TDDI16 – Föreläsning 9

Sortering i linjär tid, beräkningsbarhet

Filip Strömbäck

LINKÖPING UNIVERSITY

Planering

Vecka	Fö	Lab
35	Komplexitet, Linjära strukturer	----
36	Träd, AVL-träd	1---
37	Hashning	1---
38	Grafer och kortaste vägen	12--
39	Fler grafalgoritmer	-23-
40	Sortering	--3-
41	Mer sortering, beräkningsbarhet	--34
42	Tentaförberedelse	---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

Original:

3	1	2	3	2	0

Count sort – Idé

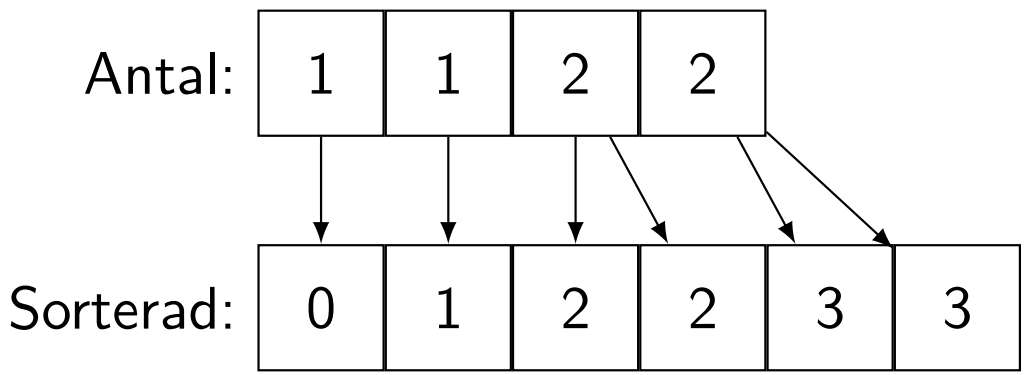
1. Skapa en array av heltal med ett index för varje möjligt tal i indatat
2. Räkna förekomster av alla heltal i indatan, lagra dem i arrayet
3. Skriv tillbaka talen i ordning utifrån antalet

Original:	3	1	2	3	2	0

Antal:	1	1	2	2

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 talen varierar mycket blir det snabbt mycket minne som krävs...

Idé: Vi sorterar flera gånger!

- Sortera efter den minst signifikanta siffran
- Fortsätt sedan 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);
  }
}
```

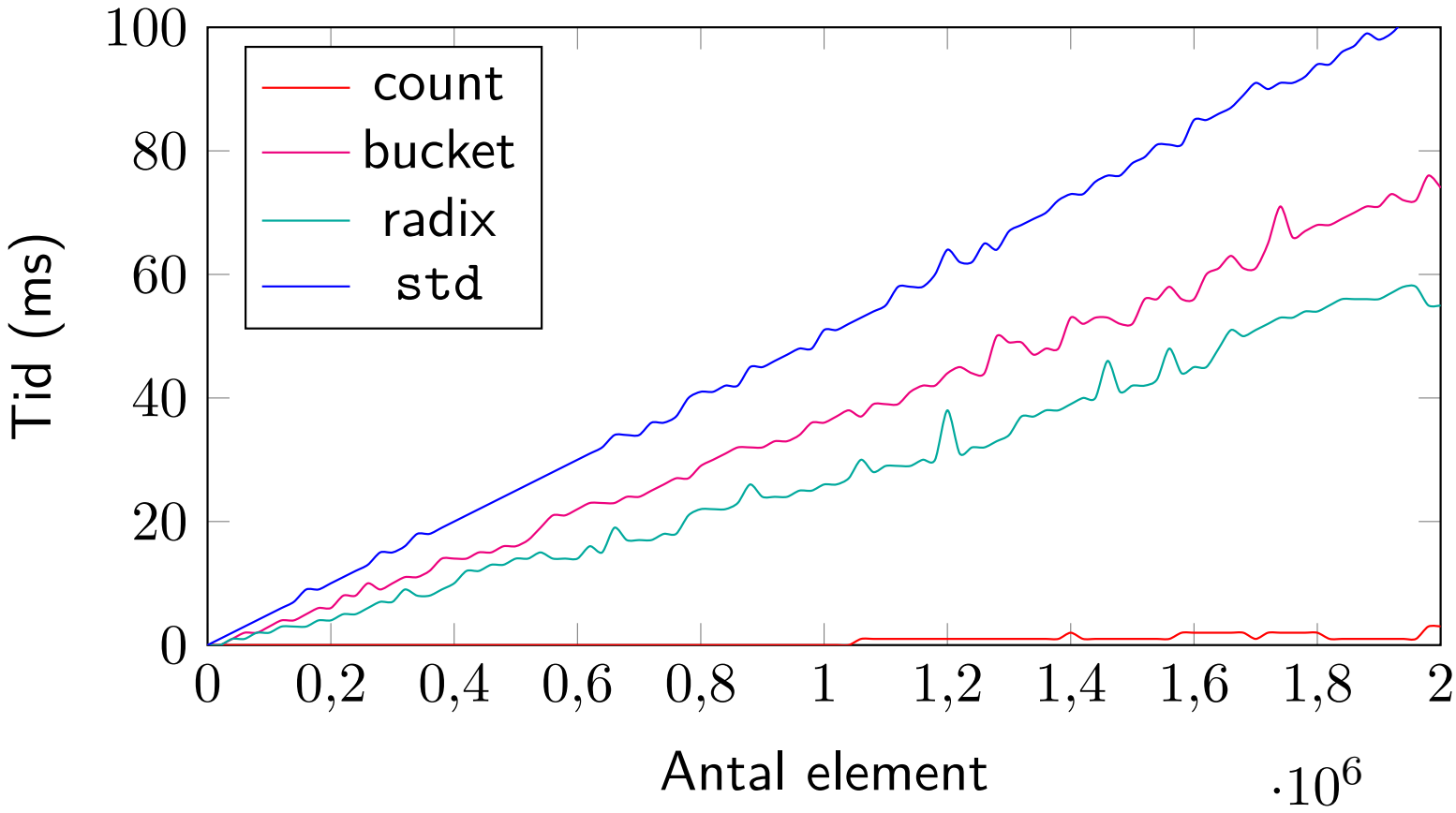
Radix sort – modifierad bucket sort

