Discrete Mathematical Models

Lecture 28

Kane Townsend

Semester 2, 2024

Cut capacities and flow values (PROPER)

After the previous lecture, I decided to give a more thorough section on the results involving cut capacities and flow values (consider last lectures as an introduction).

Cut capacities and flow values (PROPER)

After the previous lecture, I decided to give a more thorough section on the results involving cut capacities and flow values (consider last lectures as an introduction).

Recall that a **cut** is a set $K = E(D) \cap (S \times T)$ in D, where S is a set containing s and $T = V(D) \setminus S$ is a set containing t.

Cut capacities and flow values (PROPER)

After the previous lecture, I decided to give a more thorough section on the results involving cut capacities and flow values (consider last lectures as an introduction).

Recall that a **cut** is a set $K = E(D) \cap (S \times T)$ in D, where S is a set containing s and $T = V(D) \setminus S$ is a set containing t.

The **capacity** of the cut $K = E(D) \cap (S \times T)$ is the number

$$C(K) = \sum_{i \in S} \sum_{j \in T} C(i, j).$$

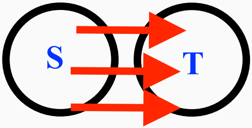
Cut capacities and flow values (PROPER)

After the previous lecture, I decided to give a more thorough section on the results involving cut capacities and flow values (consider last lectures as an introduction).

Recall that a **cut** is a set $K = E(D) \cap (S \times T)$ in D, where S is a set containing s and $T = V(D) \setminus S$ is a set containing t.

The **capacity** of the cut $K = E(D) \cap (S \times T)$ is the number

$$C(K) = \sum_{i \in S} \sum_{j \in T} C(i, j).$$



We will prove that cut capacity is at least the flow value, max flow value eq min cut capacity and the complete labelling algorithm finds max flow while giving us a minimal cut.

Cut capacity is at least the flow value

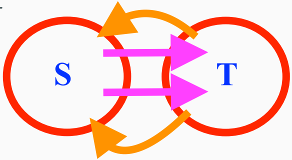
Theorem (Cut capacity is at least the flow value)
Let F be a flow and $K = E(D) \cap (S \times T)$ be a cut in D. The cut capacity of C is at least the flow value of F. That is, $C(K) \geq v(F)$.

Cut capacity is at least the flow value

Theorem (Cut capacity is at least the flow value)
Let F be a flow and $K = E(D) \cap (S \times T)$ be a cut in D. The cut capacity of C is at least the flow value of F. That is, $C(K) \geq v(F)$.

Proof.

$$v(F) = \sum_{i \in V(D)} F(s,i) = \sum_{j \in S} \sum_{i \in T} F(j,i) - \sum_{j \in S} \sum_{i \in T} F(i,j)$$
$$\leq \sum_{j \in S} \sum_{i \in T} F(j,i) \leq \sum_{j \in S} \sum_{i \in T} C(j,i) = C(K).$$



□

Explanations:

i. By definition of flow value.
ii. By conservation of flow and $S \cup T = V(D)$.
iii. By removing a negative term.
iv. Since $F(j,i) \leq C(j,i)$ for all $(j,i) \in E(D)$.
v. Definition of capacity of K.

Max flow - Min cut Theorem

Recall that a **minimal cut** is a cut having minimal capacity.

Max flow - Min cut Theorem

Recall that a **minimal cut** is a cut having minimal capacity.

Theorem (Max flow eq Min Cut Theorem)
Let F be a flow and $K = V(D) \cap (S \times T)$ a cut of D. If equality holds in the previous theorem, then the flow is maximal and the cut is minimal. This occurs if and only if

- $F(i,j) = C(i,j)$ *for all* $i \in S, j \in T$ *and*
- $F(i,j) = 0$ *for all* $i \in T, j \in S$.

Max flow - Min cut Theorem

Recall that a **minimal cut** is a cut having minimal capacity.

Theorem (Max flow eq Min Cut Theorem)
Let F be a flow and $K = V(D) \cap (S \times T)$ a cut of D. If equality holds in the previous theorem, then the flow is maximal and the cut is minimal. This occurs if and only if

- $F(i,j) = C(i,j)$ *for all* $i \in S, j \in T$ *and*
- $F(i,j) = 0$ *for all* $i \in T, j \in S$.

Proof.
The first statement follows immediately from the previous theorem.

Max flow - Min cut Theorem

Recall that a **minimal cut** is a cut having minimal capacity.

Theorem (Max flow eq Min Cut Theorem)
Let F be a flow and $K = V(D) \cap (S \times T)$ a cut of D. If equality holds in the previous theorem, then the flow is maximal and the cut is minimal. This occurs if and only if

- $F(i,j) = C(i,j)$ *for all* $i \in S, j \in T$ *and*
- $F(i,j) = 0$ *for all* $i \in T, j \in S$.

Proof.
The first statement follows immediately from the previous theorem.

Carefully looking at the proof of the previous theorem we notice that equality holds precisely when

$$\sum_{j \in S} \sum_{i \in T} F(i,j) = 0 \quad \text{and} \quad \sum_{j \in S} \sum_{i \in T} F(j,i) = \sum_{j \in S} \sum_{i \in T} C(j,i)$$

□

Finding a minimal cut

Theorem (Complete algorithm works and finds minimal cut)
The Complete Algorithm produces a maximal flow. Furthermore, if we take the cut $K = E(D) \cap (S \times T)$, where S is the labelled vertices at the termination of the Complete Algorithm and T is the unlabelled vertices at the termination of the Complete Algorithm, then K is minimal.

Finding a minimal cut

Theorem (Complete algorithm works and finds minimal cut)
The Complete Algorithm produces a maximal flow. Furthermore, if we take the cut $K = E(D) \cap (S \times T)$, where S is the labelled vertices at the termination of the Complete Algorithm and T is the unlabelled vertices at the termination of the Complete Algorithm, then K is minimal.

Proof.
For all $i \in S$ and $j \in T$, since i is labelled, we must have $F(i,j) = C(i,j)$ (otherwise we would have labelled further).

Finding a minimal cut

Theorem (Complete algorithm works and finds minimal cut)
The Complete Algorithm produces a maximal flow. Furthermore, if we take the cut $K = E(D) \cap (S \times T)$, where S is the labelled vertices at the termination of the Complete Algorithm and T is the unlabelled vertices at the termination of the Complete Algorithm, then K is minimal.

Proof.
For all $i \in S$ and $j \in T$, since i is labelled, we must have $F(i,j) = C(i,j)$ (otherwise we would have labelled further). Now consider an edge (j, i) with $j \in T$ and $i = S$. Since i is labelled, we must have $F(j, i) = 0$ (otherwise we would have labelled further for virtual flow).

Finding a minimal cut

Theorem (Complete algorithm works and finds minimal cut)
The Complete Algorithm produces a maximal flow. Furthermore, if we take the cut $K = E(D) \cap (S \times T)$, where S is the labelled vertices at the termination of the Complete Algorithm and T is the unlabelled vertices at the termination of the Complete Algorithm, then K is minimal.

Proof.
For all $i \in S$ and $j \in T$, since i is labelled, we must have $F(i,j) = C(i,j)$ (otherwise we would have labelled further). Now consider an edge (j,i) with $j \in T$ and $i = S$. Since i is labelled, we must have $F(j,i) = 0$ (otherwise we would have labelled further for virtual flow).

