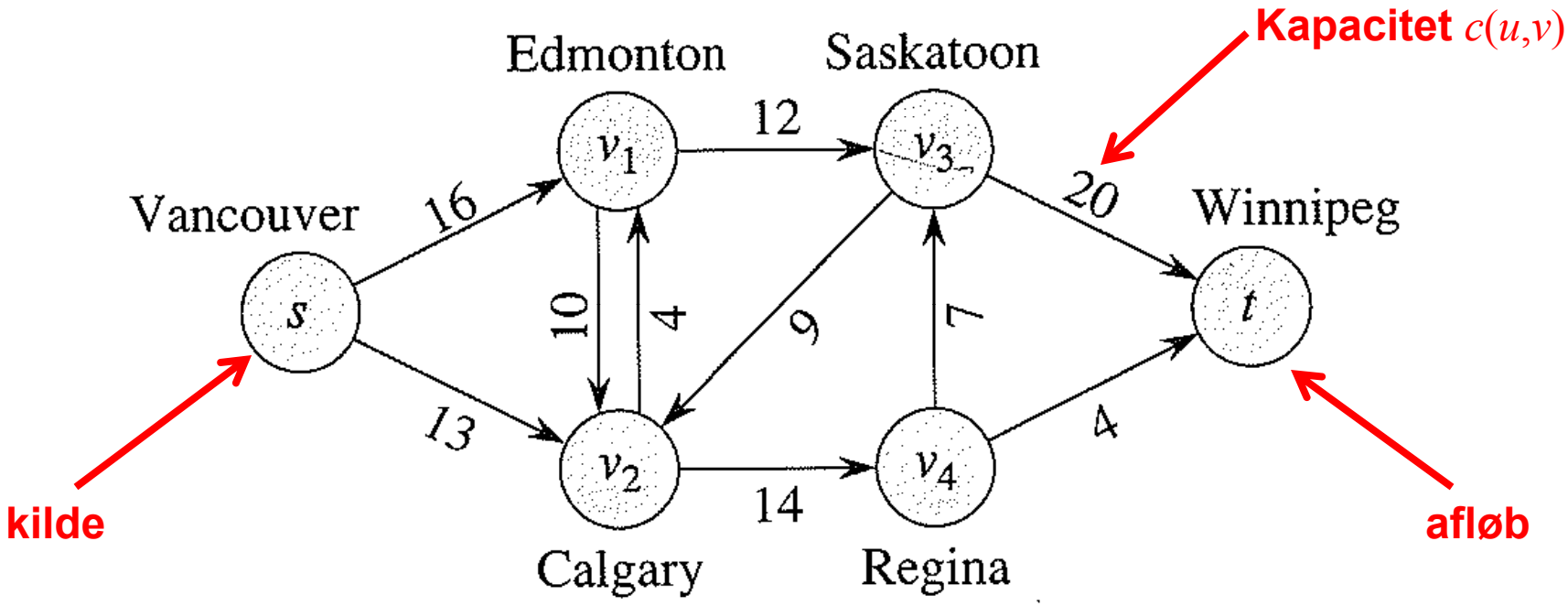
Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

Maksimale Strømninger [CLRS, kapitel 26.1-26.3]