```
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

Sök efter tal

Baserat på problem 11991 från UVA.

- Hitta indexet av det k:te förekomsten av ett tal v i en lista.
- Listan är $n \leq 100000$ element lång.
- Du ska svara på $m \leq 100000$ frågor.

Vad är värsta möjliga fallet? Hur kan vi effektivisera det?

Fördela röster

Baserat på problem 12390 från UVA.

- Det finns n städer, vardera med p_i invånare
- Alla ska rösta på valdagen
- Du har b valurnor, hur ska du fördela dem så att den urnan med flest röster innehåller så få röster som möjligt?
- $1 \leq n \leq 500000$, $n \leq b \leq 2000000$, $p_i \leq 5000000$

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: $\Omega(((n!)\ldots!)^2) = \Omega((n!^{(k)})^2)$

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

Dropsort

Iterera genom listan, börja med det andra elementet:

- Jämför elementet med det förra.
- Om det var mindre än det förra, ta bort det.

Dropsort is what is known in the computer science field as a lossy algorithm...

`https://www.dangermouse.net/esoteric/dropsort.html`

Andra intressanta algoritmer

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

LINKÖPING UNIVERSITY

Kan vi lösa alla problem?

The Halting Problem

$h(p)$ är sant om programmet p terminerar

Kan vi lösa alla problem?

The Halting Problem

$h(p)$ är sant om programmet p terminerar

Vi skriver ett program, $q(p)$:

- Anropa $h(p)$, spara resultatet i t
- Om t är sant, loopa för evigt

Vad borde $h(q)$ ge för svar?

Kan vi lösa alla problem?

The Halting Problem

$h(p)$ är sant om programmet p terminerar

Vi skriver ett program, $q(p)$:

- Anropa $h(p)$, spara resultatet i t
- Om t är sant, loopa för evigt

Vad borde $h(q)$ ge för svar?

Om $h(q)$ är sant: q terminerar ej $\Rightarrow$ motsägelse

Om $h(q)$ är falskt: q terminerar $\Rightarrow$ motsägelse

Kan vi lösa alla problem?

The Halting Problem

$h(p)$ är sant om programmet p terminerar

Vi skriver ett program, $q(p)$:

- Anropa $h(p)$, spara resultatet i t
- Om t är sant, loopa för evigt

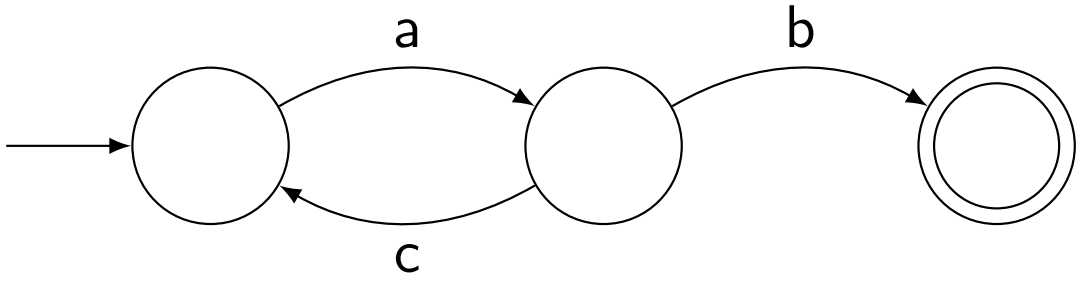
Vad borde $h(q)$ ge för svar?

