Barrington's Theorem

Malek Alsalamat

Universität Kassel

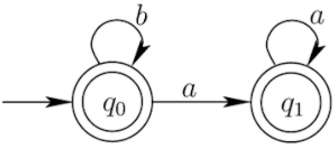
February 17, 2021

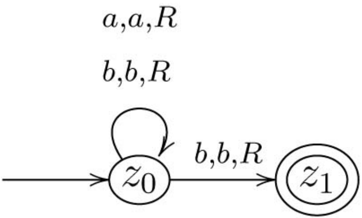
Inhalt der Präsentation

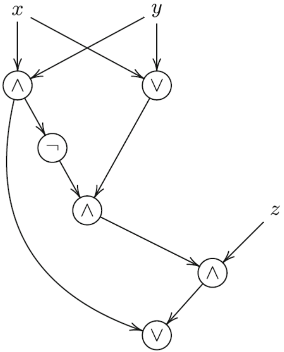
In der Präsentation werden wir folgends behandeln:

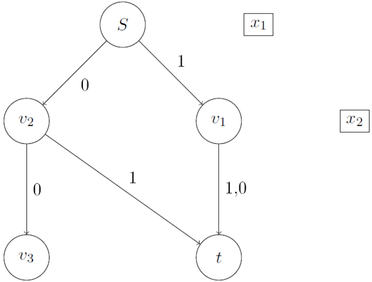
1. Arten von Berechnungsmodelle
2. Branching-Programme
3. Zyklische Permutation
4. Permutation Branching-Programme
5. Beweis von Barrington Theorem

Berechnungsmodelle









Branching-Programme

Definition von Branching-Programm

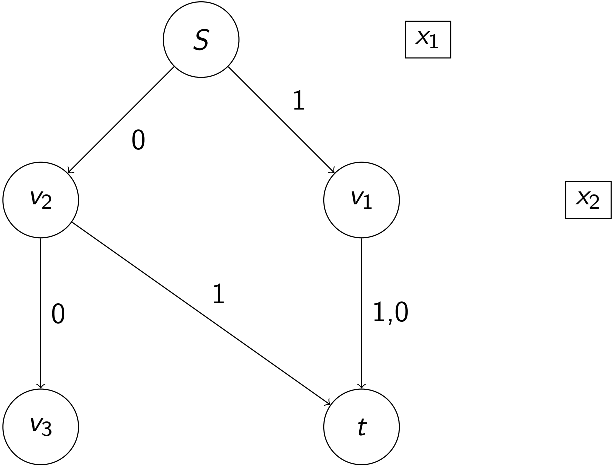
Ein n-input Branching-Programm ist ein Tupel $P = (V, E_0, E_1, \vartheta_0, t)$, wobei:

- (V, E) ist eine endliche gerichtete Graph und jeder Knote hat **fanout** 0 or 2. Wobei E ist $E_0 \cup E_1$.
- E_0 : ist die Mengen alle Kanten, die mit 0 beschriftet sind.
- E_1 : ist die Mengen alle Kanten, die mit 1 beschriftet sind.
- $\vartheta_0 \in V$ ist der *Startknote*.
- t ist der akzeptierende Knote (*Endknote*). In folgenden wird der Startknote mit S bezeichnet.

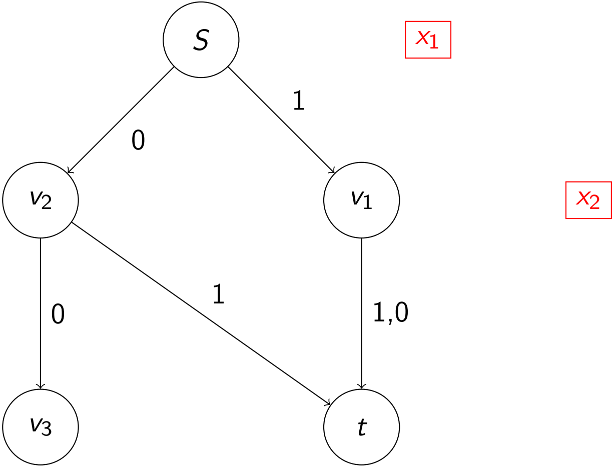
Die berechnete Funktion von P ist f_p: $\{0, 1\}^n \rightarrow \{0, 1\}$.