AARHUS UNIVERSITET

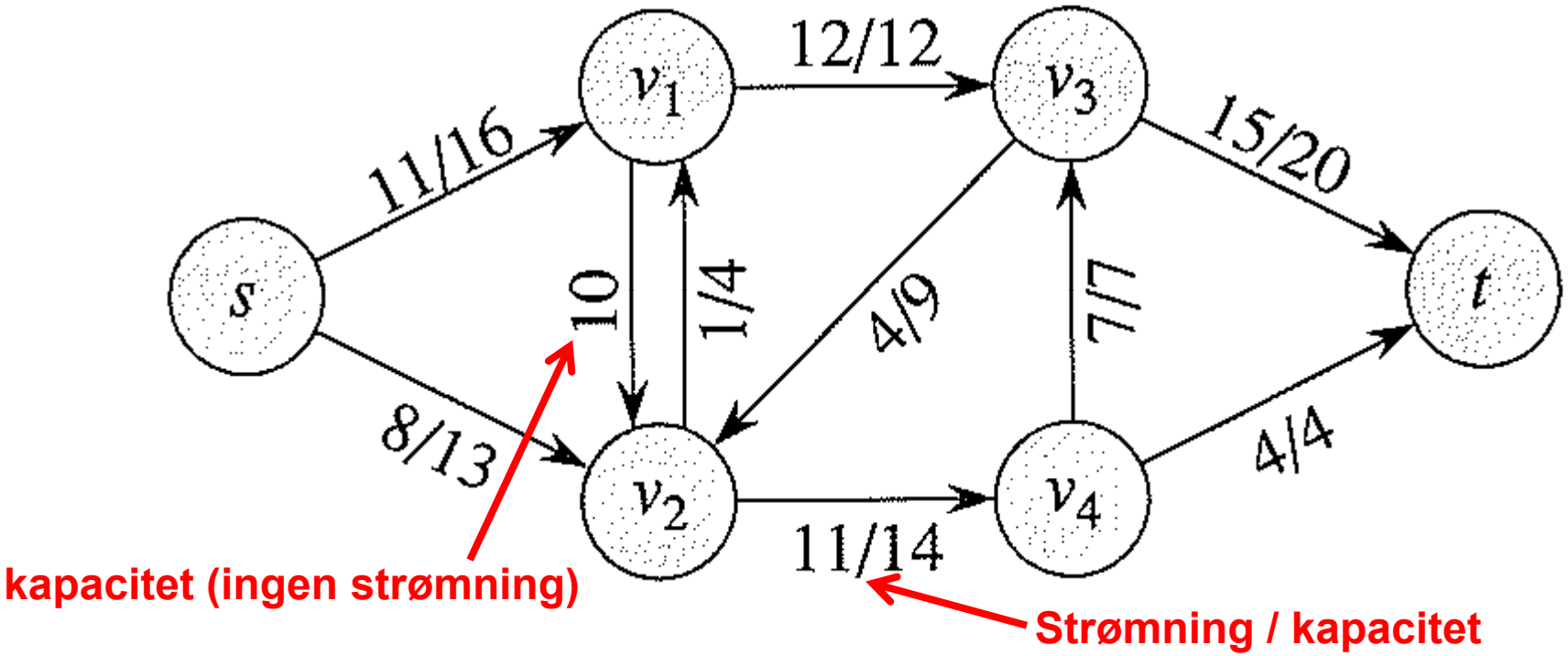
Maximale Strømninger i Netværk



Input: En orienteret graf $G=(V,E)$, hvor alle kanter (u,v) har en kapacitet $c(u,v)$, og to knuder kilde $s \in V$ (source), og afløbet $t \in V$ (sink).

Output: En maximal strømning f i netværket fra s til t.

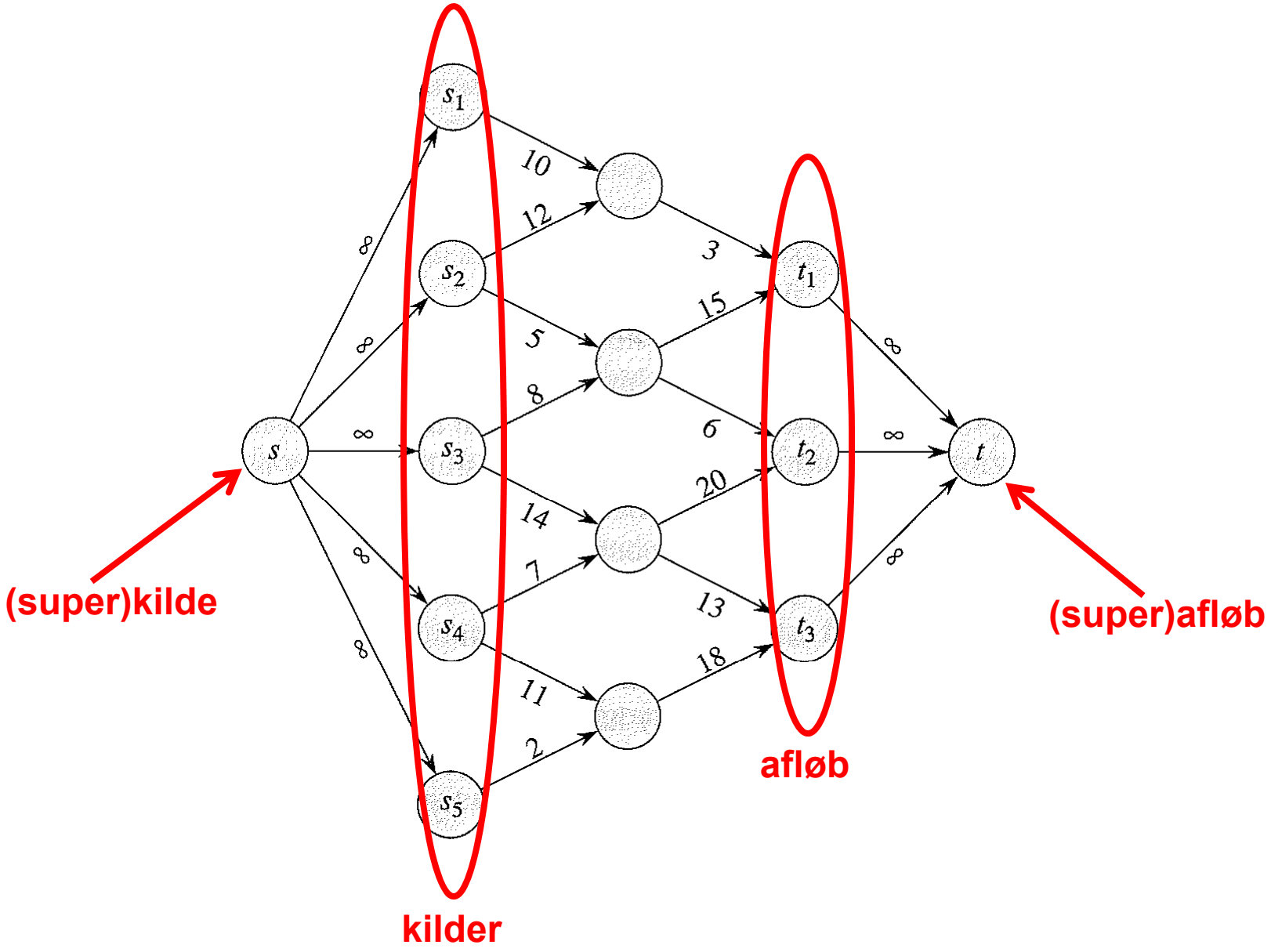
Maximale Strømninger i Netværk



Krav til f :