Om $h(q)$ är sant: q terminerar ej $\Rightarrow$ motsägelse

Om $h(q)$ är falskt: q terminerar $\Rightarrow$ motsägelse

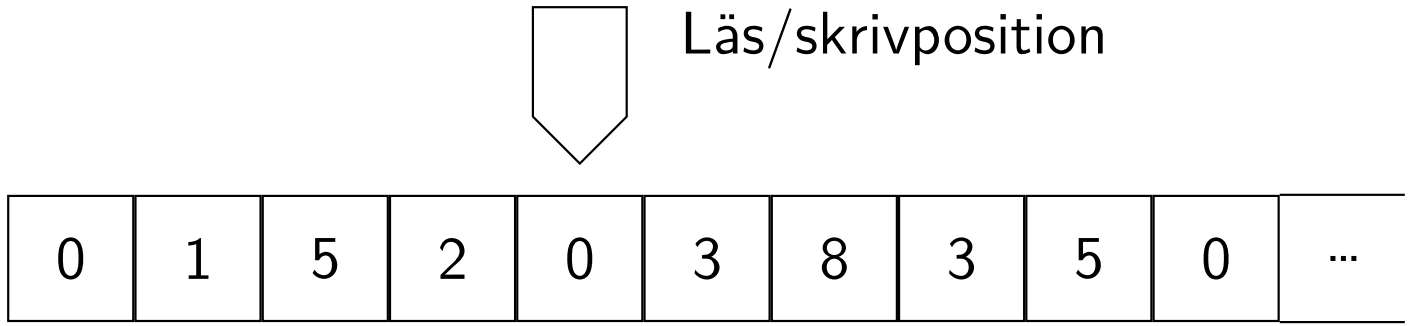
$\Rightarrow$ Nej, vi kan inte lösa alla problem.

Modell – Finit Automata



- Läser indata, går till nästa tillstånd
- Finns två varianter: deterministisk (DFA), ickedeterministisk (NFA)
- DFA och NFA är lika starka

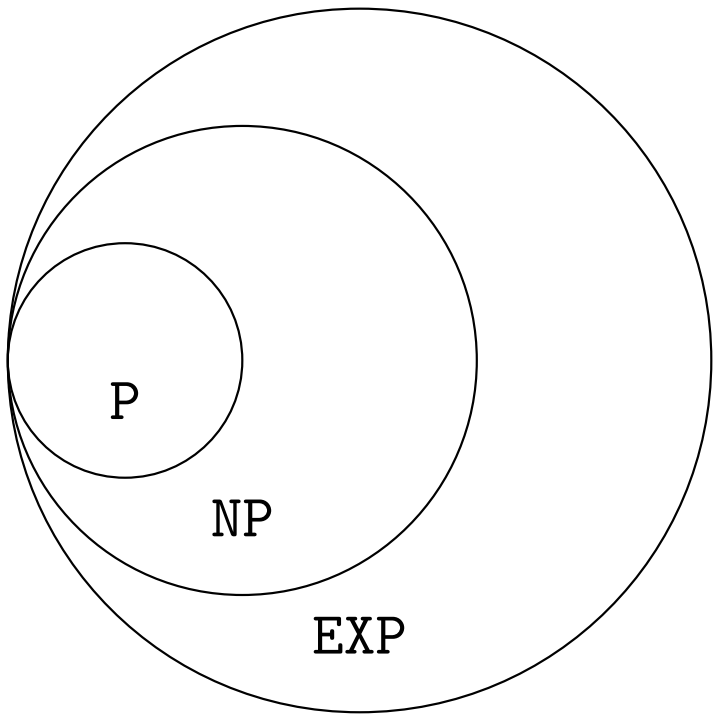
Modell – Turingmaskiner



- Kan *läsa*, *skriva*, *flytta* huvudet fram eller tillbaka, och *terminera*.
- Ekvivalent med ”vanliga” datorer (men kräver oftast fler steg)
- Enkel att göra bevis för