Branching-Programme

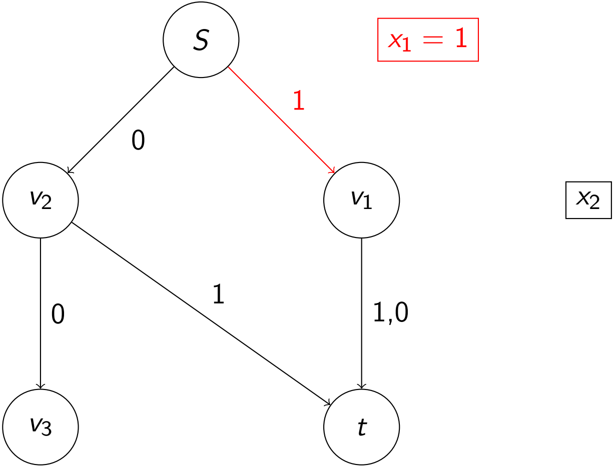
Beipiel 1: 2-input Branching-Programm



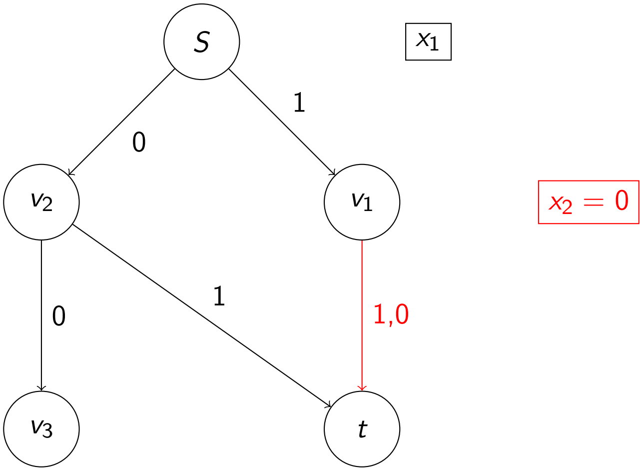
Branching-Programme



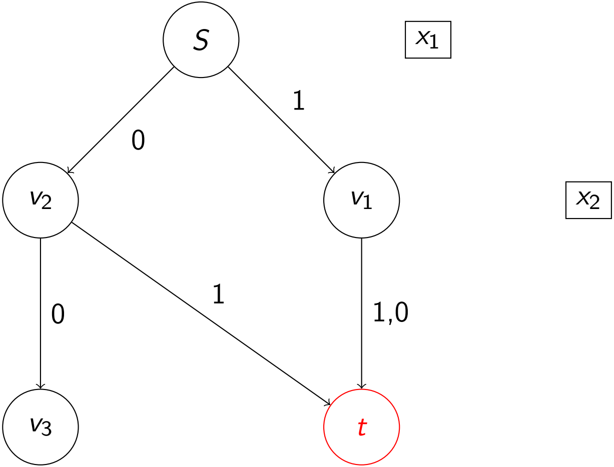
Branching-Programme



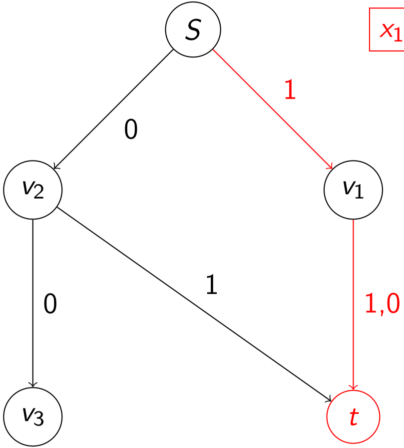
Branching-Programme



Branching-Programme



Branching-Programme

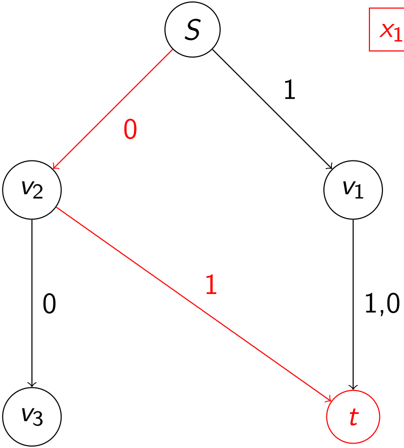


$\boxed{x_1 = 1}$

$\boxed{x_2 = 0}$

$\boxed{input = (1, 0)}$ $\boxed{output = 1}$

Branching-Programme

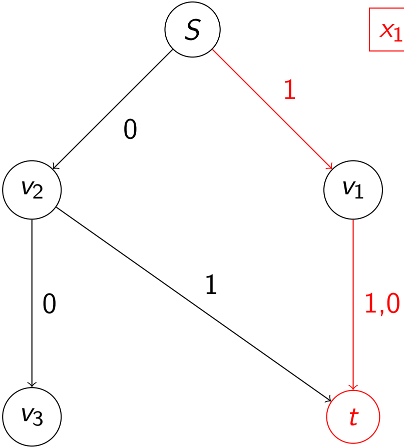


$\boxed{x_1 = 0}$

$\boxed{x_2 = 1}$

$\boxed{input = (0, 1)}$ $\boxed{output = 1}$

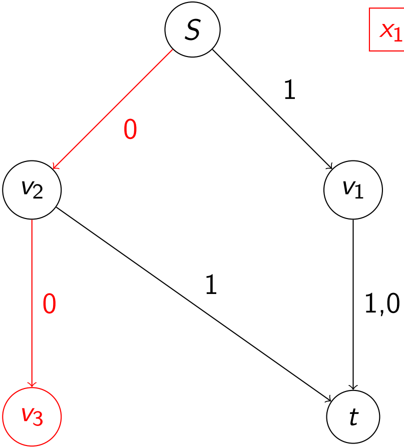
Branching-Programme



$\boxed{x_1 = 1}$

$\boxed{x_2 = 1}$

$\boxed{input = (1, 1)}$ $\boxed{output = 1}$



$\boxed{x_1 = 0}$

$\boxed{x_2 = 0}$

$\boxed{input = (0, 0)}$ $\boxed{output = 0}$

Branching-Programme

x_1	x_2	*Ausgabe*
0	0	0
0	1	1
1	0	1
1	1	1

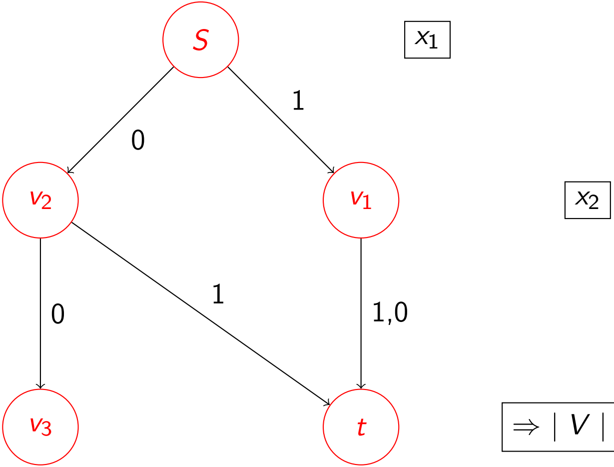
x_1	x_2	*Ausgabe*
0	0	0
0	1	1
1	0	1
1	1	1

$\Rightarrow$ das Programm berechnet die *Funktion* $f(x_1, x_2) = x_1 \vee x_2$.

Branching-Programme

Jedes Branching-Programm besitzt :

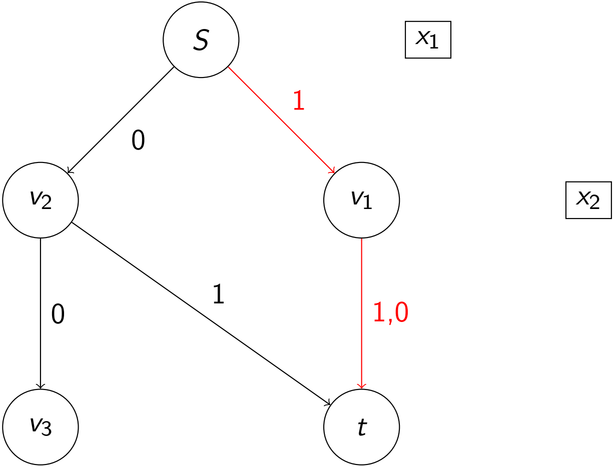
- Größe: Die Größe von einem Branching-Programm ist die Anzahl der Knoten in V und wird mit $|V|$.



