29. September 2004

Hvordan virker en rejseplanner?

Henning Christiansen
professor i datalogi, ph.d.
http://www.ruc.dk/~henning

Datalogiafdelingen, hus 42.1
Roskilde Universitetscenter
Universitetsvej 1
Postboks 260
4000 Roskilde
Telefon: 4674 2000
Fax: 4674 3072
www.ruc.dk/dat

Præsentation

Datalogi på RUC uddanner (bl.a.)

- Tværfaglig kanditater Basis + X + Datalogi = 5 år
- Datalogisk grundviden + kombi m. X
- Rejseplanner, hvordan virker den?
  - Eksempel som kræver »grundviden«

Mine egne forskningsinteresser

- Logikbaserede systemer, logik. progr.,...
- Teknologier til analyse af naturligt sprog
- Databaser
- Forespørgselssystemer generelt
- Metoder til formidling af datalogi (univ.-niveau)

Rejseplanner?

http://www.rejseplanen.dk
http://bahn.hafas.de/

Hvad skal der til for at det virker?

- Internet, tilgængelig webside
- Korrekte & fuldstændige køreplansdata (!!)
- Et program med en ret genial algoritme
  - hvad er »en algoritme«?

Det grundlæggende problem, kortest vej

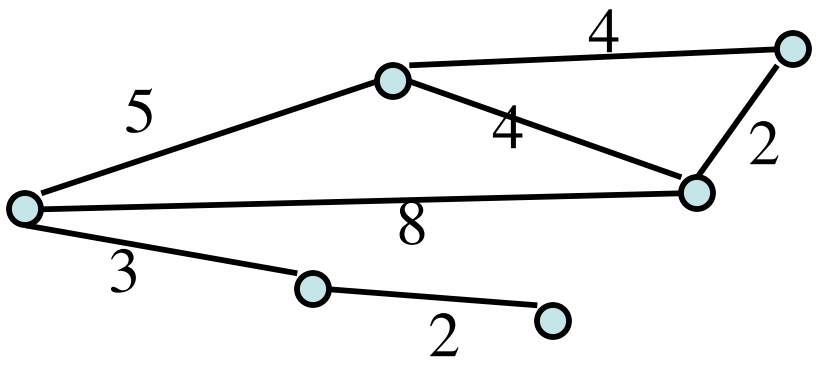
Matematiker taler om grafer, knuder + kanter

Her: vejnet, byer/stationer, direkte forbindelser som sættes sammen til rejseruter

Det grundlæggende problem, kortest vej

Matematiker taler om grafer, knuder + kanter

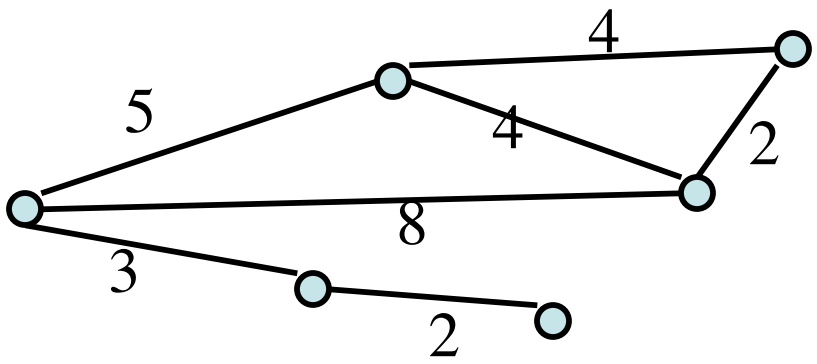
Her: vejnet, byer/stationer, direkte forbindelser som sættes sammen til rejseruter



Det grundlæggende problem, kortest vej

Matematiker taler om grafer, knuder + kanter

Her: vejnet, byer/stationer, direkte forbindelser som sættes sammen til rejseruter



Et menneske kan hurtigt regne sig frem til korteste vej her

Men
- når landkortet bliver stort?
- hurtige og langsomme tog
- forklare princip så en dum computer kan forstå det!

Klamphuggeralgoritmen

Klamphuggeralgoritmen

- Generér samtlige mulige ruter [uden cirkler]

Klamphuggeralgoritmen

- Generér samtlige mulige ruter [uden cirkler]
- Beregn længden for hver af dem (... + ... + ...)

Klamphuggeralgoritmen

- Generér samtlige mulige ruter [uden cirkler]
- Beregn længden for hver af dem (... + ... + ...)
- Vælg den som har mindst længde

Klamphuggeralgoritmen

- Generér samtlige mulige ruter [uden cirkler]
- Beregn længden for hver af dem (... + ... + ...)
- Vælg den som har mindst længde
- Voila!