By max flow eq min flow theorem our flow is maximal and the cut $K = E(D) \cap (S \times T)$ is minimal. $\square$

Finding a minimal cut

Theorem (Complete algorithm works and finds minimal cut)
The Complete Algorithm produces a maximal flow. Furthermore, if we take the cut $K = E(D) \cap (S \times T)$, where S is the labelled vertices at the termination of the Complete Algorithm and T is the unlabelled vertices at the termination of the Complete Algorithm, then K is minimal.

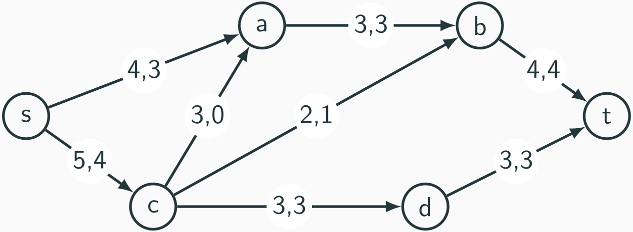
Proof.
For all $i \in S$ and $j \in T$, since i is labelled, we must have $F(i,j) = C(i,j)$ (otherwise we would have labelled further). Now consider an edge (j,i) with $j \in T$ and $i = S$. Since i is labelled, we must have $F(j,i) = 0$ (otherwise we would have labelled further for virtual flow).

By max flow eq min flow theorem our flow is maximal and the cut $K = E(D) \cap (S \times T)$ is minimal. □

Go check all the previous examples that you can find the minimal cut by running the algorithm and applying this theorem!

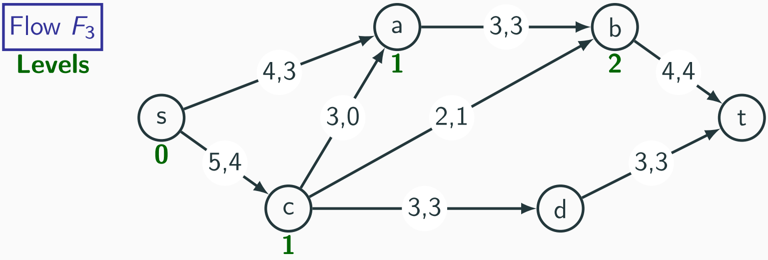
Example 2: Finding min cut via Complete Labelling Algorithm

Flow F_3



We take $S = \{s, a, b, c\}$ and $T = \{d, t\}$, so the minimal cut $K = E(D) \cap (S \times T) = \{(c, d), (b, t)\}$.

Example 2: Finding min cut via Complete Labelling Algorithm



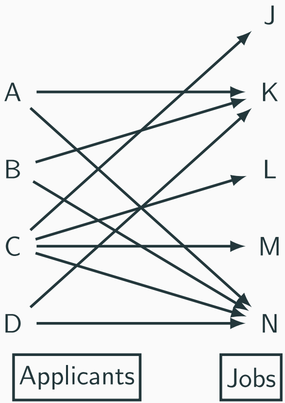
No level can be assigned to d or t !

We take $S = \{s, a, b, c\}$ and $T = \{d, t\}$, so the minimal cut $K = E(D) \cap (S \times T) = \{(c, d), (b, t)\}$.

Matching

A matching problem

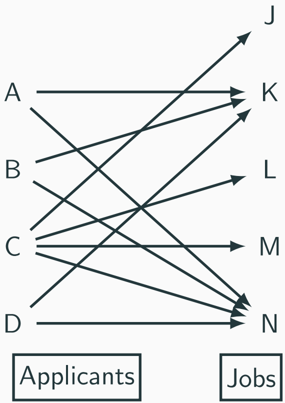
Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?



A matching problem

Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?

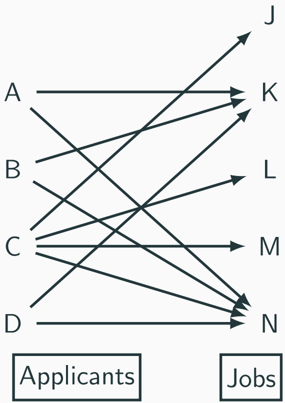
That is, can each applicant x be *matched* with a job $m(x)$?



A matching problem

Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?

That is, can each applicant x be *matched* with a job $m(x)$?   If so, how?

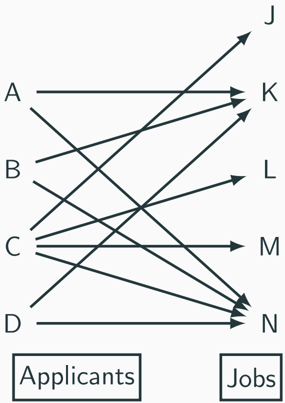


A matching problem

Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?

That is, can each applicant x be *matched* with a job $m(x)$? If so, how?

If not, what's the best that can be done?



A matching problem

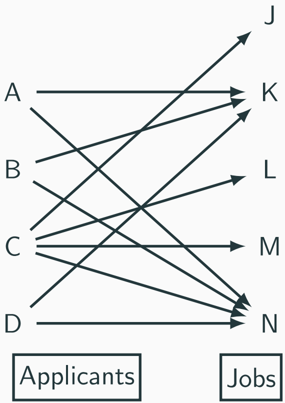
Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?

That is, can each applicant x be *matched* with a job $m(x)$?   If so, how?

If not, what's the best that can be done?

One answer:

x	A	B	C	D
$m(x)$	K	N	J	-



A matching problem

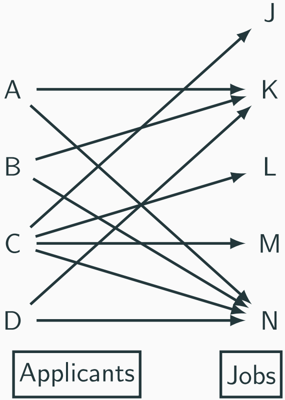
Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?

That is, can each applicant x be *matched* with a job $m(x)$? If so, how?

If not, what's the best that can be done?

One answer:

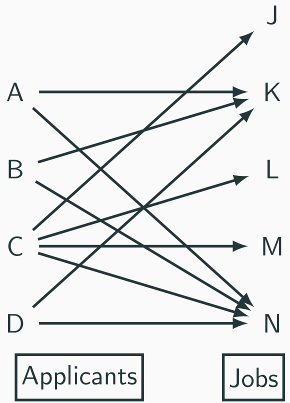
x	A	B	C	D
$m(x)$	K	N	J	-



More generally, given a relation $R \subseteq S \times T$ a **matching problem** seeks a **maximal matching function** (or just a 'matching') $m \subseteq R$.

A matching problem

Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?



That is, can each applicant x be *matched* with a job $m(x)$?   If so, how?

If not, what's the best that can be done?

One answer:

x	A	B	C	D
$m(x)$	K	N	J	-

More generally, given a relation $R \subseteq S \times T$ a **matching problem** seeks a **maximal matching function** (or just a 'matching') $m \subseteq R$.
This is a **injective** (one-to-one) function $f : S' \to T$ with domain $S' \subseteq S$ as large as possible subject to m being an injective subset of R.

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved ‘by eye’. Larger matching problems require a more systematic method.

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method.
The vertex labelling algorithm for transport networks can be used!

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method.
The vertex labelling algorithm for transport networks can be used!

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method. The vertex labelling algorithm for transport networks can be used!

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

1. Create a digraph D by adding to the arrow diagram for R

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method. The vertex labelling algorithm for transport networks can be used!