$\boxed{\Rightarrow |\ V\ | = 5}$

Jedes Branching-Programm besitzt :

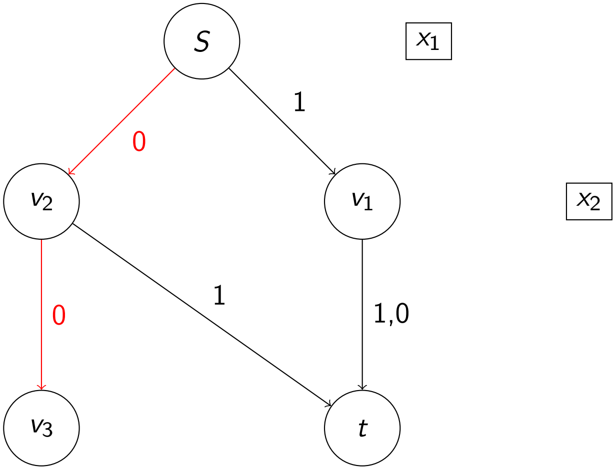
- Tiefe: Die Tiefe oder Länge von einem Branching-Programm ist der längste Pfad in unserem Graph (V, E).



Branching-Programme

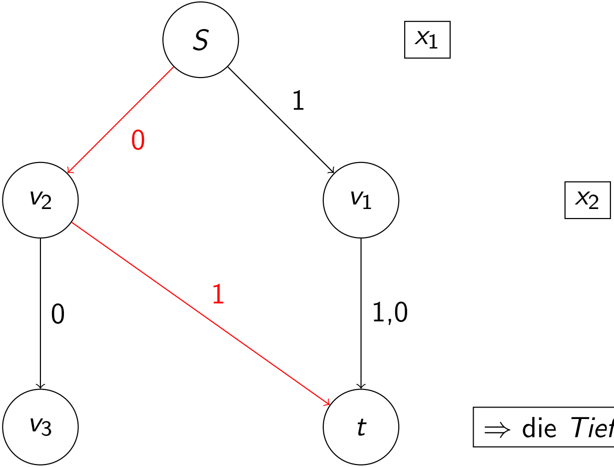
Jedes Branching-Programm besitzt :

- Tiefe: Die Tiefe oder Länge von einem Branching-Programm ist der längste Pfad in unserem Graph (V, E).



Jedes Branching-Programm besitzt :

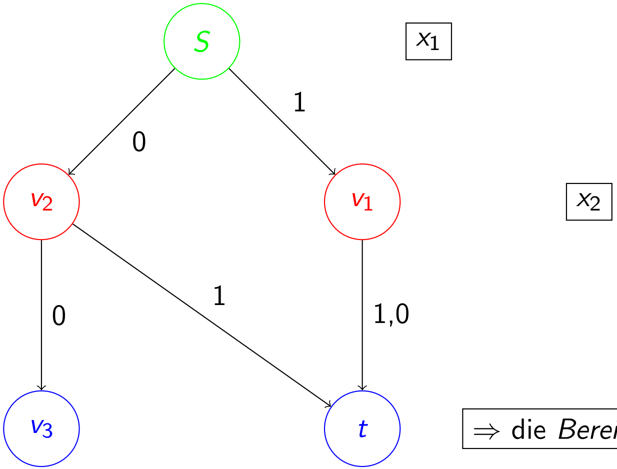
- Tiefe: Die Tiefe oder Länge von einem Branching-Programm ist der längste Pfad in unserem Graph (V, E).



$\Rightarrow$ die *Tiefe* ist 2

Jedes Branching-Programm besitzt :

- Breite: Die Breite von einem Branching-Programm ist die maximale Anzahl von Knoten in einem Level.



$\Rightarrow$ die *Bereite* ist 2

$$Majority(x) = \begin{cases} 1, & \sum_{i=0}^{N} x_i \geq n/2 \\ 0, & sonst \end{cases}$$

NC $= \bigcup_{i=0}^{\infty} NC^i$. Für alle $i \in \mathbb{N}$ ist NC^i die Klasse aller Sprachen, die von einer Schaltkreisfamilie mit polynomieller Größe, Tiefe $\mathcal{O}(log^i(n))$ und einen Fan-In von höchstens 2 erkannt werden.

In NC^1 liegen beispielsweise die Addition und Multiplikation, sowie die Majority-Funktion.

Im Jahr 1986 hat Barrington gezeigt, dass $NC^1 = $ 5-BP. Und dadurch gezeigt, dass die Majority-Funktion durch solche BP berechnet werden können.

Permutationen

Eine Permutation von $\{1, \ldots, n\}$ ist eine bijektive Abbildung $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$, $i \to \sigma(i)$.

Zyklische Schreibweiße $= (i_1, \sigma(i_1), \sigma(\sigma(i_1)), \ldots)$.

Eine Permutation heißt zyklisch wenn :

$$\sigma(i_j) = \begin{cases} i_1 & j = n \\ i_{j+1} & \text{sonst} \end{cases}$$

Beispiele:

$$a_1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$

$$a_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$$

$a_1 = (1, 2, 3)$, $a_2 = (1, 3, 2)(4, 5) \Rightarrow a_1$ ist zyklisch, aber a_2 nicht.

Permutation Branching-Programm

Ein w-Permutation Branching-Programm ist ein BP mit folgende Eigenschaften:

- Jedes Level hat genau w Knoten und somit ist die Bereite w.
- Jedes Level ist mit der selben Variable beschriftet.
- Die Verbindung zwischen zwei Levels realisieren Permutationen von Typ [w] $\rightarrow$ [w]. Wobei [w] ist $\{1, \ldots, w\}$.
- Jeder Knote v aus dem $Level_i$ wird mit dem Input 0 oder 1 nur auf Knoten aus dem $Level_{i+1}$ abgebildet. Kurz gesagt, das Programm liegt in Schichtenform.
- Die 1-Kanten sind mit $\boldsymbol{\longrightarrow}$ bezeichnet und die 0-Kanten mit $\longrightarrow$.

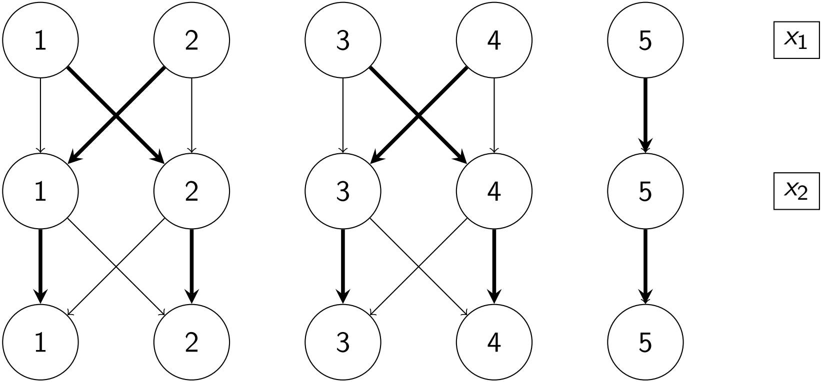
Für eine boolesche Funktion f und eine Permutation σ, sagen wir P σ-berechnet f, falls für jedes Input x gilt:

$$P(x) = \begin{cases} \sigma & , f(x) = 1 \\ \text{e} & , f(x) = 0 \end{cases}$$

wobei e ist die identität Permutation.

