy. Det vill säga, se om det är någon eller några av:

| | | |
|---|---|---|
| ett par | två par | tretal |
| fyrtal | kåk | liten stege |
| stor stege | chans | yatzy |

Hur gör vi detta på ett smidigt sätt?

LINKÖPING UNIVERSITY

# Sleep sort

Starta en tråd/process för varje element, låt dem sova i motsvarande antal sekunder och sedan skriva ut sin data.

```
#!/bin/bash
task() { sleep "$1"; echo "$1"; }

for i in "$@"
do
    task $i &
done
wait
```

Risk för fel...

Vad är tidskomplexiteten?

# Machine learning sort

Idé:

1. Sortera en delmängd av datan
2. Träna ett neuralt nätverk att beräkna den slutgiltiga positionen baserat på ett element
3. Använd det för resten av datan

Återigen: Risk att datan inte blir korrekt sorterad...

`https://arxiv.org/abs/1805.04272`

# Bogosort

```
void bogosort(vector<int> &v) {
  while (!sorted(v))
    random_shuffle(v.begin(), v.end(), generator);
}

bool sorted(const vector<int> &v) {
  for (size_t i = 1; i < v.size(); i++)
    if (v[i - 1] > v[i])
      return false;

  return true;
}
```

# Bogobogosort

```
void bogobogosort(iter begin, iter end) {
  while (!sorted(begin, end))
    random_shuffle(begin, end, generator);
}

bool sorted(iter begin, iter end) {
  if (begin + 1 == end)
    return true;
  vector<int> x(begin, end);
  bogobogosort(begin, end - 1);
  return *(end - 2) <= *(end - 1);
}
```

http://www.dangermouse.net/esoteric/bogobogosort.html

# Worstsort

1. Generera alla permutationer av indatat, spara i en lista
2. Sortera listan lexiografiskt med hjälp av bubblesort
3. Ta det första elementet

Komplexitet: $\Omega((n!)^2)$

`https://sites.math.northwestern.edu/~mlerma/papers-and-preprints/inefficient_algorithms.pdf`

# Worstsort – even worse

1. Generera alla permutationer av indatat, spara i en lista
2. Sortera listan lexiografiskt med hjälp av **worstsort**, om vi inte har nått ett förutbestämt djup ännu...
3. Ta det första elementet

Komplexitet: $\Omega(((n!)\ldots!)^2)$

`https://sites.math.northwestern.edu/~mlerma/papers-and-preprints/inefficient_algorithms.pdf`

# Solar bitflip sort

1. Se om arrayen är sorterad
2. Vänta 10 sekunder, hoppas att bitar i minnet har ändrats
3. Repetera tills sorterad!

# Andra intressanta algoritmer

- *Ineffective Sorts*
  `https://xkcd.com/1185/`
- *Stacksort:*
  `https://gkoberger.github.io/stacksort/`
- Med flera...

# I kursen framöver

- Denna veckan
  - Försök bli klar med lab 3 under veckan
  - Lektion på torsdag (sortering, 2/3 pass körs ute)
- Nästa vecka
  - Tentaförberedelse
- Uppgifter i Kattis
  - `birds` (enkel)
    Hur många fåglar får plats på en elledning?
  - `subway` (svårare)
    Du åker runt i ett tunnelbanesystem utformat som ett träd. Har du åkt runt i samma tunnelbanesystem som din kompis?

# Påbyggnadskurser (masternivå)

| | |
|---|---|
| TDDD95 | Algoritmisk problemlösning<br>Bygg ditt eget bibliotek med användbara algoritmer, använd det för att lösa Kattisproblem |
| TDIU16 | Process- och operativsystem<br>Multitrådning samt operativsystemsprogrammering |
| TDDC78 | Programmering av parallelldatorer<br>Analysera parallella algoritmer, programmera superdatorer |
| TDDD56 | Multicore- och GPU-programmering<br>Programmera och analysera parallella |

Filip Strömbäck

www.liu.se

LINKÖPING UNIVERSITY