$f(u,v) = -f(v,u)$	for alle kanter (strømbevaring)
$\Sigma_{v \in V} f(u,v) = 0$	for alle knuder ≠ s,t (strøm in=strøm ud)
$f(u,v) \leq c(u,v)$	for alle kanter (strømbevaring)

Flere kilder og afløb



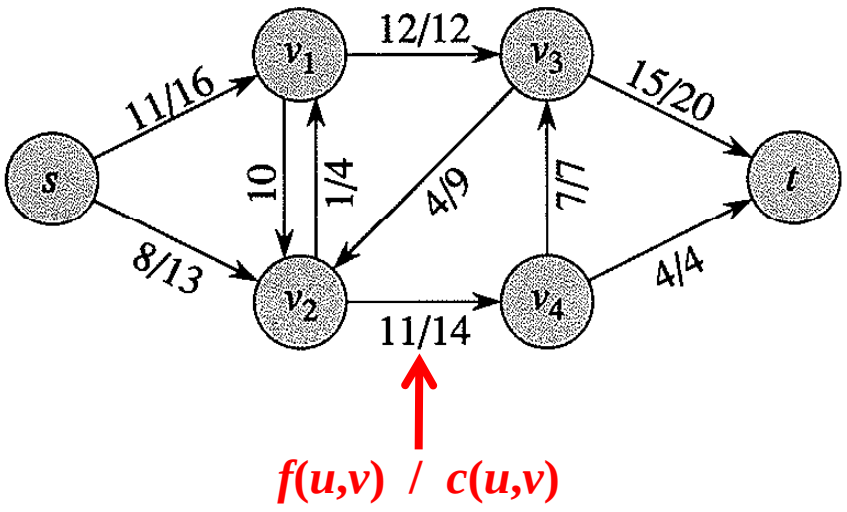
1962

Ford-Fulkerson

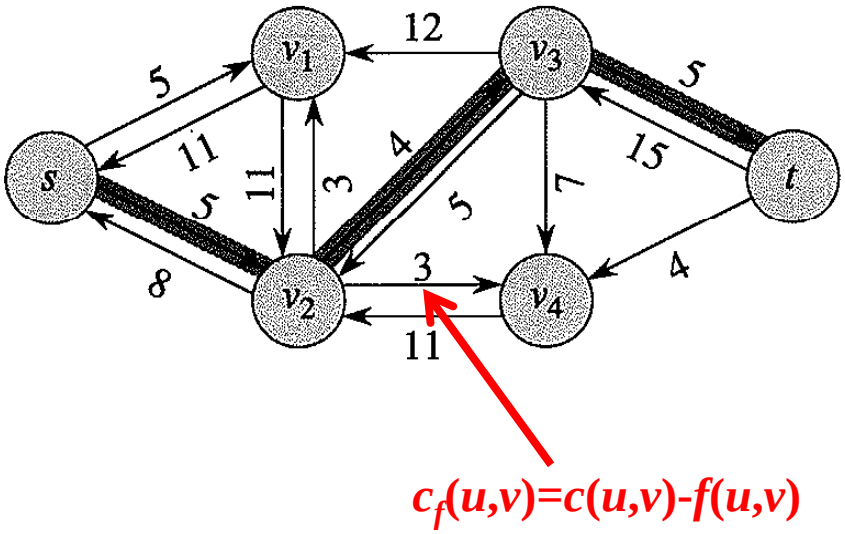
FORD-FULKERSON-METHOD(G, s, t)

1 initialize flow f to 0
2 **while** there exists an augmenting path p in the residual network G_f
3 augment flow f along p
4 **return** f