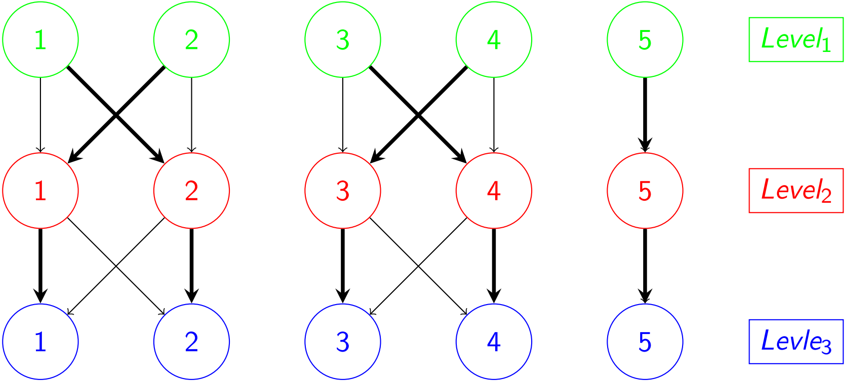
Permutation Branching Programm

Beispiel 2:

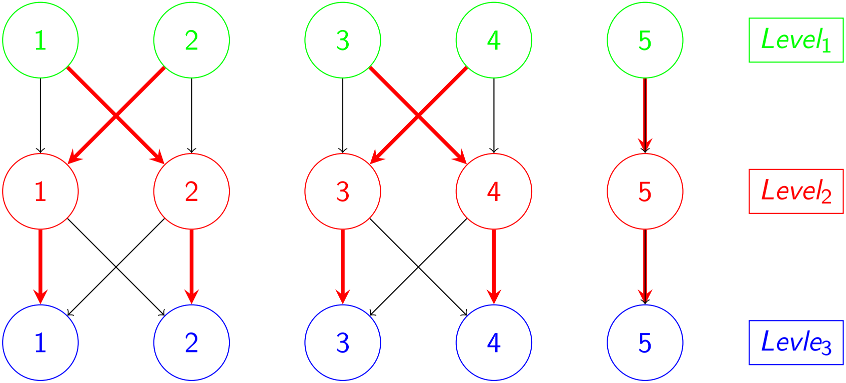


Permutation Branching Programm

Beispiel 2:

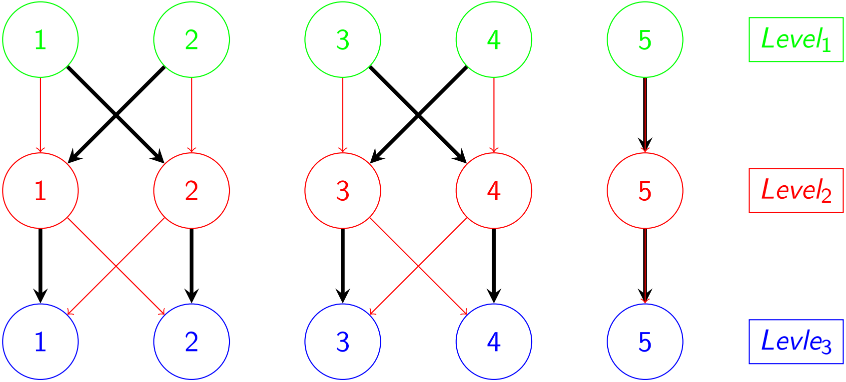


Beispiel 2:



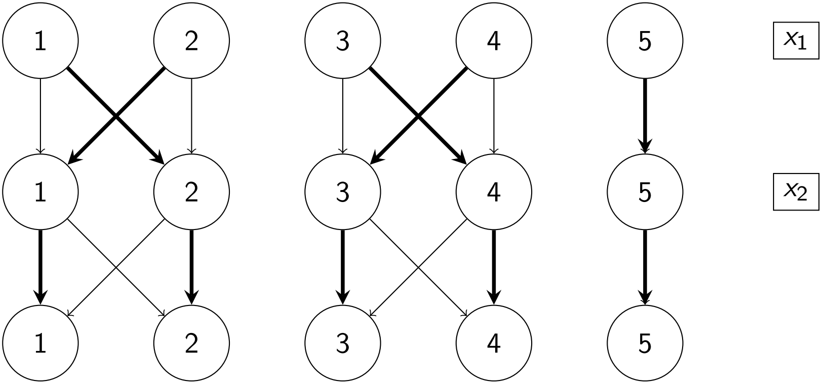
Permutation Branching Programm

Beispiel 2:



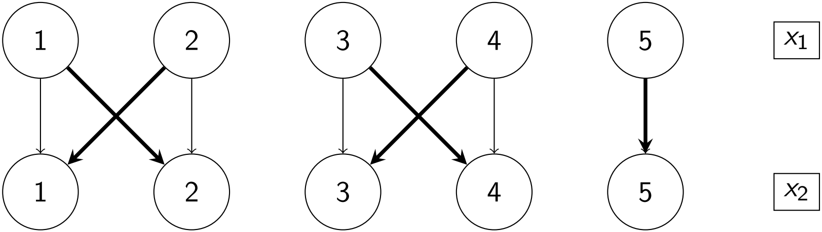
Permutation Branching Programm

Was berechnet P? und was passiert auf die Eingabe (0,0)?

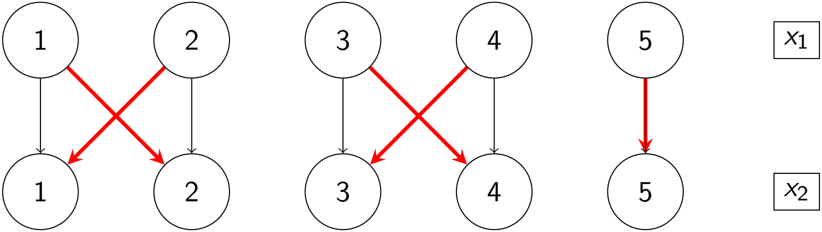


Permutation Branching Programm

Die Permutation für zwischen ersten und zweiten Level:



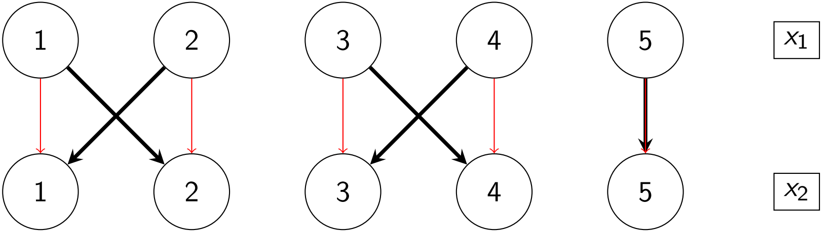
Die Permutation für $1 - \textit{Kanten}$ zwischen dem ersten und zweiten Level:



$$\sigma_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{pmatrix} = (1,2)(3,4)(5)$$

Permutation Branching Programm

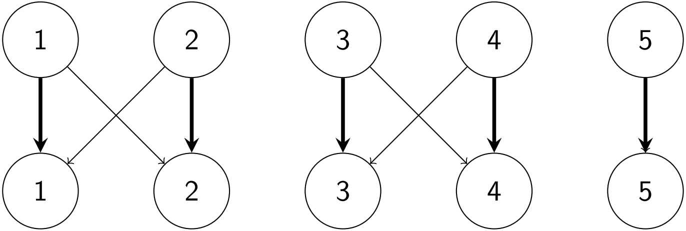
Die Permutation für $0 - \textit{Kanten}$ zwischen dem ersten und zweiten Level:



$$e_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} = id$$

Permutation Branching Programm

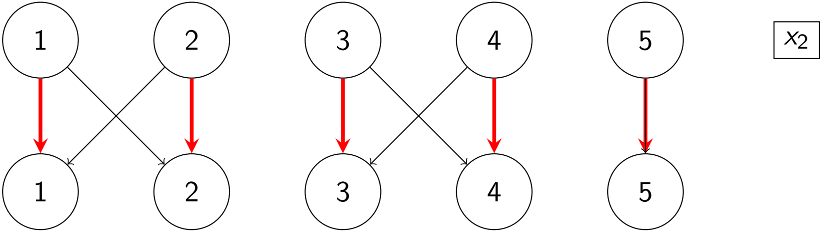
Die Permutation zwischen dem zweiten und dritten Level:



$\boxed{x_2}$

Permutation Branching Programm

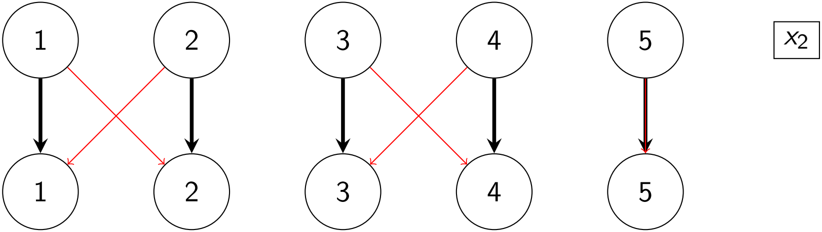
Die Permutation für $1 - \textit{Kanten}$ zwischen dem zweiten und dritten Level:



$$\sigma_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} = id$$

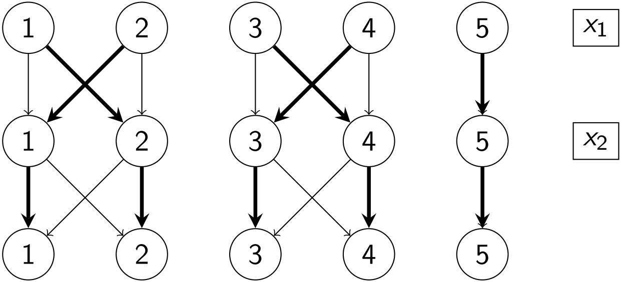
Permutation Branching Programm

Die Permutation für $0-$*Kanten* zwischen dem zweiten und dritten Level:



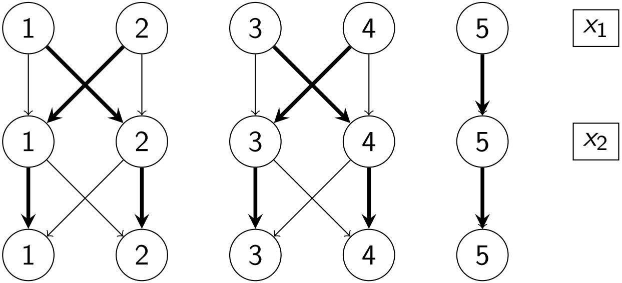
$$e_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 2 & 5 \end{pmatrix} = (1,2)(3,4)(5)$$

Permutation Branching Programm



Bei Eingabe (0,0) $\Rightarrow$ Ausgabe $= e_1 e_2 = (1,2)(3,4)(5)$

Permutation Branching Programm



Bei Eingabe (0,0) $\Rightarrow$ Ausgabe $= e_1 e_2 = (1,2)(3,4)(5)$

Bei Eingabe (1,1) $\Rightarrow$ Ausgabe $= \sigma_1 \sigma_2 = (1,2)(3,4)(5)$

Bei Eingabe (0,1) $\Rightarrow$ Ausgabe $= e_1 \sigma_2 = id$

Bei Eingabe (1,0) $\Rightarrow$ Ausgabe $= \sigma_1 e_2 = id$

Barrington Theorem

Wenn eine boolesche Funktion durch DeMorgan Formel mit polynomischer Größe berechnet werden kann, dann kann sie auch durch ein 5-Branching Programm mit polynomischer Tiefe berechnet werden.

DeMorgan Formel

Formel: ist ein Schaltkreis dessen Gattern höchsten **fan-out 1** haben. Die Größe von einem Formel ist die Anzahl von Gattern.

DeMorgan Schaltkreis: ist ein Schaltkreis über die boolesche Operationen $\{\vee, \wedge\}$, aber die Eingaben sind die Variable und deren Negation.

Also man kann sagen, dass eine DeMorgan Formel ein Schaltkreis mit höchsten **fan-out 1** über die boolesche Operationen $\{\vee, \neg, \wedge\}$ ist.

Barrington Theorem

Satz 1

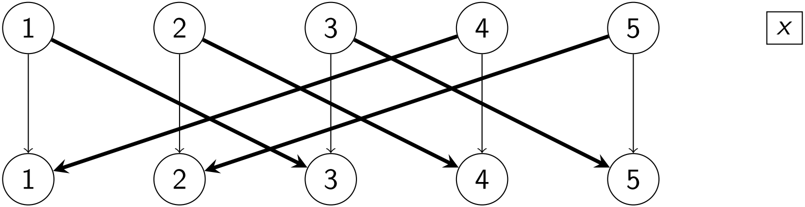
Wenn P σ-*berechnet* f und σ eine zyklische Permutation ist, dann existiert ein Permutation-Branching-Programm $P^{'}$ mit der gleichen Größe wie P, welches τ-*berechnet* f für eine zyklische Permutation τ .