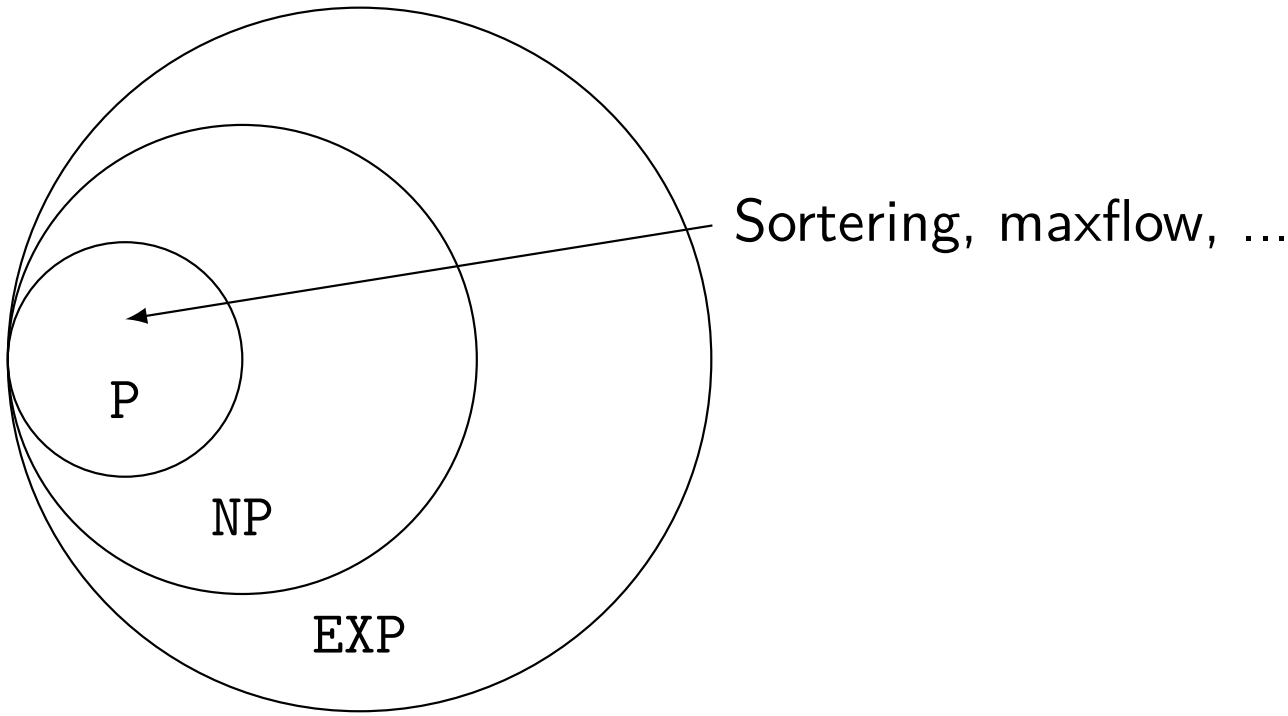
Komplexitetsklasser

Olika problem är olika “svåra”:



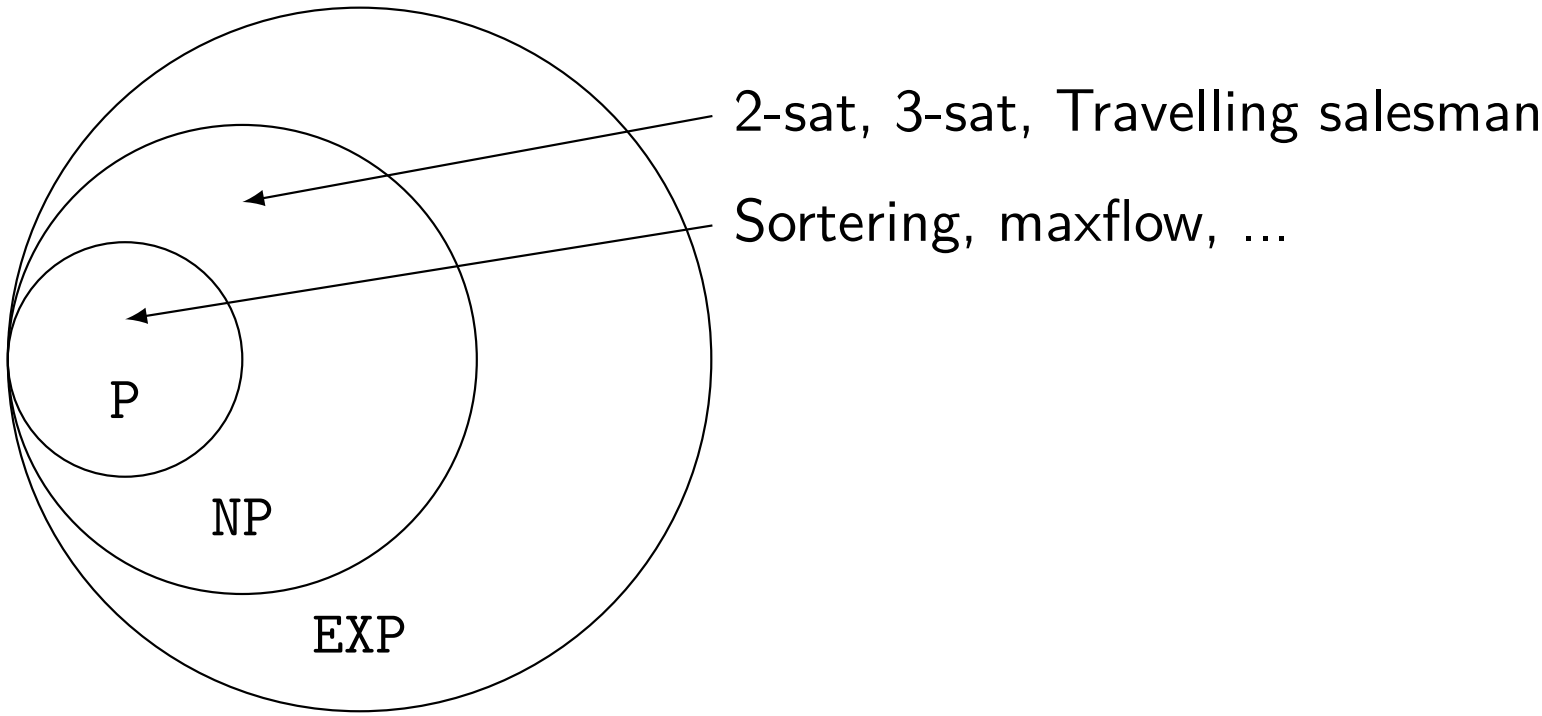
Komplexitetsklasser

Olika problem är olika “svåra”:



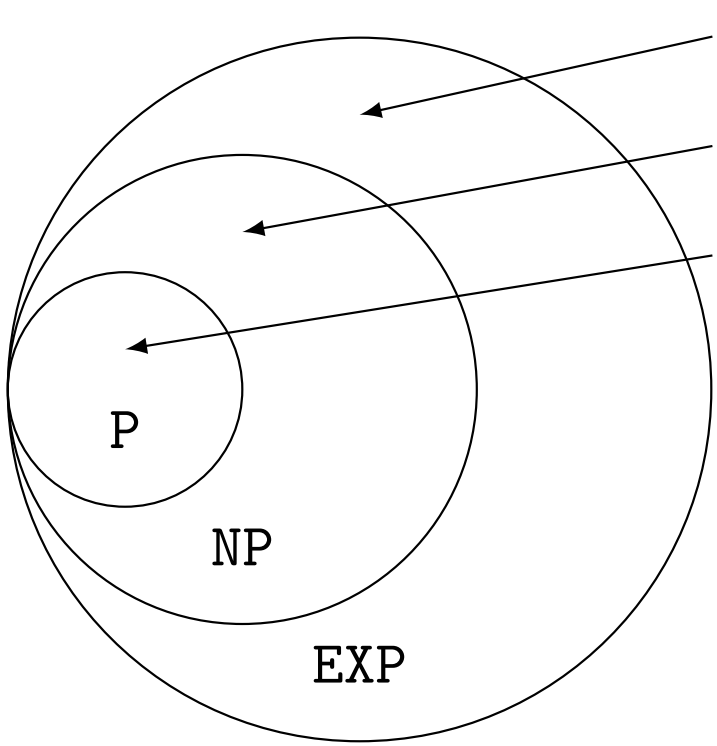
Komplexitetsklasser

Olika problem är olika “svåra”:



Komplexitetsklasser

Olika problem är olika “svåra”:



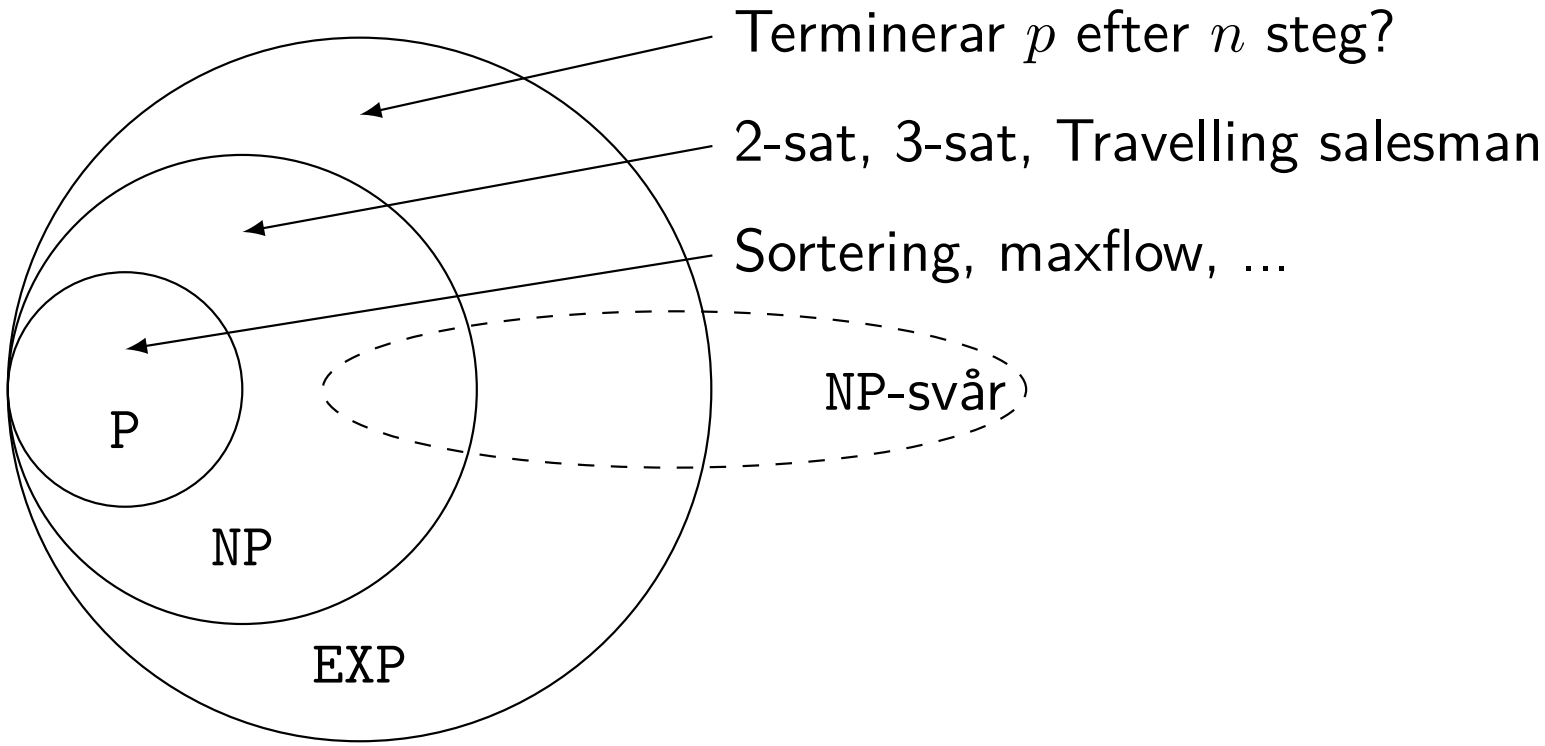
Terminerar p efter n steg?

2-sat, 3-sat, Travelling salesman

Sortering, maxflow, ...