Klamphuggeralgoritmen

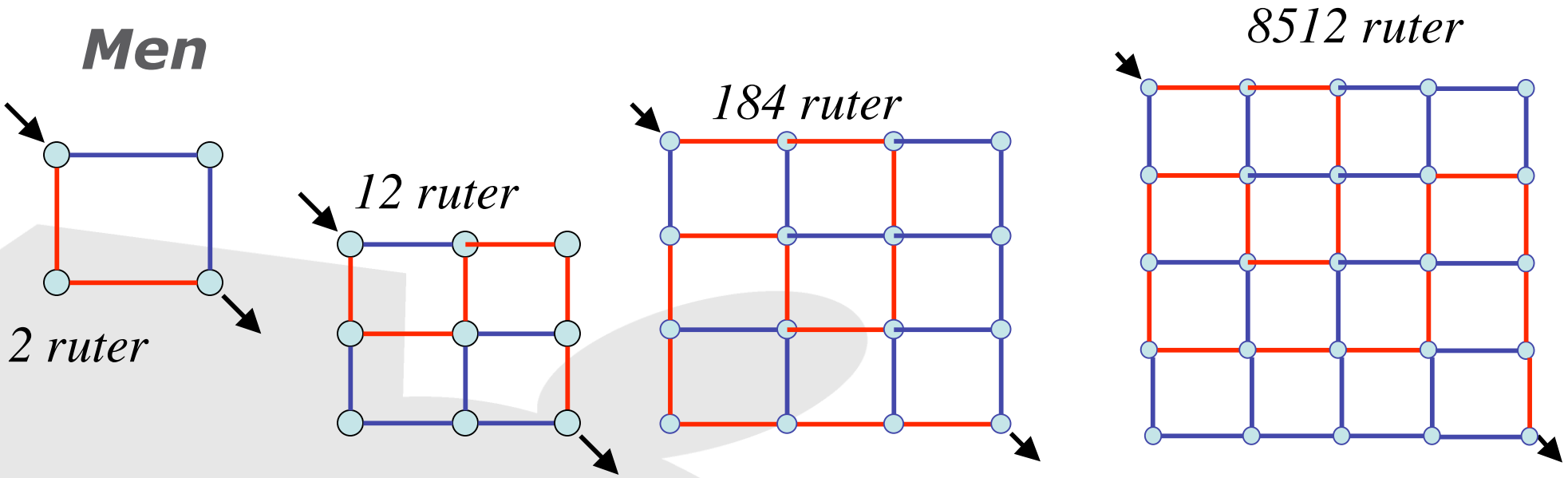
- Generér samtlige mulige ruter [uden cirkler]
- Beregn længden for hver af dem (... + ... + ...)
- Vælg den som har mindst længde
- Voila!

Men

Klamphuggeralgoritmen

- Generér samtlige mulige ruter [uden cirkler]
- Beregn længden for hver af dem (... + ... + ...)
- Vælg den som har mindst længde
- Voila!

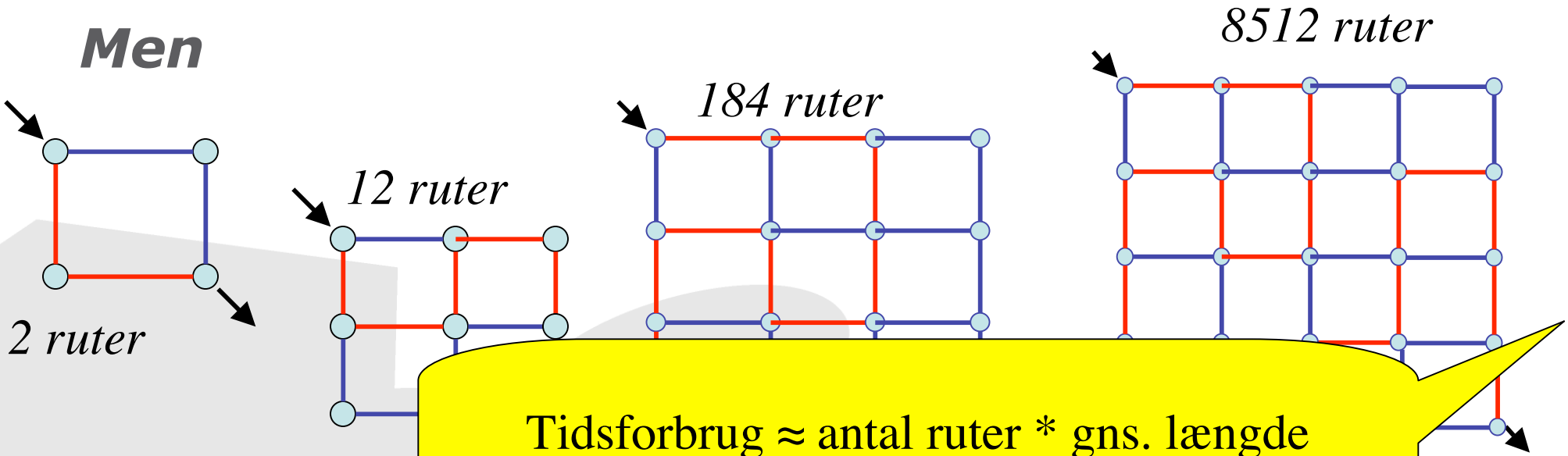
Men



Klamphuggeralgoritmen

- Generér samtlige mulige ruter [uden cirkler]
- Beregn længden for hver af dem (... + ... + ...)
- Vælg den som har mindst længde
- Voila!