Strømning / Kapacitet



Rest-netværk & forbedrende sti



Observation: *s-t* **sti i rest-netværket ≡ forbedrende sti i netværket**

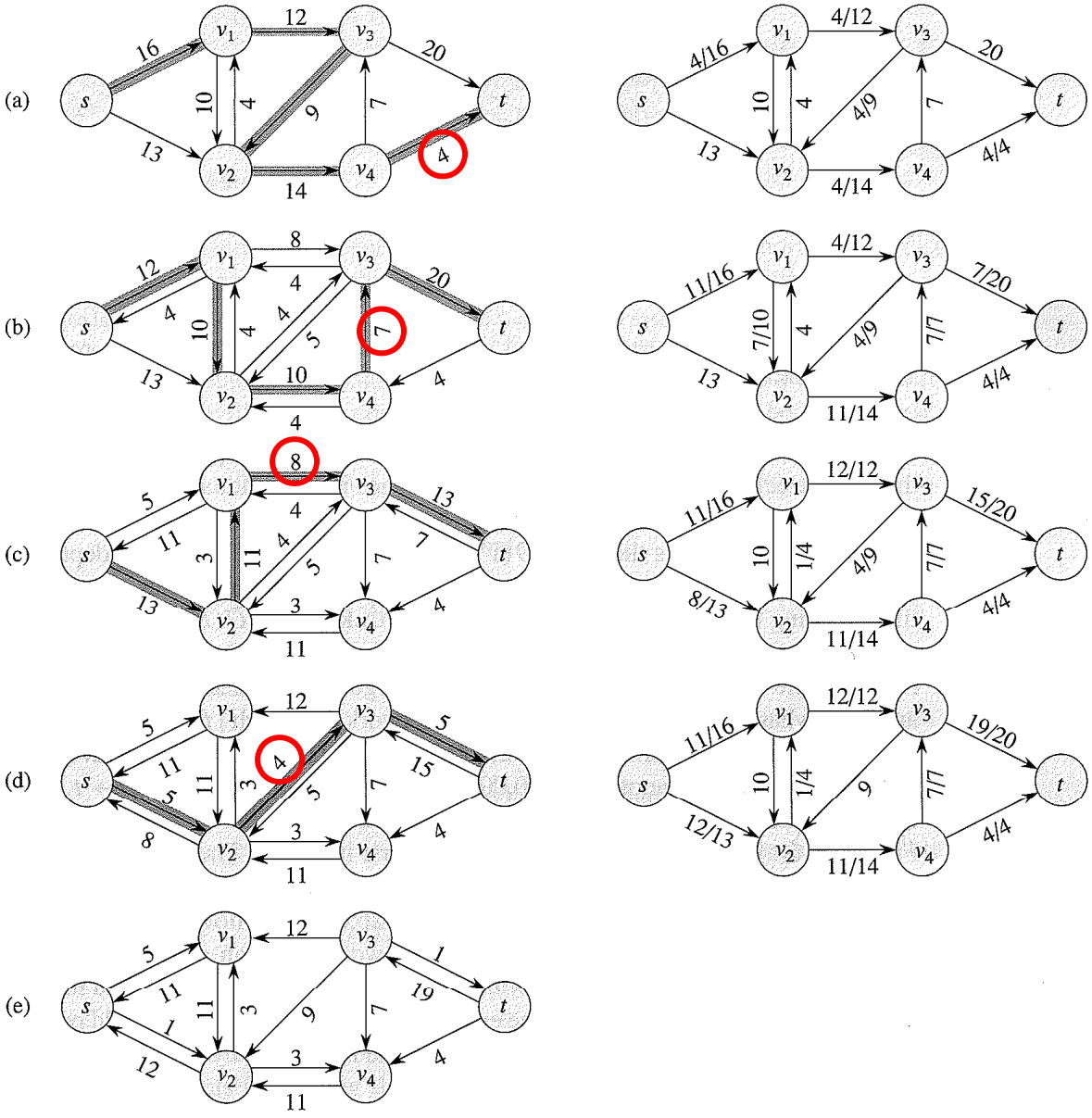
1962

Ford-Fulkerson

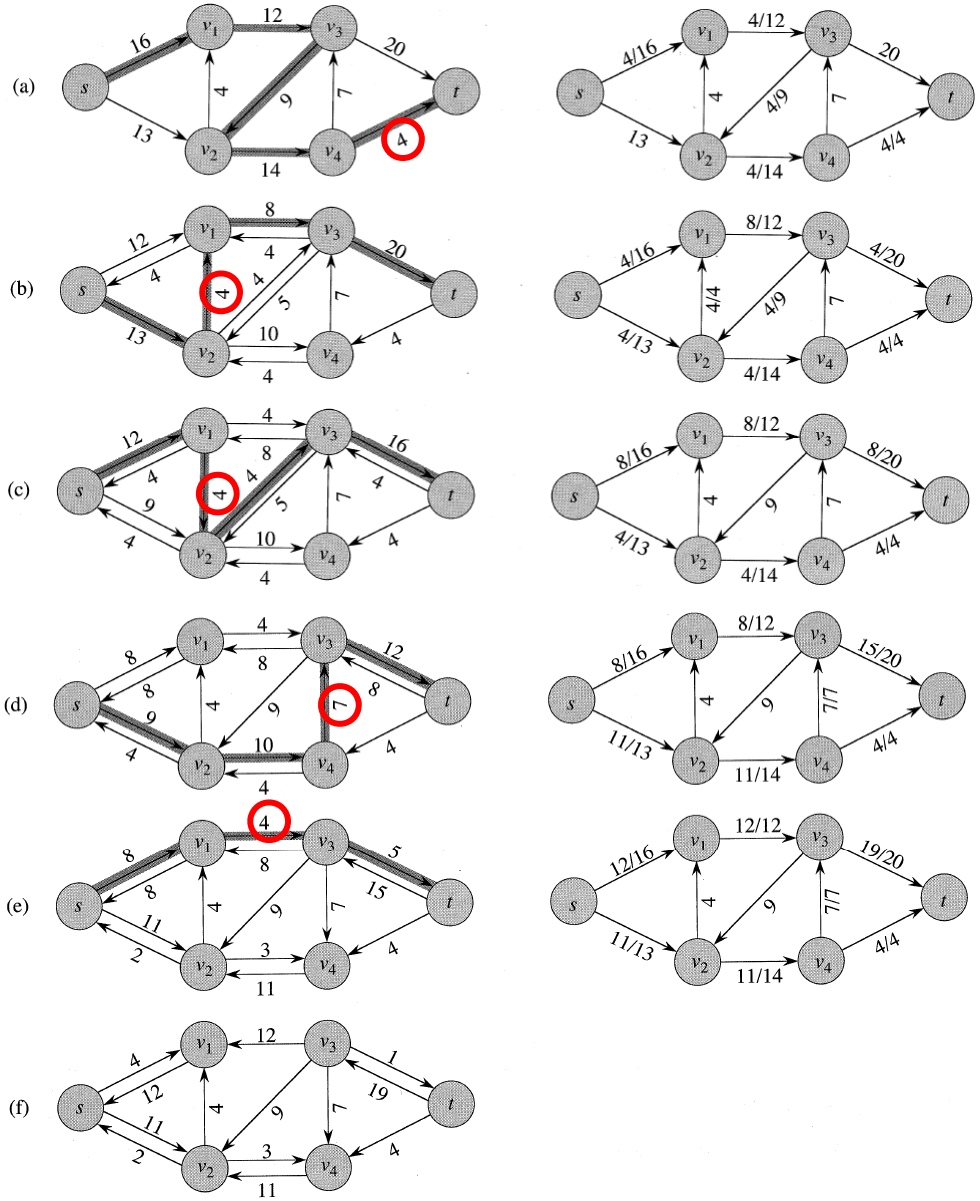
```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ G.E
2      (u, v).f = 0
3  while there exists a path p from s to t in the residual network G_f
4      c_f(p) = min{c_f(u, v) : (u, v) is in p}
5      for each edge (u, v) in p
6          if (u, v) ∈ E
7              (u, v).f = (u, v).f + c_f(p)
8          else (v, u).f = (v, u).f − c_f(p)
```