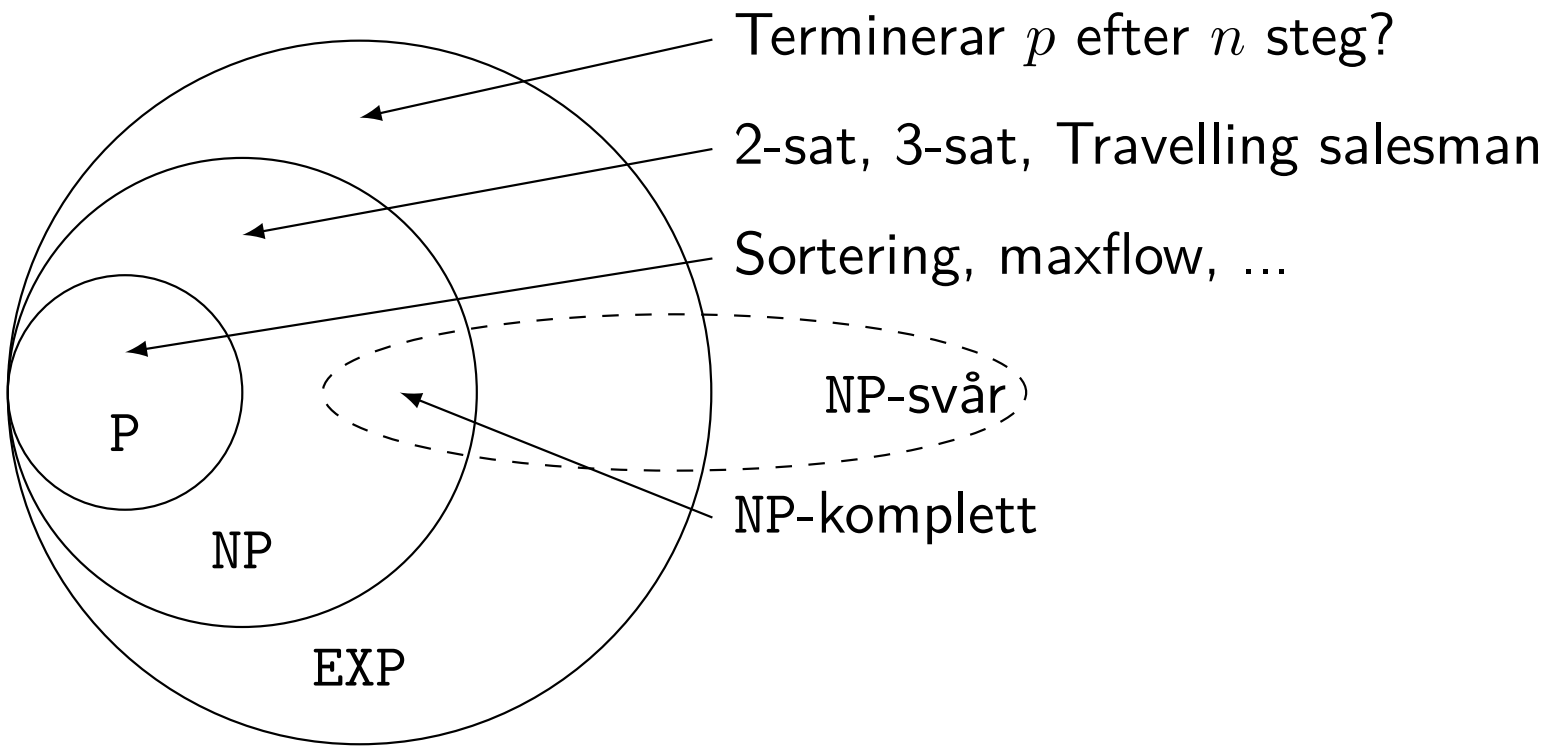
Komplexitetsklasser

Olika problem är olika "svåra":



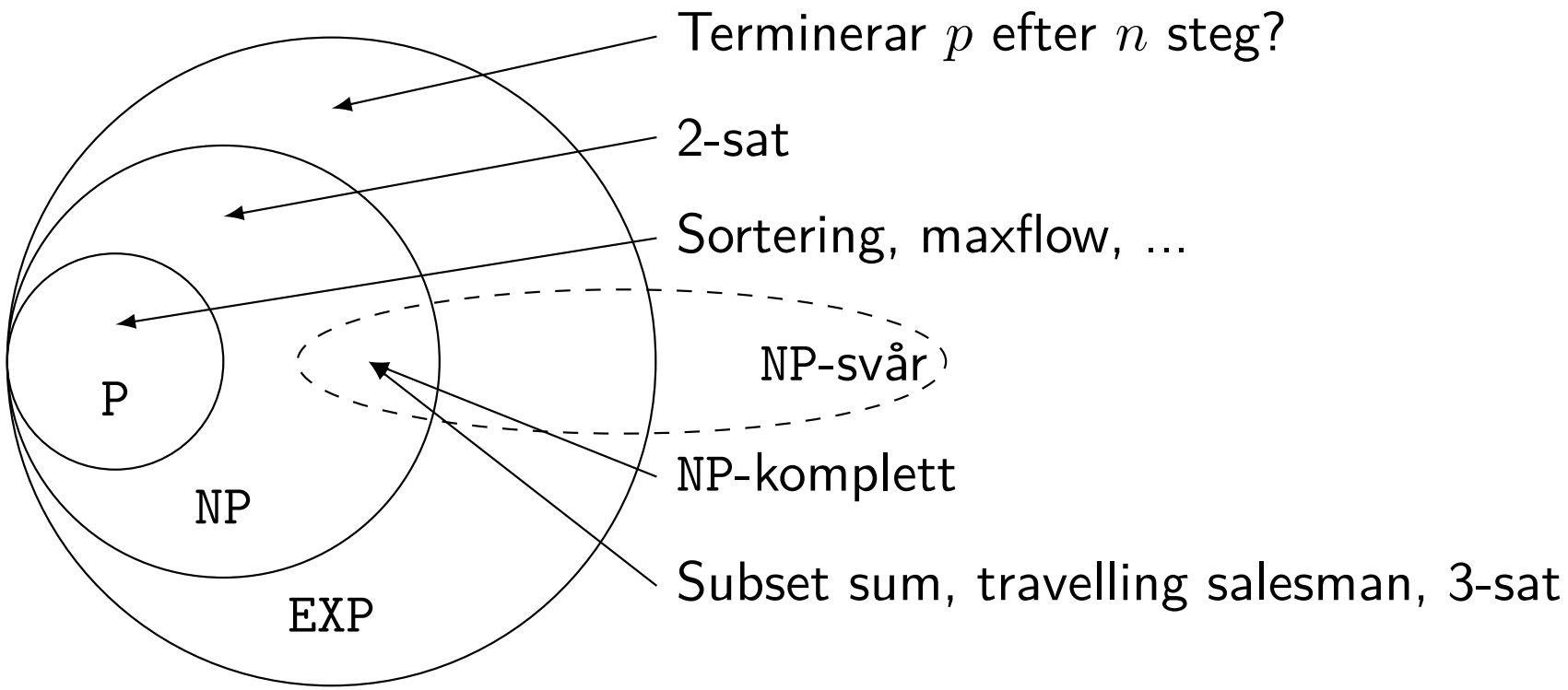
Komplexitetsklasser

Olika problem är olika "svåra":