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

1. Create a digraph D by adding to the arrow diagram for R
   - a **supersource** s and edges (s,x) for each $x \in S$ and

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method. The vertex labelling algorithm for transport networks can be used!

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

1. Create a digraph D by adding to the arrow diagram for R
    - a **supersource** s and edges (s,x) for each $x \in S$ and
    - a **supertarget** t and edges (y,t) for each $y \in T$.

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method.
The vertex labelling algorithm for transport networks can be used!

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

1. Create a digraph D by adding to the arrow diagram for R
   - a **supersource** s and edges (s,x) for each $x \in S$ and
   - a **supertarget** t and edges (y,t) for each $y \in T$.
2. Assign capacity 1 to every edge of D. (*i.e.* $\forall e \in E(D)\ C(e) = 1$.)

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved 'by eye'. Larger matching problems require a more systematic method. The vertex labelling algorithm for transport networks can be used!

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

1. Create a digraph D by adding to the arrow diagram for R
   - a **supersource** s and edges (s,x) for each $x \in S$ and
   - a **supertarget** t and edges (y,t) for each $y \in T$.
2. Assign capacity 1 to every edge of D. (*i.e.* $\forall e \in E(D)\ C(e) = 1$.)

A solution to the max flow problem provides the matching:

$$\boxed{m = \{(x, y) \in S \times T \ :\ F_{\max}((x, y)) = 1\}.}$$

Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem.

Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.

Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.

Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

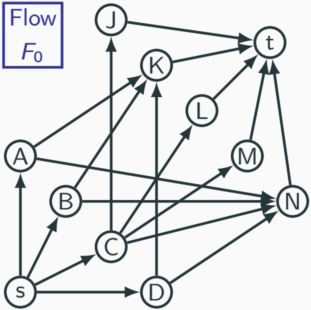
- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

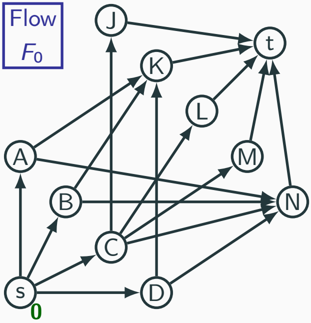


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

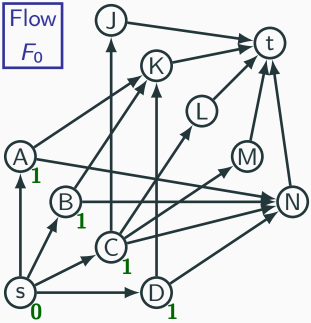


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

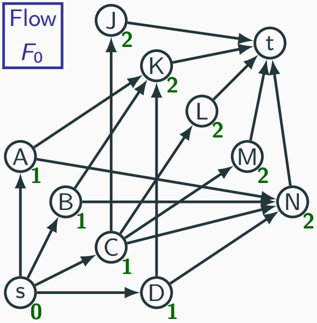


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

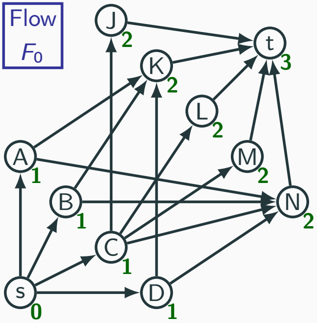


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

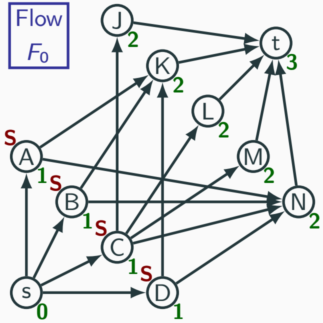


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

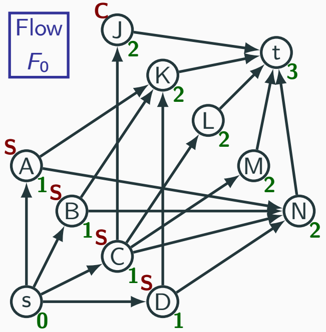


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

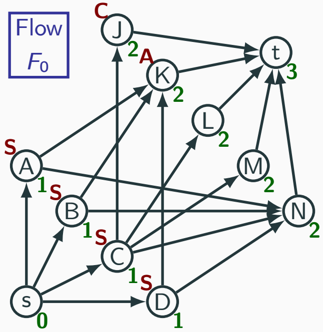


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

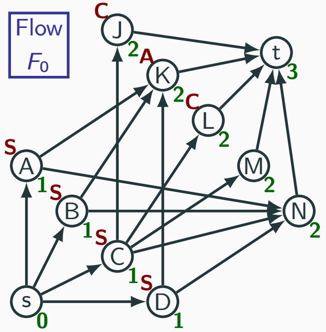


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

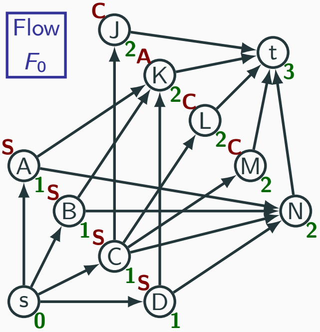


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

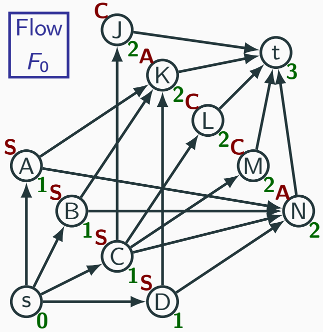


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.

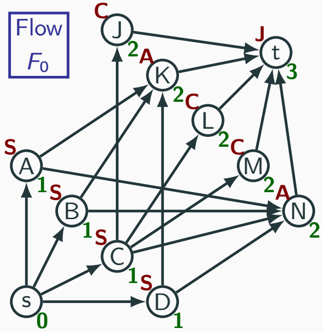


Vertex labelling for matching; Example 1

Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1.

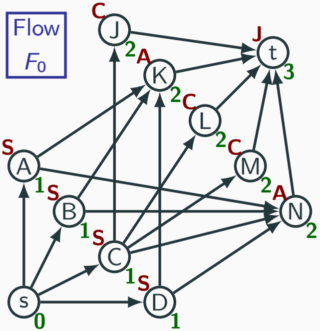
- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.



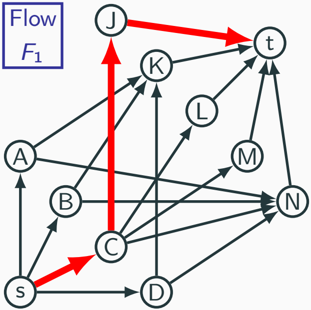
Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

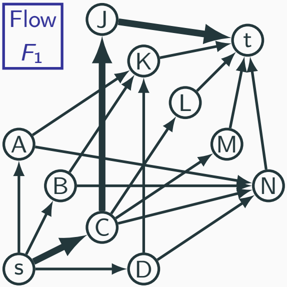
With all edge capacities 1, edge flows are either 0 or 1.

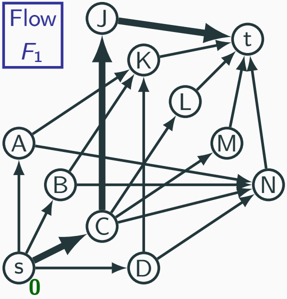
- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.