maximale forøgelse langs stien p

Ford-Fulkerson : eksempel



Ford-Fulkerson : eksempel



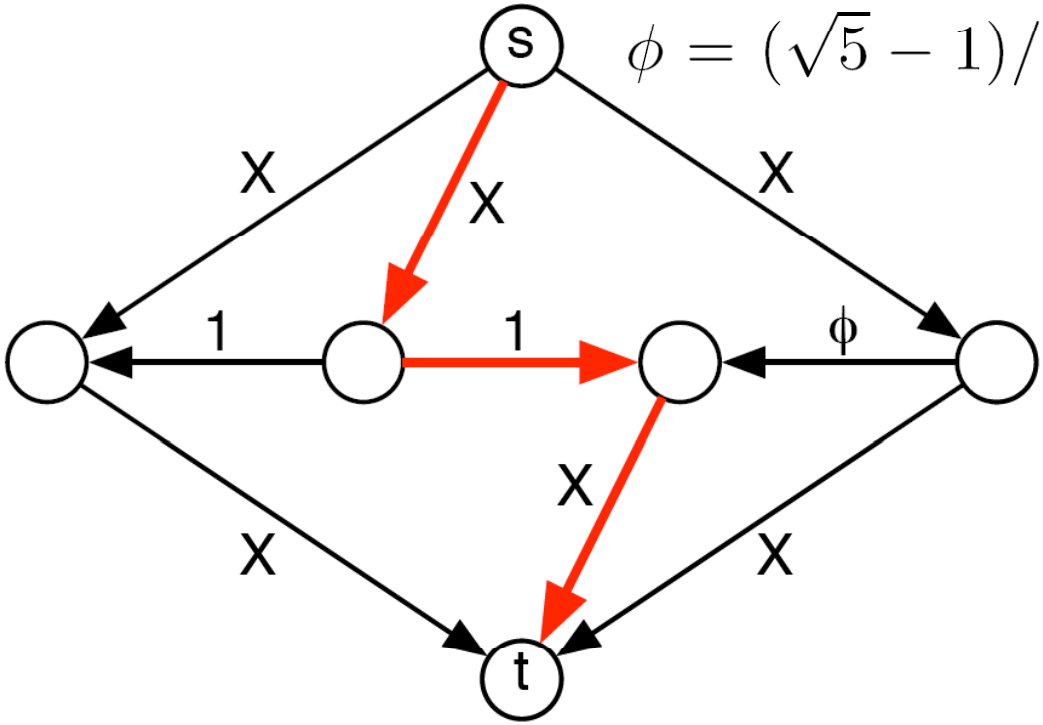
rest-netværk **aktuelle strømning**

Ford-Fulkerson : Analyse

Sætning

Hvis alle kapaciteterne i et netværk er heltallige og f^* er en maximal strømning, så tager Ford-Fulkerson algoritmen tid $O(E \cdot |f^*|)$.

Ford-Fulkerson : Eksempel
∞ mange forbedrende stier



$\phi = (\sqrt{5} - 1)/2 \approx 0.618034$

X = stort tal

Der findes en række (hvilken?) af forbedrende stier som konvergerer mod en strømning af størrelse

$$1 + 2\sum_{i=1}^{\infty} \phi^i = 1 + \frac{2}{1-\phi} = 4 + \sqrt{5} < 7$$

hvilket er langt fra det optimale på 2X+1.