Men



E.W.Dijkstras algoritme til korteste rute [1959]

E.W.Dijkstras algoritme til korteste rute [1959]

Korteste vej fra X til Y.

E.W.Dijkstras algoritme til korteste rute [1959]

Korteste vej fra X til Y.

- Princip: generer korteste veje i »cirkler« væk fra X indtil vi har indfanget Y

E.W.Dijkstras algoritme til korteste rute [1959]

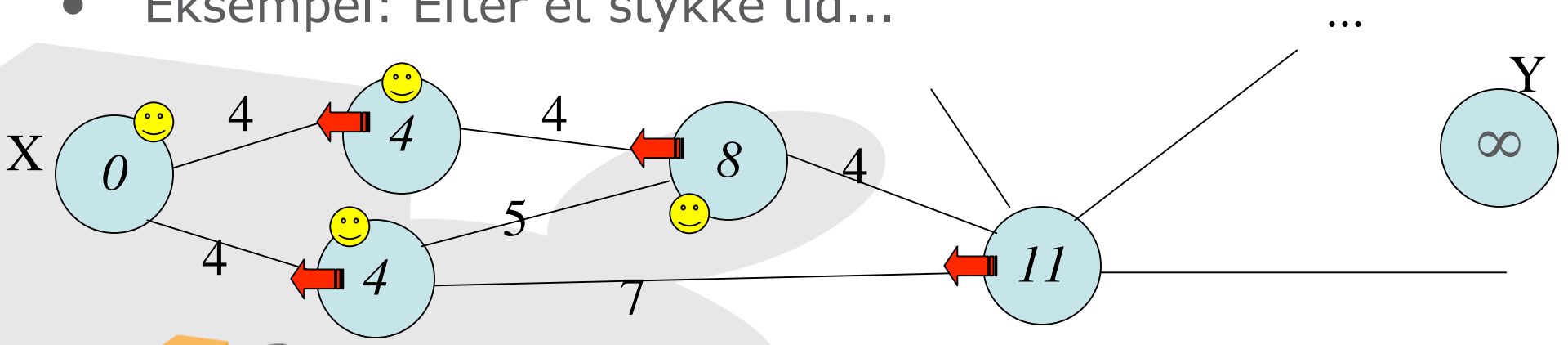
Korteste vej fra X til Y.

- Princip: generer korteste veje i »cirkler« væk fra X indtil vi har indfanget Y
- Datastruktur:
  - Et vejnet med X og Y markeret
  - Hver by kan huske
    - *afstand:* lgd. af korteste, *kendte* rute; start:X = 0, alle andre = ∞
    - *Forgænger:* Hvor kom den rute fra  (hvis relevant)
    - Evt. *mærke for »færdig«* 

E.W.Dijkstras algoritme til korteste rute [1959]

Korteste vej fra X til Y.

- Princip: generer korteste veje i »cirkler« væk fra X indtil vi har indfanget Y
- Datastruktur:
  - Et vejnet med X og Y markeret
  - Hver by kan huske
    - *afstand:* lgd. af korteste, *kendte* rute; start:X = 0, alle andre = ∞
    - *Forgænger:* Hvor kom den rute fra ⬅ (hvis relevant)
    - Evt. *mærke for »færdig«* ☺
- Eksempel: Efter et stykke tid...



E.W.Dijkstras algoritme til korteste rute [1959]

Vej fra X til Y i vejnet.

1. Sæt X afstand = 0
2. Lad A være by *med mindst afst.* blandt ikke-☺
- Sæt status for A til ☺
- For alle A's naboer N som ikke er ☺:
  - Sæt afstand til mindste af
    - evt. gammel afstand for N
    - A's afstand + vejlængde A→N
- Hvis A ikke var Y, gå til 2

Hvor effektiv?

Hvor effektiv?

Propertional med antal veje undersøgt op til Y:

- hver vej op til Y kun passeret 1 gang,
- (plus en lille sjat til at finde den »mindste«)

Hvor effektiv?

Propertional med antal veje undersøgt op til Y:

- hver vej op til Y kun passeret 1 gang,
- (plus en lille sjat til at finde den »mindste«)

Det er ikke godt nok:

- Kvadratisk i rutens længde (suk)

Hvor effektiv?

Propertional med antal veje undersøgt op til Y:

- hver vej op til Y kun passeret 1 gang,
- (plus en lille sjat til at finde den »mindste«)

Det er ikke godt nok:

- Kvadratisk i rutens længde (suk)

Hvor effektiv?

Propertional med antal veje undersøgt op til Y:

- hver vej op til Y kun passeret 1 gang,
- (plus en lille sjat til at finde den »mindste«)

Det er ikke godt nok:

- Kvadratisk i rutens længde (suk)

Optimering af Dijkstra's (Hafas o.a.)

- Tommelfingerregler om lyntog o.lign.
- Analysere landkort i forvejen, opdele i regioner ... *(Find selv artikler på nettet)*