Beweis: sei $P(x) = \sigma = \sigma_1\sigma_2 \dots \sigma_t$.

- σ und τ sind beide zyklische Permutationen.
- dann gilt $\tau = \theta\sigma\theta^{-1}$ für beliebig Permutation θ.
- dann nimm $P^{'}(x) = \theta\sigma_1\sigma_2 \dots \sigma_t\theta^{-1} = \theta\sigma\theta^{-1} = \tau$, indem σ_1 durch $\theta\sigma_1$ und σ_t durch $\sigma_t\theta^{-1}$ ersetzt werden.

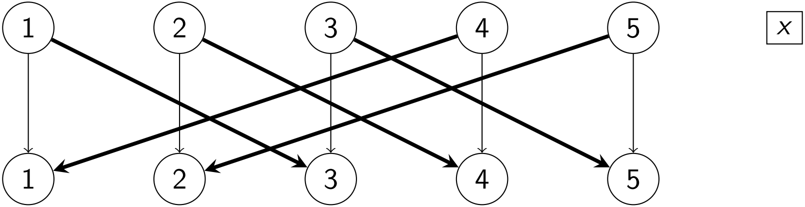
Satz 1

Sei nun P durch die folgende Graph gegeben:



Satz 1

Sei nun P durch die folgende Graph gegeben:

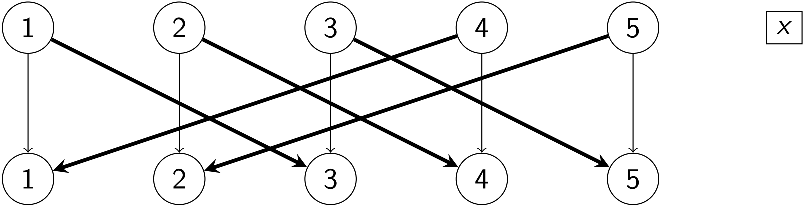


Die 1-Kanten realisieren die Permutation $\sigma_1 = (1, 3, 5, 2, 4)$.

Die 0-Kanten realisieren die Permutation $\sigma_0 = id$.

Satz 1

Sei nun P durch die folgende Graph gegeben:



Die 1-Kanten realisieren die Permutation $\sigma_1 = (1, 3, 5, 2, 4)$.

Die 0-Kanten realisieren die Permutation $\sigma_0 = id$.

$\Rightarrow$ die berechnete Funktion ist $f(x) = x$ mit $\sigma = (1, 3, 5, 2, 4)$.

Satz 1

Sei jetzt

$$\theta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix} \Rightarrow \theta^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}.$$

Satz 1

Sei jetzt

$$\theta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix} \Rightarrow \theta^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}.$$

Jetzt berechnen wir $\theta\sigma_0\theta^{-1}$ und $\theta\sigma_1\theta^{-1}$.

$\theta\sigma_0\theta^{-1} = (1,3)(2,4,5) \circ id \circ (1,3)(2,5,4) = id.$

Satz 1

Sei jetzt

$$\theta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix} \Rightarrow \theta^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}.$$

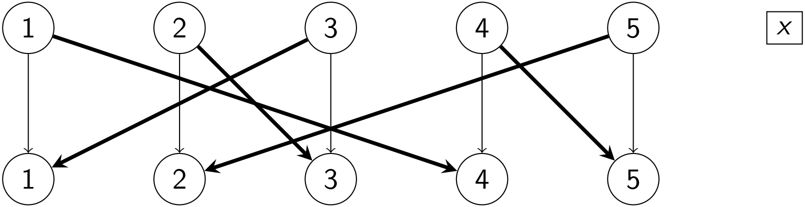
Jetzt berechnen wir $\theta\sigma_0\theta^{-1}$ und $\theta\sigma_1\theta^{-1}$.

$\theta\sigma_0\theta^{-1} = (1,3)(2,4,5) \circ id \circ (1,3)(2,5,4) = id.$

$\theta\sigma_1\theta^{-1} = (1,3)(2,4,5) \circ (1,3,5,2,4) \circ (1,3)(2,5,4) =$
$(1,4,5,2,3).$

Satz 1

Und somit sieht das Graph wie folgt aus:



Das Programm berechnet auch die Funktion $f(x) = x$ aber durch andere Permutation und zwar $\tau = (1, 4, 5, 2, 3)$.

Satz 2 (Negation)

Wenn P *σ-berechnet* f und σ eine zyklische Permutation ist, dann existiert ein Permutation-Branching-Programm mit der selben Größe von P, welches *σ-berechnet* $\neg f$.

Beweis: Nach dem Satz 1 können wir ein PBP $P^{'}$ kriegen, welches σ^{-1}-berechnet f.

- rechne $P^{'}(x) = \sigma^{-1} = \sigma_1\sigma_2\ldots\sigma_t$, σ^{-1}-berechnet f mit Satz-1.
- dann gilt: $P^{'}(x) = \sigma^{-1}$ falls $f(x) = 1$, und $P^{'}(x) = e$ falls $f(x) = 0$.
- nimm $P^{''}(x) = \sigma_1\sigma_2\ldots\sigma_t\sigma$, indem σ_t durch $\sigma_t\sigma$ ersetzt wird.

Angewendet auf Beispiel von Satz 1:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix} \Rightarrow \sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 1 & 2 & 3 \end{pmatrix}.$$

Angewendet auf Beispiel von Satz 1:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix} \Rightarrow \sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 1 & 2 & 3 \end{pmatrix}.$$

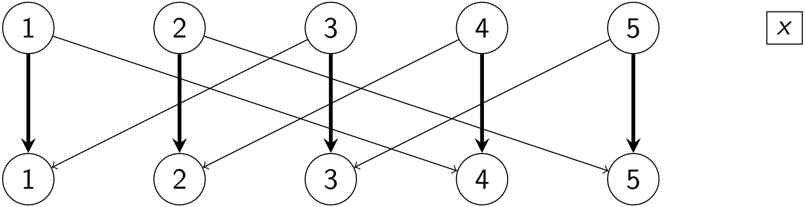
Die neue Permutation für die 1-Kanten ist:

$\sigma^{-1}\sigma = id$.

Die neue Permutation für die 0-Kanten ist:

$\sigma^{-1} id = \sigma^{-1} = (1, 4, 2, 5, 3)$.

Und somit sieht das Graph wie folgt aus:



Das Programm berechnet die Funktion $\neg f(x) = x$.