Edmonds-Karp

Edmonds-Karp algoritmen =

Ford-Fulkerson algoritmen der altid vælger en forbedrende sti med færrest mulige kanter (vha BFS)

Sætning

Edmonds-Karp algoritmen finder højst $O(V \cdot E)$ forbedrende stier, d.v.s. kører i tid $O(V \cdot E^2)$.

Maksimale strømninger – Historisk overblik

year	discoverer(s)	bound
1951	Dantzig [11]	$O(n^2mU)$
1956	Ford & Fulkerson [17]	$O(nmU)$
1970	Dinitz [13] Edmonds & Karp [15]	$O(nm^2)$
1970	Dinitz [13]	$O(n^2m)$
1972	Edmonds & Karp [15] Dinitz [14]	$O(m^2 \log U)$
1973	Dinitz [14] Gabow [19]	$O(nm \log U)$
1974	Karzanov [36]	$O(n^3)$
1977	Cherkassky [9]	$O(n^2m^{1/2})$
1980	Galil & Naamad [20]	$O(nm \log^2 n)$
1983	Sleator & Tarjan [46]	$O(nm \log n)$
1986	Goldberg & Tarjan [26]	$O(nm \log(n^2/m))$
1987	Ahuja & Orlin [2]	$O(nm + n^2 \log U)$
1987	Ahuja et al. [3]	$O(nm \log(n\sqrt{\log U}/m))$
1989	Cheriyan & Hagerup [7]	$E(nm + n^2 \log^2 n)$
1990	Cheriyan et al. [8]	$O(n^3/\log n)$
1990	Alon [4]	$O(nm + n^{8/3} \log n)$
1992	King et al. [37]	$O(nm + n^{2+\epsilon})$
1993	Phillips & Westbrook [44]	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
1994	King et al. [38]	$O(nm \log_{m/(n \log n)} n)$
1997	Goldberg & Rao [24]	$O(\min(n^{2/3}, m^{1/2})m \log(n^2/m) \log U)$

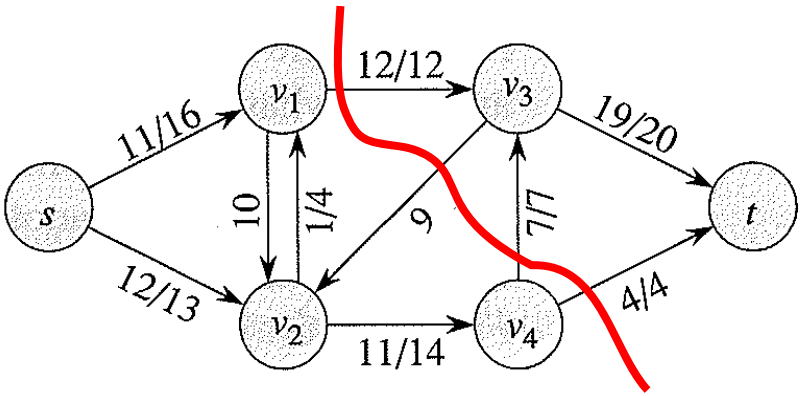
Andrew Goldberg, 1998

***n* = # knuder, *m* = # kanter, kapaciteter i intervallet [1..*U*]**

Maximale strømninger

Sætning

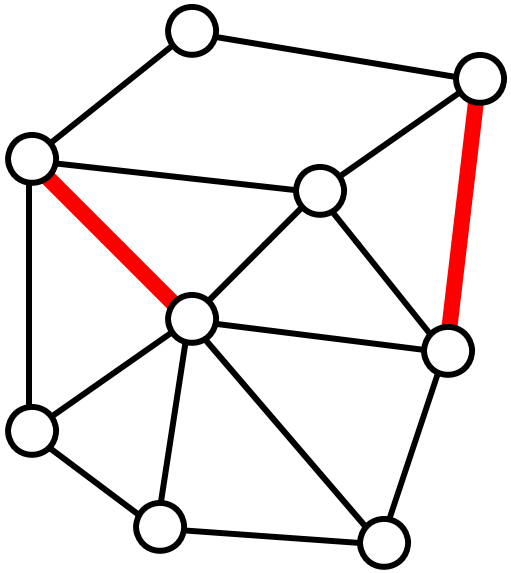
Maximal strømning = minimal snit.



Sætning

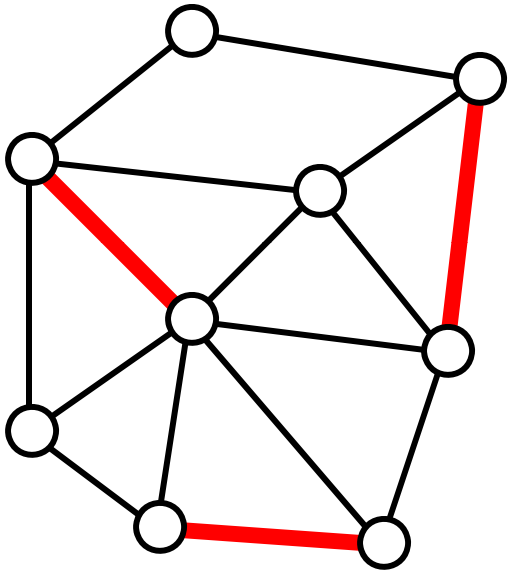
Hvis alle kapaciteter er heltallige så finder Ford-Fulkerson og Edmonds-Karp algoritmerne en strømning hvor *strømmen langs alle kanter er heltalligt*