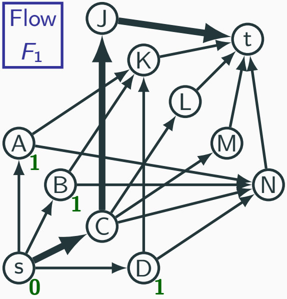
⇒




Flow
F_1
J
t
K
L
A
M
B
N
C
s
D

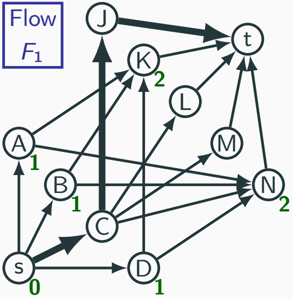

Flow
F_1
J
t
K
L
A
M
B
N
C
s
D
0

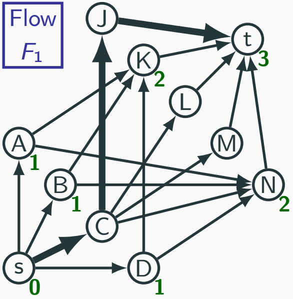
Flow
F_1



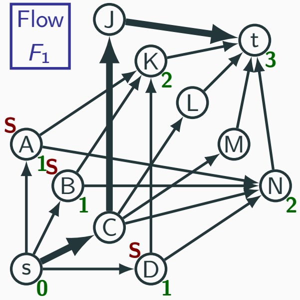
Flow

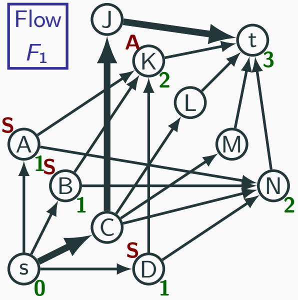
F_1

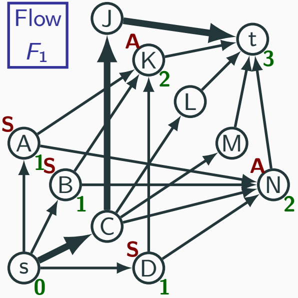




Flow
F_1
J
t
3
K
2
L
A
1
M
B
1
N
2
C
s
0
D
1

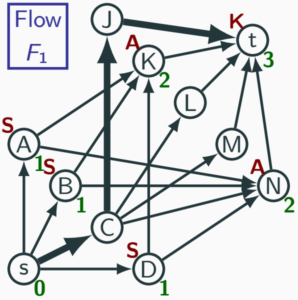

Flow
F_1
J
t
K
L
M
N
A
B
C
D
s
S
S
S
0
1
1
1
2
2
3

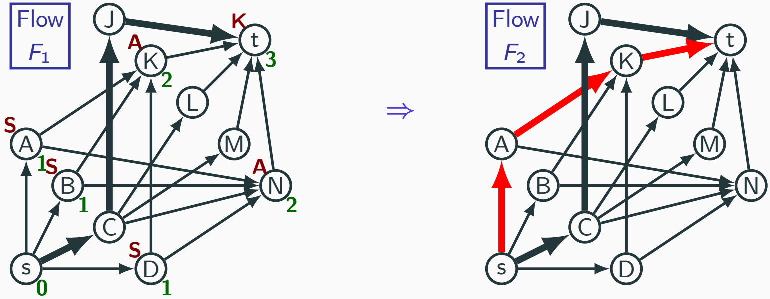

Flow
F_1
J
t
A
K
2
3
L
S
A
M
1
S
B
N
1
2
C
S
s
D
0
1



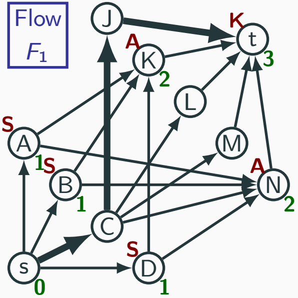
Flow
F_1
J
t
K
L
M
A
B
C
N
s
D
S
A
S
A
S
0
1
1
1
2
2
3

Flow F_1

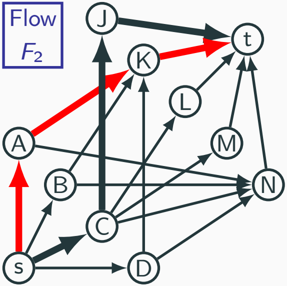


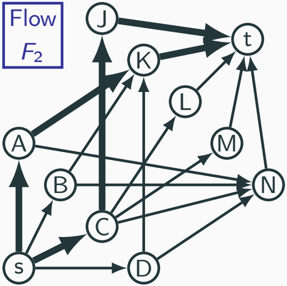


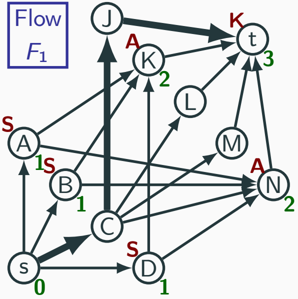
Flow
F_1
J
t
K
A
K
2
3
L
S
A
M
1
S
B
1
A
N
2
C
S
s
0
D
1
⇒
Flow
F_2
J
t
K
L
A
M
B
N
C
s
D


Flow
F_1
J
t
K
A
S
B
C
D
L
M
N
s
0
1
2
3

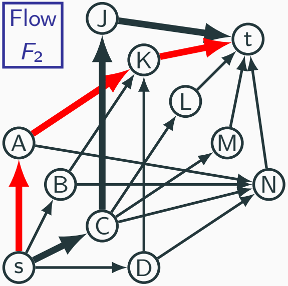
⇒

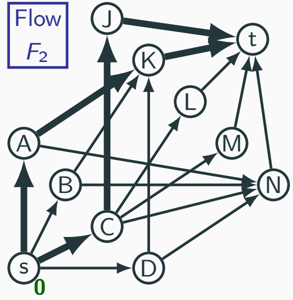

Flow
F_2
J
t
K
A
B
C
D
L
M
N
s

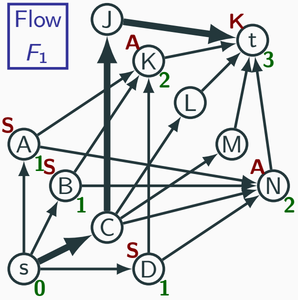

Flow
F_2
J
t
K
A
B
C
D
L
M
N
s


Flow F_1
J
K
t
3
A
K
2
L
S
A
M
1
S
A
B
N
1
2
C
S
s
D
0
1

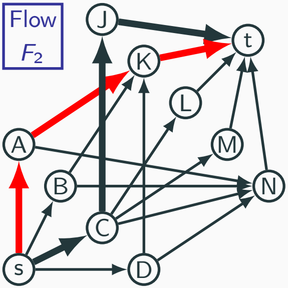
⇒

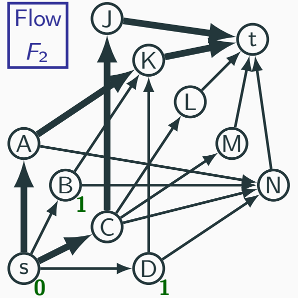

Flow F_2
J
t
K
L
A
M
B
N
C
s
D

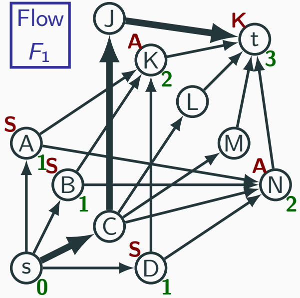

Flow F_2
J
t
K
L
A
M
B
N
C
s
D
0


Flow
F_1
J
K
t
A
K
2
3
L
S
A
M
1
S
A
B
N
1
2
C
S
s
D
0
1

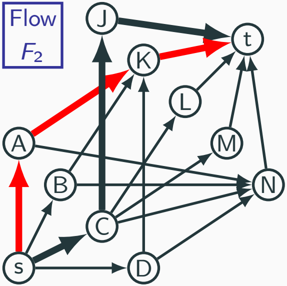
⇒

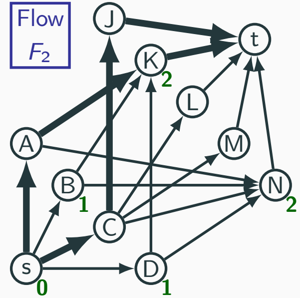

Flow
F_2
J
t
K
L
A
M
B
N
C
s
D

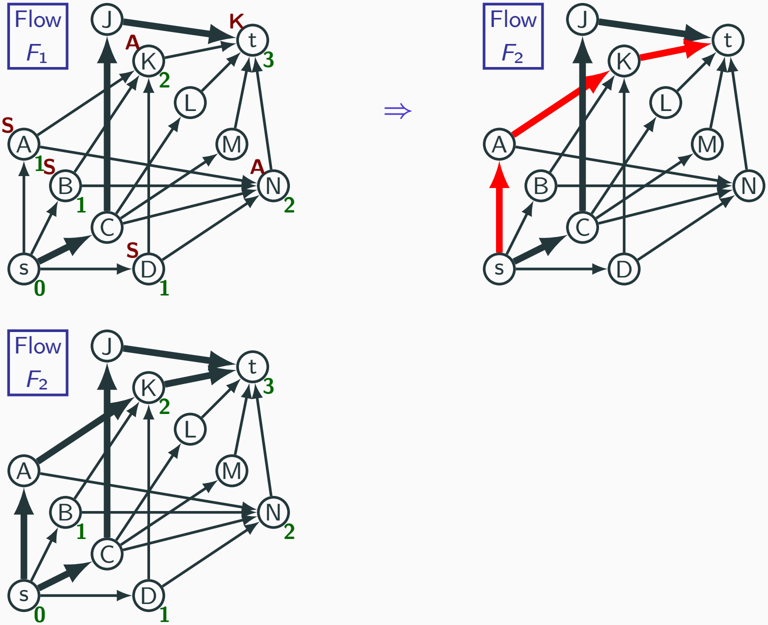

Flow
F_2
J
t
K
L
A
M
B
N
1
C
s
D
0
1


Flow F_1
J
K
t
A
K
3
2
L
S
A
M
1
S
B
A
N
1
2
C
S
s
D
0
1

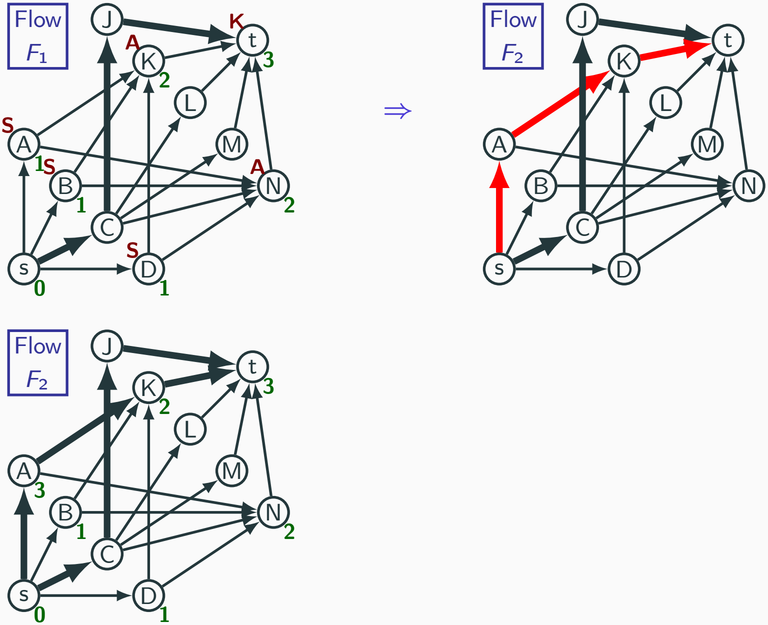
⇒

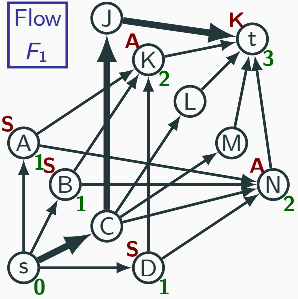

Flow F_2
J
t
K
L
A
M
B
N
C
s
D


Flow F_2
J
t
K
2
L
A
M
B
N
1
2
C
s
D
0
1

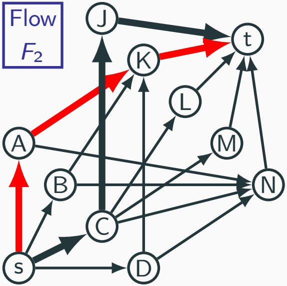


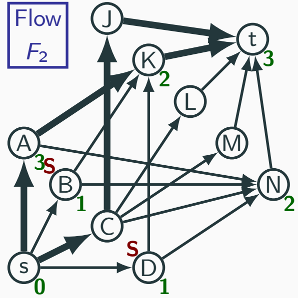
Flow F_1
J t K A L M S N S B C D s 0 1 1 1 2 2 3
⇒
Flow F_2
J t K L A M B N C s D
Flow F_2
J t K L A M B N C s D 0 1 1 2 2 3