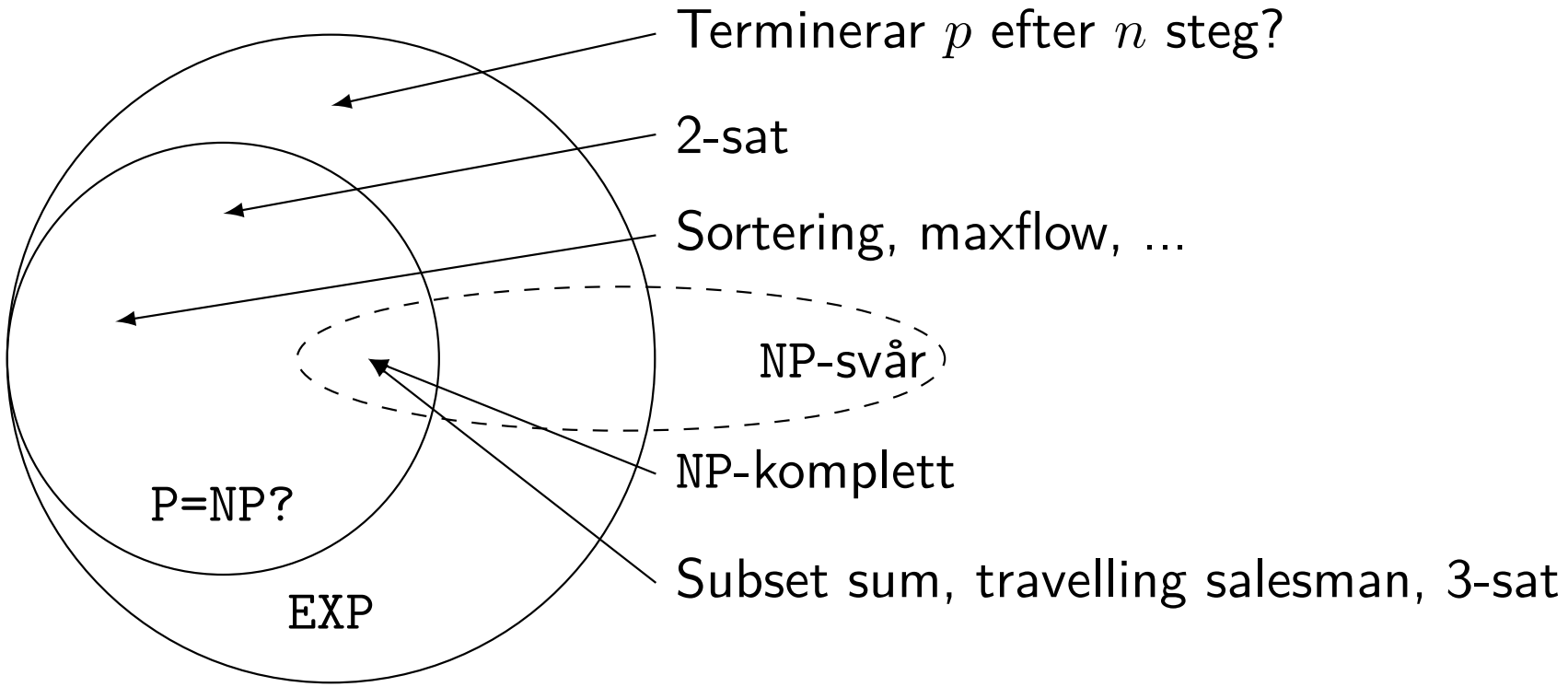
Komplexitetsklasser

Olika problem är olika “svåra”:



Komplexitetsklasser

Olika problem är olika “svåra”:



LINKÖPING UNIVERSITY

I kursen framöver

- Denna veckan
  - Börja med lab 4 denna vecka, nästa vecka är mycket självstudietid
- 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 (vissa på masternivå)

TDIU16	Process- och operativsystem Multitrådning och operativsystem
TDDD95	Algoritmisk problemlösning Bygg ditt eget bibliotek med användbara algoritmer för att lösa Kattisproblem
TDDC78	Programmering av parallelldatorer Analysera parallella algoritmer, programmera superdatorer
TDDD56	Multicore- och GPU-programmering Programmera och analysera parallella algoritmer, delvis på GPU

Filip Strömbäck

www.liu.se