Parringer



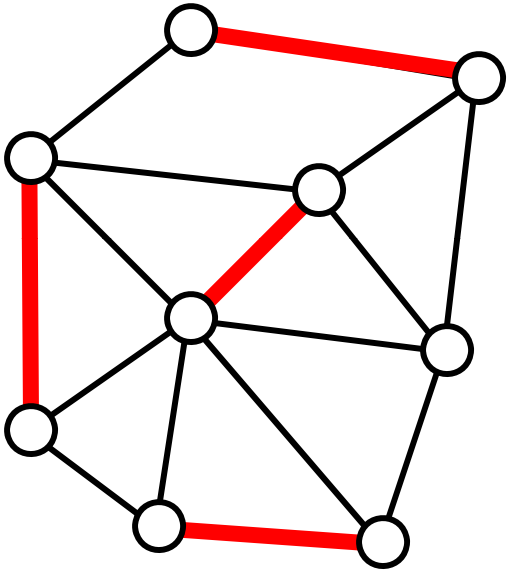
Parring

(en delmængde af kanterne hvor hver knude indgår max én gang)



Maximal Parring

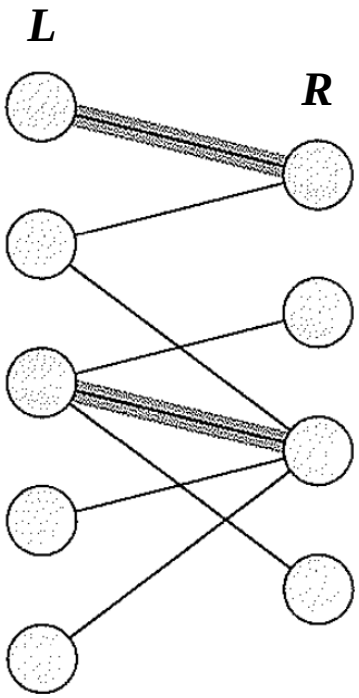
(parring hvor ingen kanter kan tilføjes – kan findes v.h.a grådig algoritme)



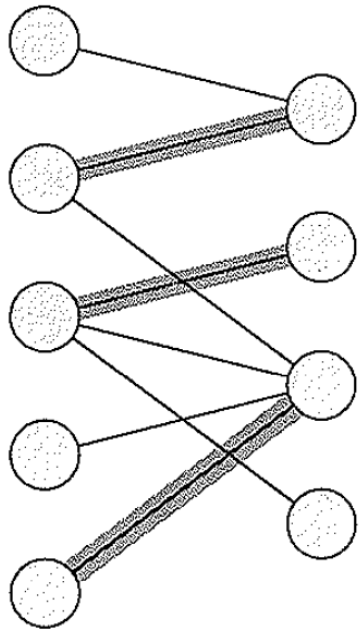
Maximum Parring

(findes ingen parringer med flere kanter)