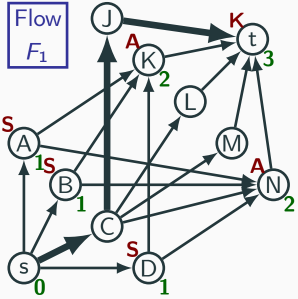

Flow F_1
⇒
Flow F_2
Flow F_2
s
A
B
C
D
J
K
L
M
N
t


Flow
F_1
J
t
K
A
L
M
N
B
C
D
s
S
S
A
S
K
A
0
1
1
1
2
2
3

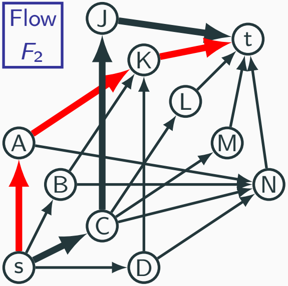
⇒

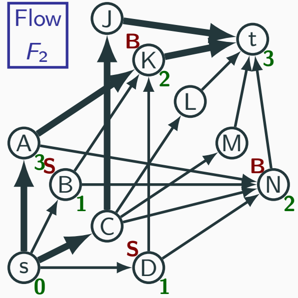

Flow
F_2
J
t
K
A
L
M
N
B
C
D
s

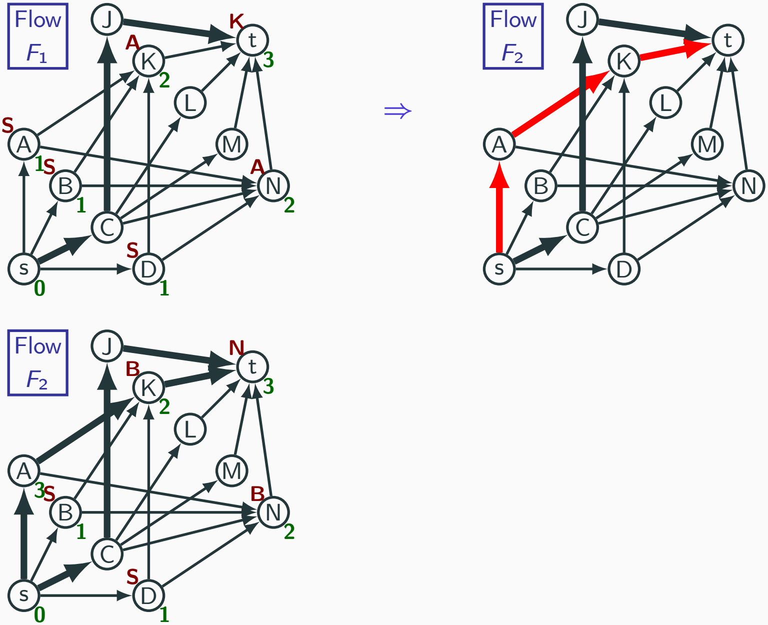

Flow
F_2
J
t
K
A
L
M
N
B
C
D
s
S
S
0
1
1
3
2
2
3

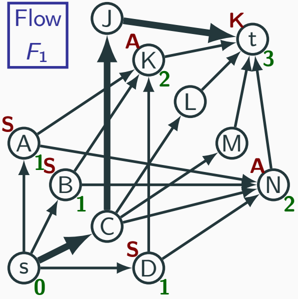

Flow
F_1
J
K
t
3
A
K
2
L
S
A
M
1
S
A
B
N
1
2
C
S
s
D
0
1

$\Rightarrow$

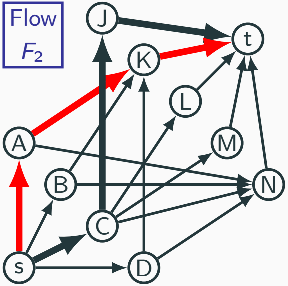

Flow
F_2
J
t
K
L
A
M
B
N
C
s
D

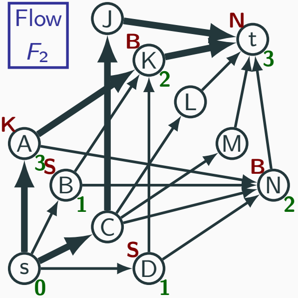

Flow
F_2
J
B
t
K
3
2
L
A
M
3
S
B
B
N
1
2
C
S
s
D
0
1

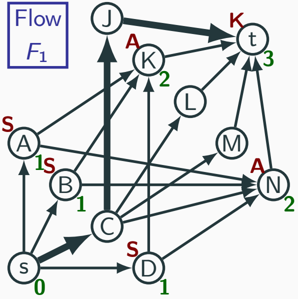

Flow F_1
⇒
Flow F_2
Flow F_2


Flow
F_1
J t K L M A B C D N s
K A S S A S
3 2 1 1 2 0 1

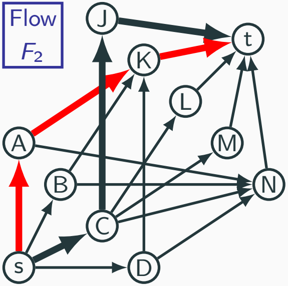
⇒

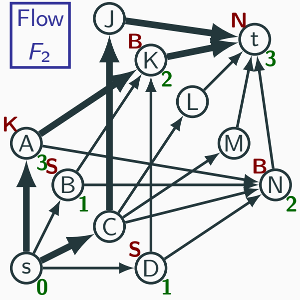

Flow
F_2
J t K L M A B C D N s


Flow
F_2
J t K L M A B C D N s
N B K S B S
3 2 3 1 2 0 1

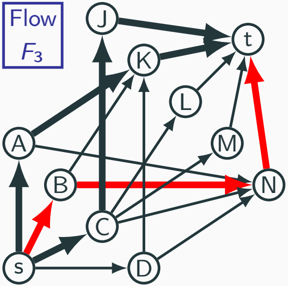

Flow
F_1
J
K
t
A
L
M
N
B
C
s
D
S
0
1
2
3

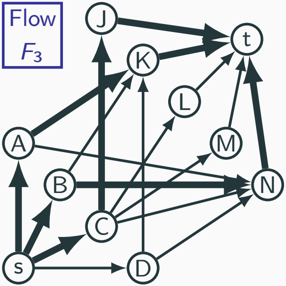
⇒

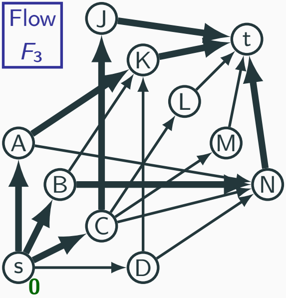

Flow
F_2
J
K
t
A
L
M
N
B
C
s
D

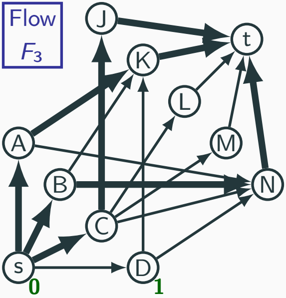

Flow
F_2
J
K
t
A
L
M
N
B
C
s
D
S
0
1
2
3

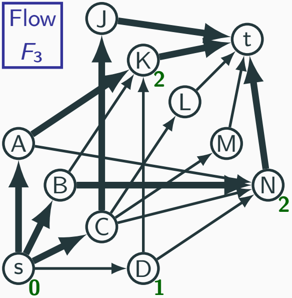
⇒


Flow
F_3
J
K
t
A
L
M
N
B
C
s
D

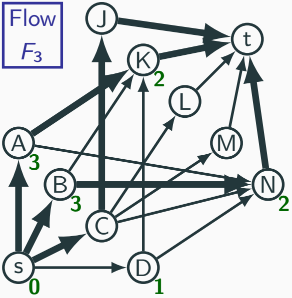

Flow
F_3
J
t
K
L
M
A
B
N
C
s
D

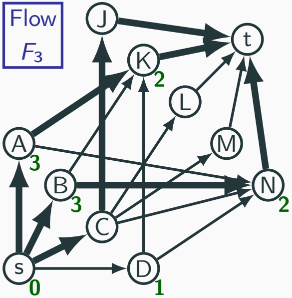

Flow
F_3
J
t
K
L
M
A
B
N
C
s
D
0


Flow
F_3
J
t
K
L
M
A
B
N
C
s
D
0
1



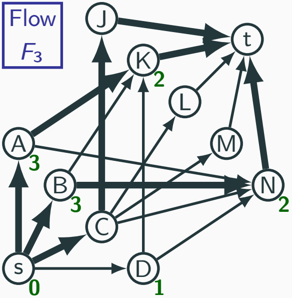
Flow
F_3
J
t
K
2
L
A
M
B
N
2
C
s
0
D
1


Flow
F_3
J
t
K
2
L
A
3
M
B
3
N
2
C
s
0
D
1



$\Rightarrow$

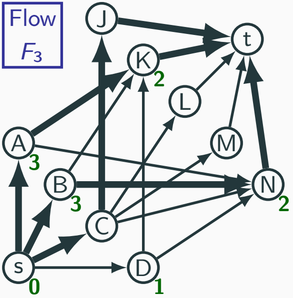
Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t, so the algorithm terminates.



$\Rightarrow$

Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t, so the algorithm terminates.

The matchings are indicated by the edges between $\{A, B, C, D\}$ and $\{J, K, L, M, N\}$ that have flow 1 (the thick edges).



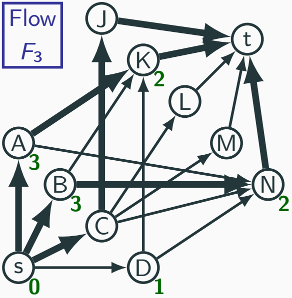
$\Rightarrow$

Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t, so the algorithm terminates.

The matchings are indicated by the edges between $\{A, B, C, D\}$ and $\{J, K, L, M, N\}$ that have flow 1 (the thick edges).

So the matching is

$$m = \{(A, K), (B, N), (C, J)\}.$$



$\Rightarrow$

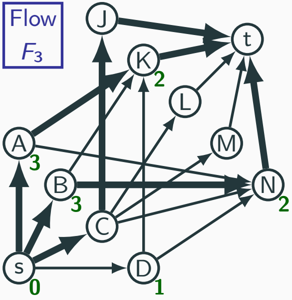
Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t, so the algorithm terminates.

The matchings are indicated by the edges between $\{A, B, C, D\}$ and $\{J, K, L, M, N\}$ that have flow 1 (the thick edges).

So the matching is

$m = \{(A, K), (B, N), (C, J)\}$.

As remarked earlier, the above solution is very easy to get without the use of the vertex labelling algorithm. We have just matched the first member of T with the first 'available' member of S, then the next member of T with the next 'available' member of S and so on.



$\Rightarrow$

Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t, so the algorithm terminates.

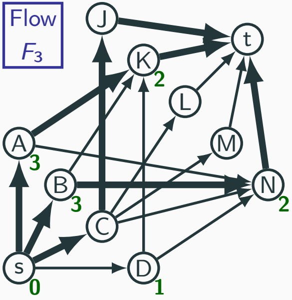
The matchings are indicated by the edges between $\{A, B, C, D\}$ and $\{J, K, L, M, N\}$ that have flow 1 (the thick edges).

So the matching is

$m = \{(A, K), (B, N), (C, J)\}$.

As remarked earlier, the above solution is very easy to get without the use of the vertex labelling algorithm. We have just matched the first member of T with the first 'available' member of S, then the next member of T with the next 'available' member of S and so on.

The benefit of using the algorithm lies in its ability to 'undo' initial pairings and replace them by new ones when this becomes necessary to enable later pairings.



$\Rightarrow$

Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t, so the algorithm terminates.

The matchings are indicated by the edges between $\{A, B, C, D\}$ and $\{J, K, L, M, N\}$ that have flow 1 (the thick edges).

So the matching is

$m = \{(A, K), (B, N), (C, J)\}$.

As remarked earlier, the above solution is very easy to get without the use of the vertex labelling algorithm. We have just matched the first member of T with the first 'available' member of S, then the next member of T with the next 'available' member of S and so on.

The benefit of using the algorithm lies in its ability to 'undo' initial pairings and replace them by new ones when this becomes necessary to enable later pairings.

The final example shows how it does this in the simplest possible case.

Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$,

Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p.

Vertex labelling for matching; Example 2

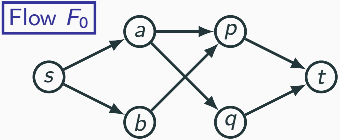
Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead.

Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

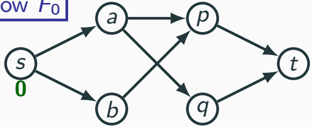


Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

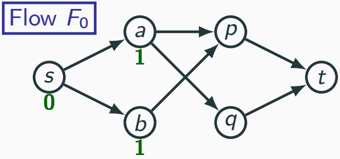
Flow F_0



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

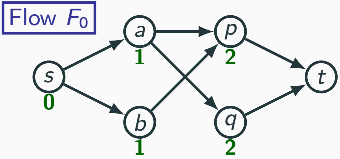
Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

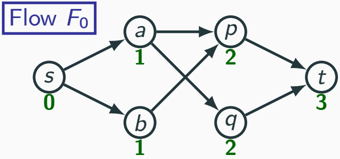
Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

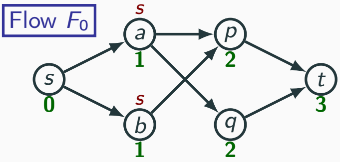
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

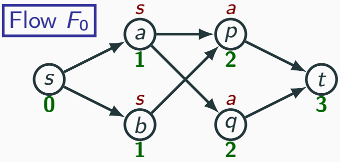
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

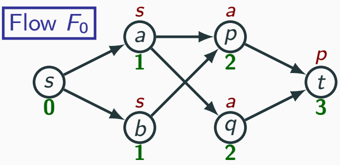
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

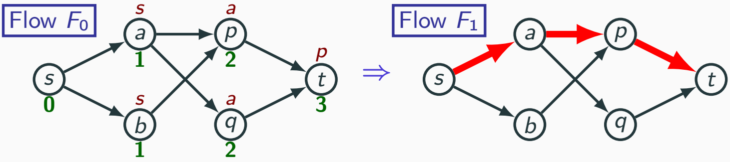
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

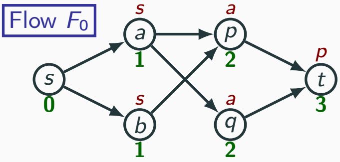
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



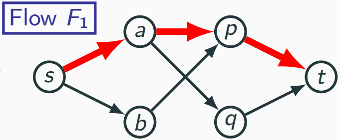
Vertex labelling for matching; Example 2

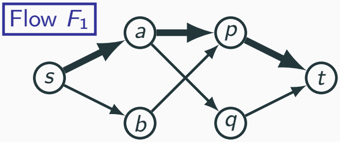
Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



$\Rightarrow$

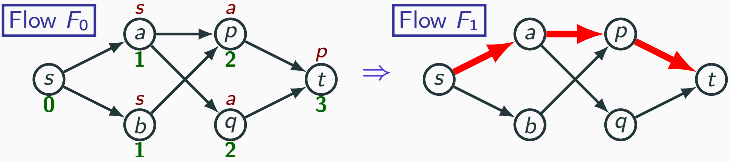


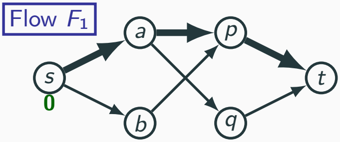


Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

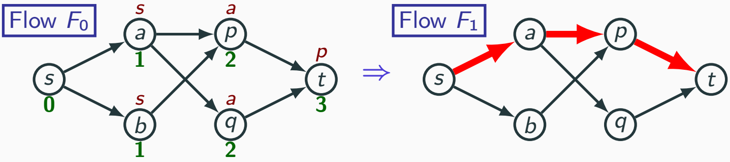


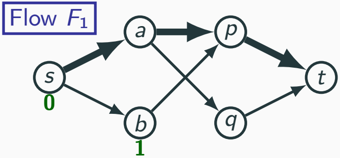


Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

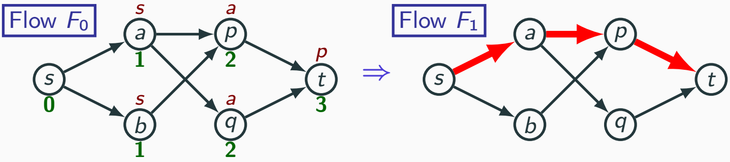


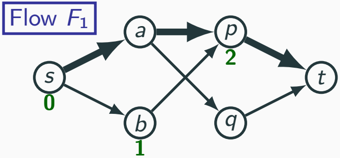


Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



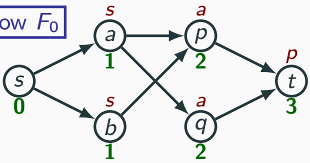


Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

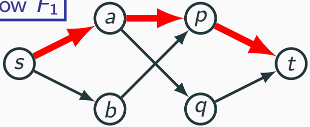
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

Flow F_0

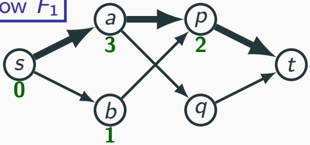


$\Rightarrow$

Flow F_1



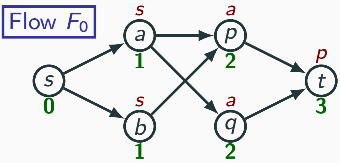
Flow F_1



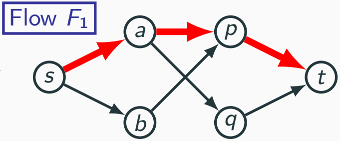
Vertex labelling for matching; Example 2

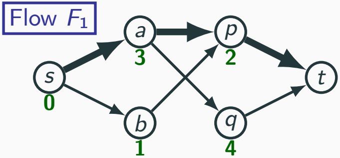
Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



$\Rightarrow$

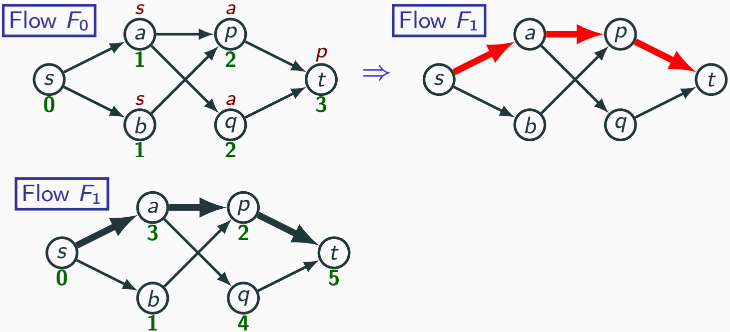




Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

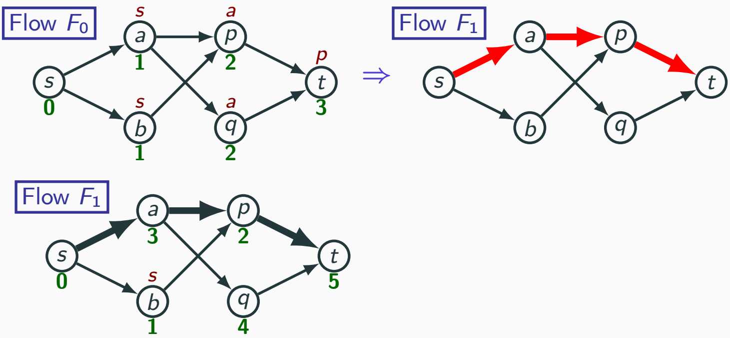
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

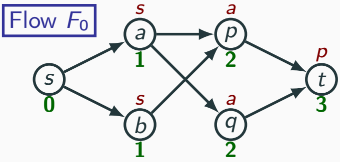
Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



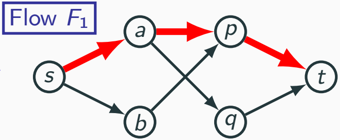
Vertex labelling for matching; Example 2

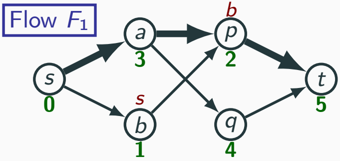
Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



$\Rightarrow$

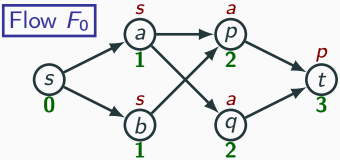




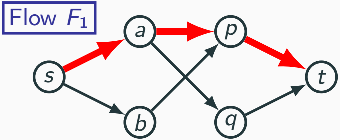
Vertex labelling for matching; Example 2

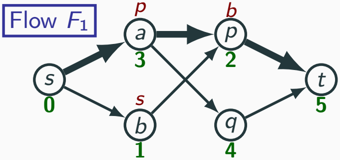
Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



$\Rightarrow$

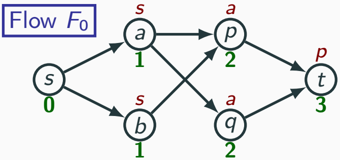




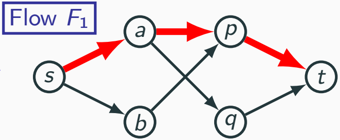
Vertex labelling for matching; Example 2

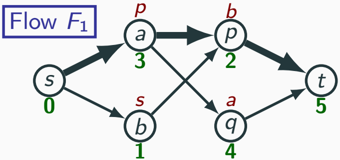
Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:



$\Rightarrow$

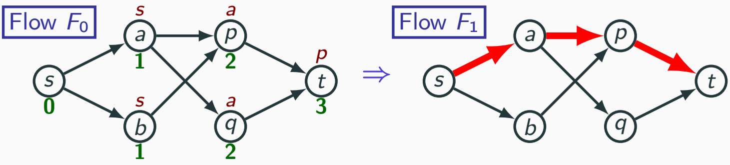


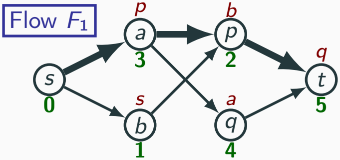


Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

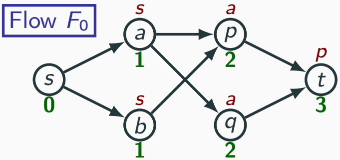




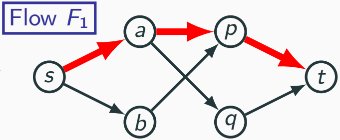
Vertex labelling for matching; Example 2

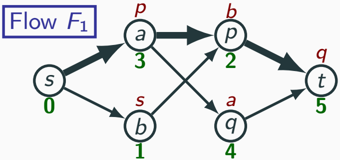
Find a (maximal) matching function m
for the relation $R = \{(a,p),(a,q),(b,p)\}$.

Obviously the only answer is $m = \{(a,q),(b,p)\}$, but the vertex labelling algorithm will first match a with p. However it will then use a virtual flow to undo this and match a with q, allowing b to be matched with p instead. Here are the diagrams:

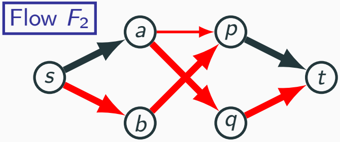


$\Rightarrow$





$\Rightarrow$



End of Weighted directed graphs

END OF SECTION D2

End of Weighted directed graphs

END OF SECTION D2

Next lecture we begin D3: Random walks on graphs and the Google Page Rank Algorithm. This will use our understanding of Markov processes.

END OF SECTION D2

Next lecture we begin D3: Random walks on graphs and the Google Page Rank Algorithm. This will use our understanding of Markov processes.

Next week the Quiz 9 (Week 11) will be on 'D1: Introduction to Graph Theory' (Lectures 22,23,24).