Satz 3 (AND)

Wenn P *σ-berechnet* f und Q *τ-berechnet* g, dann existiert ein PBP mit der Tiefe $2(|P| + |Q|)$, welches $\sigma\tau\sigma^{-1}\tau^{-1}$*-berechnet* $f \wedge g$.

Beweis: Nach dem Satz 1 bekommen wir ein Programm mit σ^{-1}*-berechnet* f und das andere mit τ^{-1}*-berechnet* g.

Wir komponieren die 4 Programme in diese Reihenfolge $\sigma' = \sigma\tau\sigma^{-1}\tau^{-1}$. Wenn $f = 0$ und $g = 0$, dann ist $\sigma' = \mathrm{id} \circ \mathrm{id} \circ \mathrm{id} \circ \mathrm{id} = \mathrm{id}$. Für den Fall $f = 1$ und $g = 1$ brauchen wir den Folgenden Satz.

Satz 4

Es gibt zwei zyklische Permutationen auf $[5]$ σ und τ so, dass $\sigma\tau\sigma^{-1}\tau^{-1}$ zyklisch ist.

Beweis: Durch ein Beispiel.

Barrington Theorem

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}, \tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \end{pmatrix}$$

Barrington Theorem

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}, \tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \end{pmatrix}$$

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix}, \tau^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 1 & 5 & 3 \end{pmatrix}$$

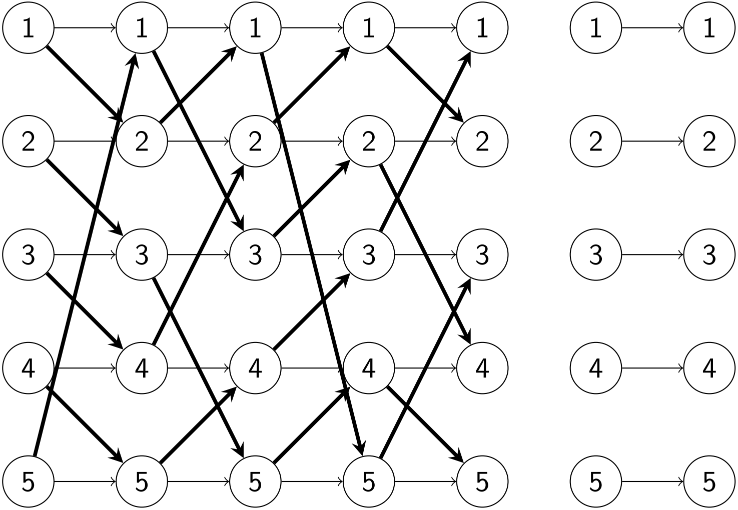
Barrington Theorem

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}, \tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \end{pmatrix}$$

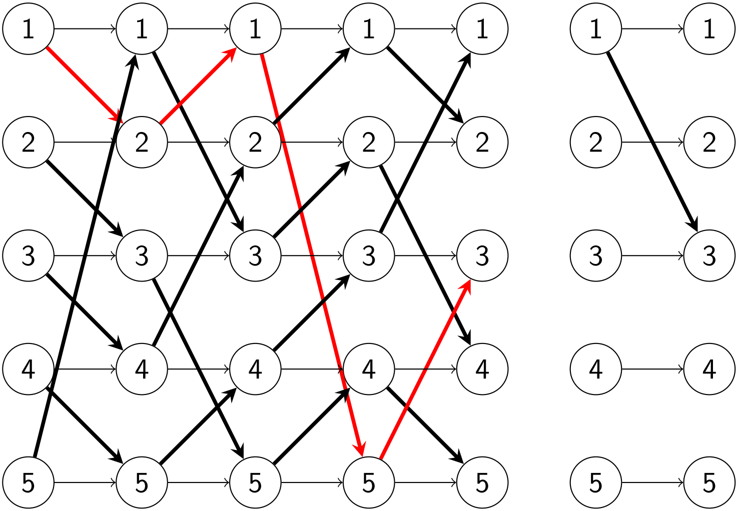
$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix}, \tau^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 1 & 5 & 3 \end{pmatrix}$$

$$\sigma\tau\sigma^{-1}\tau^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 2 & 1 & 4 \end{pmatrix}$$