Parringer i todelte grafer

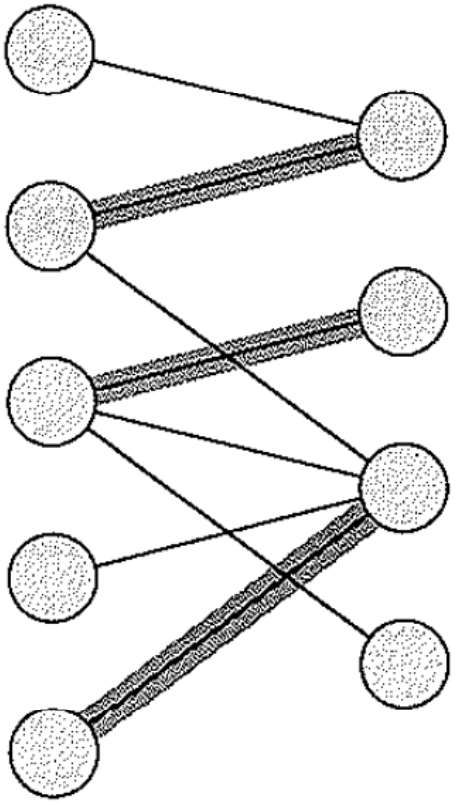


Parring af størrelse 2



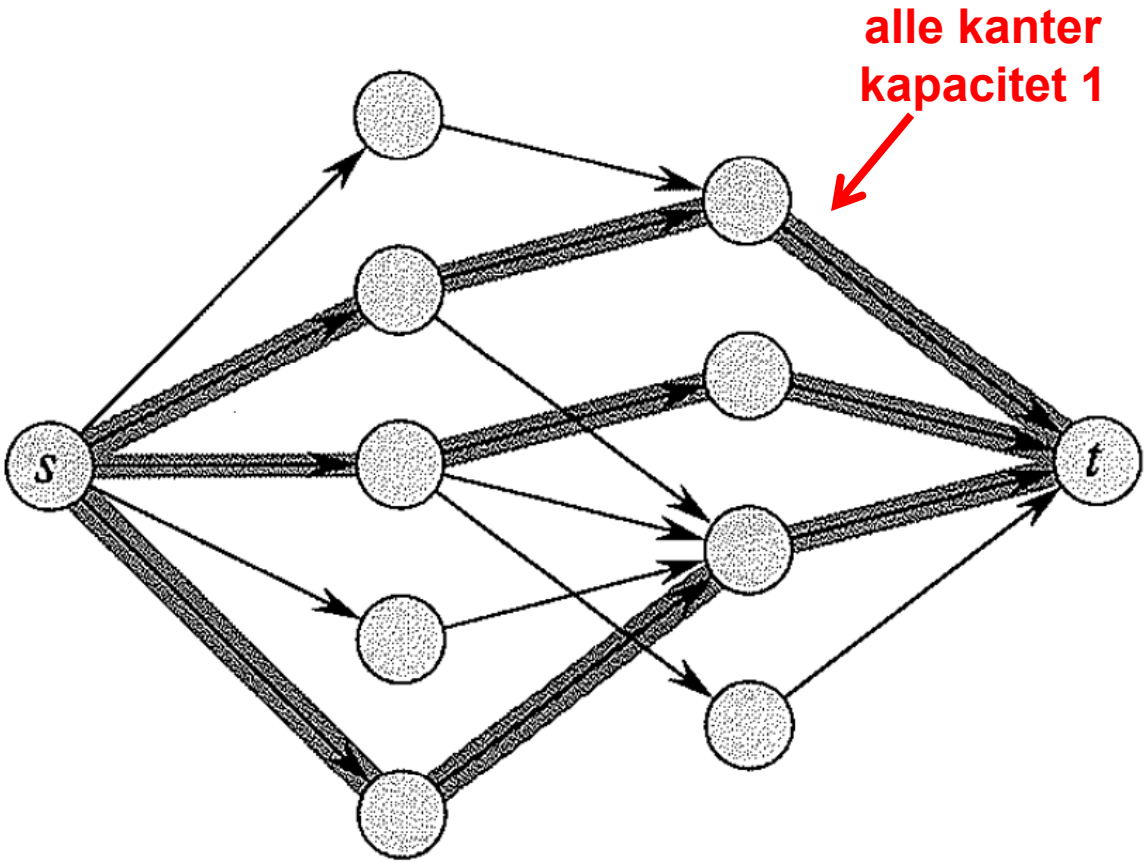
Parring af størrelse 3

Parring vs. Strømning



Todelt graf
Maximum matching

≡



Strømnings netværk
Maximum strømning
(i en heltallig løsning,
f.x. Ford-Fulkerson)

NWERC 2007

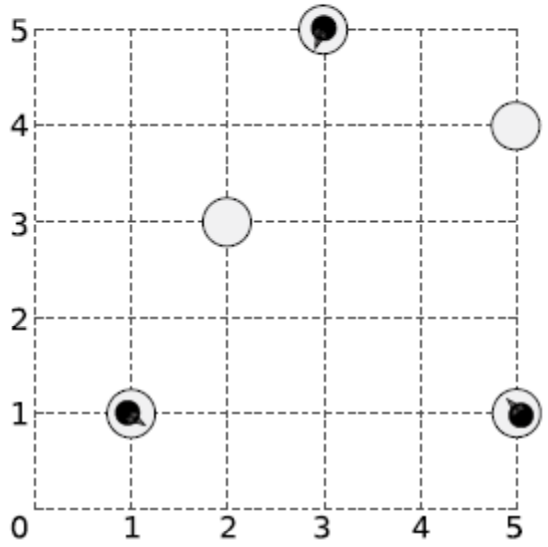
The 2007 ACM Northwestern European Programming Contest
Utrecht University, The Netherlands

Problem B: March of the Penguins

2007.nwerc.eu/problems/nwerc07-problemset.pdf

B March of the Penguins

Somewhere near the south pole, a number of penguins are standing on a number of ice floes. Being social animals, the penguins would like to get together, all on the same floe. The penguins do not want to get wet, so they have use their limited jump distance to get together by jumping from piece to piece. However, temperatures have been high lately, and the floes are showing cracks, and they get damaged further by the force needed to jump to another floe. Fortunately the penguins are real experts on cracking ice floes, and know exactly how many times a penguin can jump off each floe before it disintegrates and disappears. Landing on an ice floe does not damage it. You have to help the penguins find all floes where they can meet.



A sample layout of ice floes with 3 penguins on them.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with the integer N ($1 \leq N \leq 100$) and a floating-point number D ($0 \leq D \leq 100\,000$), denoting the number of ice pieces and the maximum distance a penguin can jump.
- N lines, each line containing x_i, y_i, n_i and m_i, denoting for each ice piece its X and Y coordinate, the number of penguins on it and the maximum number of times a penguin can jump off this piece before it disappears ($-10\,000 \leq x_i, y_i \leq 10\,000$, $0 \leq n_i \leq 10$, $1 \leq m_i \leq 200$).

Output

Per testcase:

- One line containing a space-separated list of 0-based indices of the pieces on which all penguins can meet. If no such piece exists, output a line with the single number -1.

Sample in- and output

Input	Output
2 5 3.5 1 1 1 1 2 3 0 1 3 5 1 1 5 1 1 1 5 4 0 1 3 1.1 -1 0 5 10 0 0 3 9 2 0 1 1	1 2 4 -1