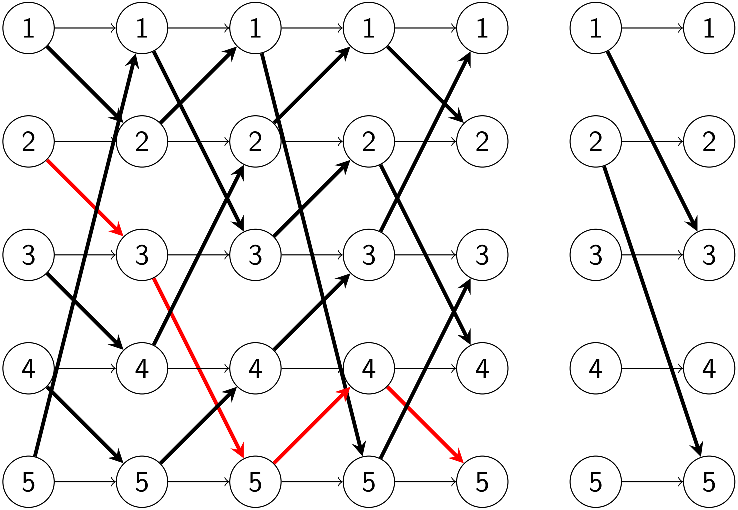
Barrington Theorem



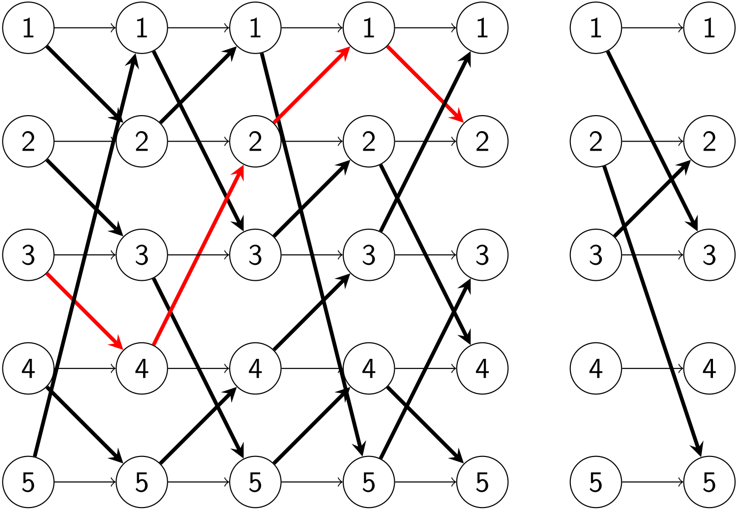
Barrington Theorem



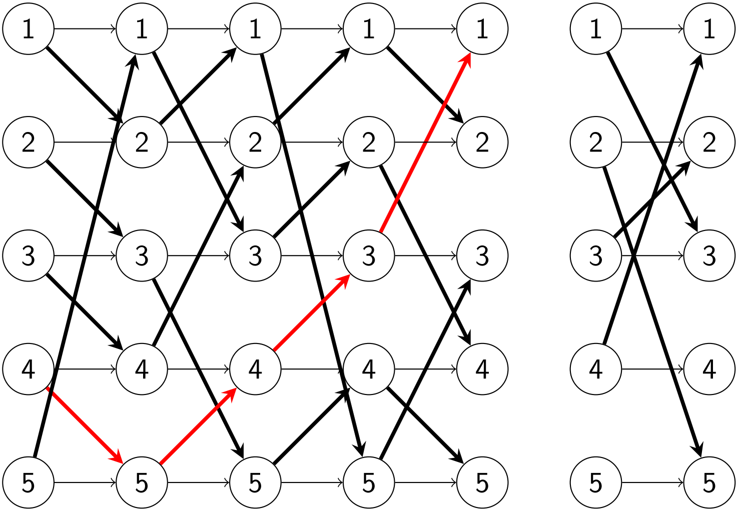
Barrington Theorem

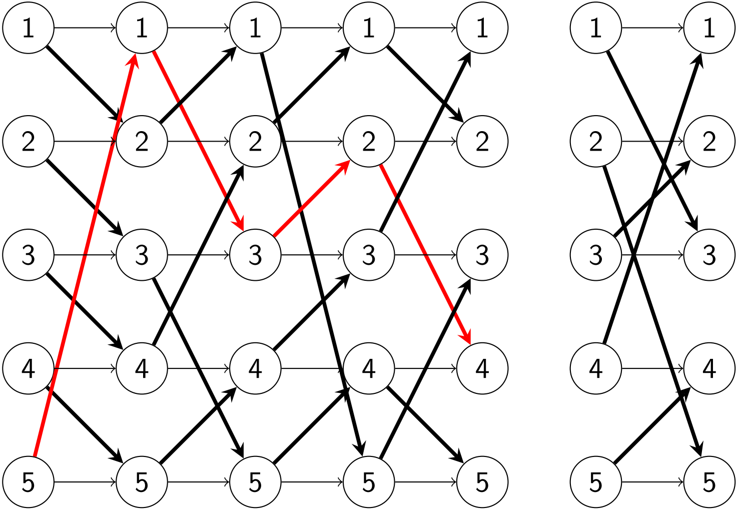


Barrington Theorem



Barrington Theorem




1
1
1
1
1
1
1
2
2
2
2
2
2
2
3
3
3
3
3
3
3
4
4
4
4
4
4
4
5
5
5
5
5
5
5

Barrington Theorem

Theorem 1

Für jede zyklische Permutation auf [5] σ und für jedes DeMorgan Schaltkreis der Tiefe d, die entsprechende Boolesche Funktion kann dank einem 5-PBP der Tiefe höchstens 4^d σ-berechnet werden.

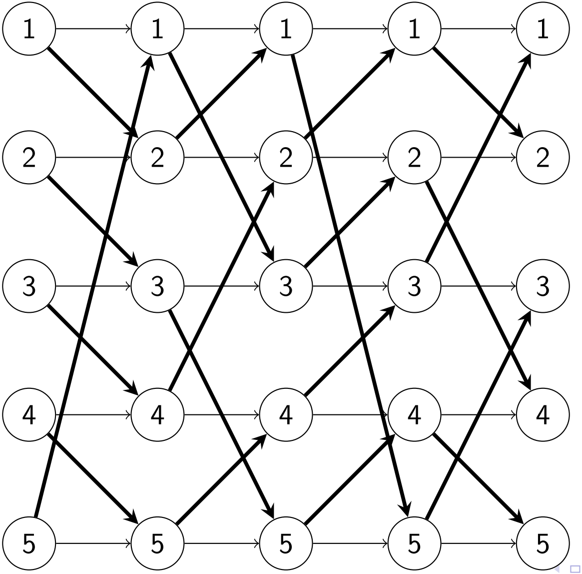
Beweis: Durch Induktion über d.
IA: Für d $= 0 \Rightarrow$ der Schaltkreis ist entweder die Variable x oder ihre Negation $\neg x$. Für $f(x) = x$ wurde ein Beispiel gegeben und nach dem Satz 2 können wir ein 5-PBP konstruieren, welches $f(x) = \neg x$ berechnet.

Barrington Theorem

IS: Für d $\geq$ 1. Nach dem Satz 3 können wir annehmen, dass $f = g \wedge h$, wobei g und h Formeln deren Tiefe d - 1 und deren 5-PBP (nach der Induktion Hypothese) G und H haben höchsten die Tiefe 4^{d-1}.
Nach dem Satz 1 können wir annehmen, dass G σ-*berechnet* g und H τ-*berechnet* h. Nach dem Satz 3 existiert ein 5-PBP mit der Tiefe $2(size(G) + size(H)) \leq 4^d$, welches $\sigma\tau\sigma^{-1}\tau^{-1}$-*berechnet* f.
Nach dem Satz 4 ist dies eine zyklische Permutation $\square$.

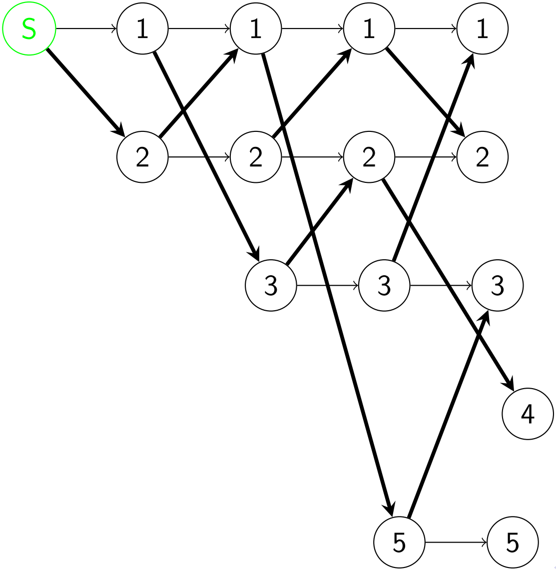
Barrington Theorem

w-Permutation Branching-Programm zu BP umwandeln:



Barrington Theorem

w-Permutation Branching-Programm zu BP umwandeln:



w-Permutation Branching-Programm zu BP umwandeln: