4.1.1 Die Klasse P

Worst-Case Laufzeit $t_M(n)$ einer Turingmaschine M auf Eingaben der Länge n:

$$t_M(n) = \max_{w \in \Sigma^n} t_M(w).$$

4.1.1 Die Klasse P

Worst-Case Laufzeit $t_M(n)$ einer Turingmaschine M auf Eingaben der Länge n:

$$t_M(n) = \max_{w \in \Sigma^n} t_M(w).$$

Definition 4.1

Entscheidungsproblem L gehört zu der **Komplexitätsklasse P**, wenn es eine TM M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

4.1.1 Die Klasse P

Worst-Case Laufzeit $t_M(n)$ einer Turingmaschine M auf Eingaben der Länge n:

$$t_M(n) = \max_{w \in \Sigma^n} t_M(w).$$

Definition 4.1

Entscheidungsproblem L gehört zu der **Komplexitätsklasse P**, wenn es eine TM M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

Beobachtung: Die Klasse P ändert sich nicht, wenn **Registermaschinen im logarithmischen Kostenmaß** statt Turingmaschinen eingesetzt werden.

4.1.1 Die Klasse P

Worst-Case Laufzeit $t_M(n)$ einer Turingmaschine M auf Eingaben der Länge n:

$$t_M(n) = \max_{w \in \Sigma^n} t_M(w).$$

Definition 4.1

Entscheidungsproblem L gehört zu der **Komplexitätsklasse P**, wenn es eine TM M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

Beobachtung: Die Klasse P ändert sich nicht, wenn **Registermaschinen im logarithmischen Kostenmaß** statt Turingmaschinen eingesetzt werden.

Idee: P enthält die **effizient lösbaren Probleme**.
Sinnvoll? $\Theta(n^{100})$ vs. $O(1{,}00000001^n)$

4.1.1 Die Klasse P

Clique in einem Graphen $G = (V, E)$ ist
$V' \subseteq V$ mit $\{u, v\} \in E$ für alle $u, v \in V'$

4.1.1 Die Klasse P

Clique in einem Graphen $G = (V, E)$ ist
$V' \subseteq V$ mit $\{u, v\} \in E$ für alle $u, v \in V'$

Varianten des Cliquenproblems

- **Optimierungsvariante**

  Eingabe: ungerichteter Graph $G = (V, E)$

  Aufgabe: Berechne eine Clique von G mit maximaler Kardinalität.

4.1.1 Die Klasse P

Clique in einem Graphen $G = (V, E)$ ist
$V' \subseteq V$ mit $\{u, v\} \in E$ für alle $u, v \in V'$

Varianten des Cliquenproblems

- **Optimierungsvariante**

  Eingabe: ungerichteter Graph $G = (V, E)$

  Aufgabe: Berechne eine Clique von G mit maximaler Kardinalität.

- **Wertvariante**

  Eingabe: ungerichteter Graph $G = (V, E)$

  Aufgabe: Berechne das größte $k^\star \in \mathbb{N}$, für das es eine $k^\star$-Clique in G gibt.

4.1.1 Die Klasse P

Clique in einem Graphen $G = (V, E)$ ist
$V' \subseteq V$ mit $\{u, v\} \in E$ für alle $u, v \in V'$

Varianten des Cliquenproblems

- **Optimierungsvariante**
  Eingabe: ungerichteter Graph $G = (V, E)$
  Aufgabe: Berechne eine Clique von G mit maximaler Kardinalität.
- **Wertvariante**
  Eingabe: ungerichteter Graph $G = (V, E)$
  Aufgabe: Berechne das größte $k^\star \in \mathbb{N}$, für das es eine $k^\star$-Clique in G gibt.
- **Entscheidungsvariante**
  Eingabe: ungerichteter Graph $G = (V, E)$ und ein Wert $k \in \mathbb{N}$
  Aufgabe: Entscheide, ob es in G eine Clique der Größe mindestens k gibt.

Theorem 4.2

Entweder gibt es für alle drei Varianten des Cliquenproblems polynomielle Algorithmen oder für gar keine.

Theorem 4.2

Entweder gibt es für alle drei Varianten des Cliquenproblems polynomielle Algorithmen oder für gar keine.

Beweis:

Optimierungsvariante polynomiell lösbar.

$\Rightarrow$ Wertvariante polynomiell lösbar.

$\Rightarrow$ Entscheidungsvariante polynomiell lösbar.

4.1.1 Die Klasse P

Entscheidungsvariante polynomiell lösbar. $\Rightarrow$ Wertvariante polynomiell lösbar.

Eingabe für Wertvariante: Graph G mit N Knoten,
Codierungslänge $n = N^2$ als Adjazenzmatrix

4.1.1 Die Klasse P

Entscheidungsvariante polynomiell lösbar. $\Rightarrow$ Wertvariante polynomiell lösbar.

Eingabe für Wertvariante: Graph G mit N Knoten,
Codierungslänge $n = N^2$ als Adjazenzmatrix

Annahme: Es gibt polynomiellen Algorithmus A für Entscheidungsvariante.
Sei $A(G, k) \in \{0, 1\}$ die Ausgabe von A bei Eingabe (G, k).

4.1.1 Die Klasse P

Entscheidungsvariante polynomiell lösbar. $\Rightarrow$ Wertvariante polynomiell lösbar.

Eingabe für Wertvariante: Graph G mit N Knoten,
Codierungslänge $n = N^2$ als Adjazenzmatrix

Annahme: Es gibt polynomiellen Algorithmus A für Entscheidungsvariante.
Sei $A(G, k) \in \{0, 1\}$ die Ausgabe von A bei Eingabe (G, k).

Vorgehen: Löse Wertvariante mithilfe von maximal N Aufrufen des Algorithmus A:

$$k^\star = \max\{k \in \{1, \ldots, N\} \mid A(G, k) = 1\}.$$

4.1.1 Die Klasse P

Entscheidungsvariante polynomiell lösbar. $\Rightarrow$ Wertvariante polynomiell lösbar.

Eingabe für Wertvariante: Graph G mit N Knoten,
Codierungslänge $n = N^2$ als Adjazenzmatrix

Annahme: Es gibt polynomiellen Algorithmus A für Entscheidungsvariante.
Sei $A(G, k) \in \{0, 1\}$ die Ausgabe von A bei Eingabe (G, k).

Vorgehen: Löse Wertvariante mithilfe von maximal N Aufrufen des Algorithmus A:

$$k^\star = \max\{k \in \{1, \ldots, N\} \mid A(G, k) = 1\}.$$

Laufzeit: Es gibt Konstante $\alpha \in \mathbb{N}$, sodass die Laufzeit durch $O(N \cdot (n')^\alpha)$ beschränkt ist.
Dabei ist $n' \leq N^2 + \lceil \log_2(N) \rceil = O(N^2)$ die Codierungslänge von (G, k).

4.1.1 Die Klasse P

Entscheidungsvariante polynomiell lösbar. $\Rightarrow$ Wertvariante polynomiell lösbar.

Eingabe für Wertvariante: Graph G mit N Knoten,
Codierungslänge $n = N^2$ als Adjazenzmatrix

Annahme: Es gibt polynomiellen Algorithmus A für Entscheidungsvariante.
Sei $A(G, k) \in \{0, 1\}$ die Ausgabe von A bei Eingabe (G, k).

Vorgehen: Löse Wertvariante mithilfe von maximal N Aufrufen des Algorithmus A:

$$k^\star = \max\{k \in \{1, \ldots, N\} \mid A(G, k) = 1\}.$$

Laufzeit: Es gibt Konstante $\alpha \in \mathbb{N}$, sodass die Laufzeit durch $O(N \cdot (n')^\alpha)$ beschränkt ist.
Dabei ist $n' \leq N^2 + \lceil \log_2(N) \rceil = O(N^2)$ die Codierungslänge von (G, k).

Insgesamt erhalten wir $O(N \cdot (N^2)^\alpha) = O(n^{\alpha+1})$.

4.1.1 Die Klasse P

Wertvariante polynomiell lösbar. $\Rightarrow$ Optimierungsvariante polynomiell lösbar.

A sei polynomieller Algorithmus für die Wertvariante.

Wertvariante polynomiell lösbar. $\Rightarrow$ Optimierungsvariante polynomiell lösbar.

A sei polynomieller Algorithmus für die Wertvariante.

$\mathbf{A_{opt}(G)}$

1 $k^\star = A(G)$;
2 $V' := V = \{v_1, \ldots, v_N\}$;
3 **for** $(i = 1; i \leq N; i{+}{+})$
4 $\quad$ G' sei induzierter Teilgraph von G mit Knotenmenge $V' \setminus \{v_i\}$.
5 $\quad$ **if** $(A(G') == k^\star)$ $V' := V' \setminus \{v_i\}$;
6 **return** V';

4.1.1 Die Klasse P

Wertvariante polynomiell lösbar. $\Rightarrow$ Optimierungsvariante polynomiell lösbar.

A sei polynomieller Algorithmus für die Wertvariante.

```
    A_opt(G)
1   k* = A(G);
2   V' := V = {v_1, ..., v_N};
3   for (i = 1; i ≤ N; i++)
4       G' sei induzierter Teilgraph von G mit Knotenmenge V' \ {v_i}.
5       if (A(G') == k*) V' := V' \ {v_i};
6   return V';
```

Invariante: V' enthält zu jedem Zeitpunkt eine $k^\star$-Clique. Ein Knoten, der in Zeile 5 nicht aus der Menge V' entfernt wird, ist in jeder $k^\star$-Clique $V^\star \subseteq V'$ enthalten.

Wertvariante polynomiell lösbar. $\Rightarrow$ Optimierungsvariante polynomiell lösbar.

A sei polynomieller Algorithmus für die Wertvariante.

```
    A_opt(G)
1   k* = A(G);
2   V' := V = {v_1, ..., v_N};
3   for (i = 1; i ≤ N; i++)
4       G' sei induzierter Teilgraph von G mit Knotenmenge V' \ {v_i}.
5       if (A(G') == k*) V' := V' \ {v_i};
6   return V';
```

Invariante: V' enthält zu jedem Zeitpunkt eine $k^\star$-Clique. Ein Knoten, der in Zeile 5 nicht aus der Menge V' entfernt wird, ist in jeder $k^\star$-Clique $V^\star \subseteq V'$ enthalten.

Laufzeit von A_{opt} beträgt $O(N \cdot (N^2)^\alpha)$, wenn die Laufzeit von A auf Graphen mit N Knoten durch $O((N^2)^\alpha)$ nach oben beschränkt ist. $\square$

Beispiele:

Zusammenhangsproblem in ungerichteten Graphen:

Eingabe: ungerichteter Graph G mit n Knoten.

Codierungslänge n^2 als Adjazenzmatrix

Beispiele:

Zusammenhangsproblem in ungerichteten Graphen:

Eingabe: ungerichteter Graph G mit n Knoten.

Codierungslänge n^2 als Adjazenzmatrix

Lösung mittels **Tiefensuche**:

Laufzeit auf RAM im logarithmischen Kostenmaß $O(n^2 \log n) = O(n^3)$.

$\Rightarrow$ Das Zusammenhangsproblem gehört zu P.

4.1.1 Die Klasse P

Beispiele:

Zusammenhangsproblem in ungerichteten Graphen:

Eingabe: ungerichteter Graph G mit n Knoten.

Codierungslänge n^2 als Adjazenzmatrix

Lösung mittels **Tiefensuche**:

Laufzeit auf RAM im logarithmischen Kostenmaß $O(n^2 \log n) = O(n^3)$.

$\Rightarrow$ Das Zusammenhangsproblem gehört zu P.

Generell trifft dies auf alle Graphprobleme zu, die mit Algorithmen gelöst werden können, deren Laufzeit polynomiell in der Anzahl der Knoten und Kanten beschränkt ist.

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

4.1.1 Die Klasse P

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

Lösung mittels **Algorithmus von Kruskal**

4.1.1 Die Klasse P

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

Lösung mittels **Algorithmus von Kruskal**
Laufzeit für Sortieren der Kanten: $O(m \log(m) \cdot \log(W))$ für $W = \max_{e \in E} w(e)$

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

Lösung mittels **Algorithmus von Kruskal**
Laufzeit für Sortieren der Kanten: $O(m \log(m) \cdot \log(W))$ für $W = \max_{e \in E} w(e)$
Laufzeit für restliche Schritte: $O(m \log(m) \cdot \log(n))$

4.1.1 Die Klasse P

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

Lösung mittels **Algorithmus von Kruskal**
Laufzeit für Sortieren der Kanten: $O(m \log(m) \cdot \log(W))$ für $W = \max_{e \in E} w(e)$
Laufzeit für restliche Schritte: $O(m \log(m) \cdot \log(n))$
Insgesamt: $O(m \log m \cdot \max\{\log(W), \log(n)\})$

4.1.1 Die Klasse P

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

Lösung mittels **Algorithmus von Kruskal**
Laufzeit für Sortieren der Kanten: $O(m \log(m) \cdot \log(W))$ für $W = \max_{e \in E} w(e)$
Laufzeit für restliche Schritte: $O(m \log(m) \cdot \log(n))$
Insgesamt: $O(m \log m \cdot \max\{\log(W), \log(n)\})$

Eingabelänge als Adjazenzliste: $\Omega(m \log(n) + \log(W))$

Spannbaumproblem

Eingabe: ungerichteter Graph $G = (V, E)$ mit n Knoten und $m \geq n - 1$ Kanten
Gewichte $w : E \to \mathbb{N}$

Lösung mittels **Algorithmus von Kruskal**
Laufzeit für Sortieren der Kanten: $O(m \log(m) \cdot \log(W))$ für $W = \max_{e \in E} w(e)$
Laufzeit für restliche Schritte: $O(m \log(m) \cdot \log(n))$
Insgesamt: $O(m \log m \cdot \max\{\log(W), \log(n)\})$

Eingabelänge als Adjazenzliste: $\Omega(m \log(n) + \log(W))$

Es gilt

$$m \log m \cdot \max\{\log(W), \log(n)\} = O((m \log(n) + \log(W))^2)$$

$\Rightarrow$ Das Spannbaumproblem gehört zu P.

Cliquenproblem

Eingabe: $G = (V, E)$ mit $n = |V|$ und $k \in \mathbb{N}$

Cliquenproblem

Eingabe: $G = (V, E)$ mit $n = |V|$ und $k \in \mathbb{N}$

Algorithmus: Teste alle Teilmengen von V der Größe k darauf, ob sie eine Clique bilden.

4.1.1 Die Klasse P

Cliquenproblem

Eingabe: $G = (V, E)$ mit $n = |V|$ und $k \in \mathbb{N}$

Algorithmus: Teste alle Teilmengen von V der Größe k darauf, ob sie eine Clique bilden.

Laufzeit: $\Theta\left(\text{poly}(n) \cdot \binom{n}{k}\right) = \Theta\left(\text{poly}(n) \cdot \left(\frac{n}{k}\right)^k\right)$

4.1.1 Die Klasse P

Cliquenproblem

Eingabe: $G = (V, E)$ mit $n = |V|$ und $k \in \mathbb{N}$

Algorithmus: Teste alle Teilmengen von V der Größe k darauf, ob sie eine Clique bilden.

Laufzeit: $\Theta\left(\text{poly}(n) \cdot \binom{n}{k}\right) = \Theta\left(\text{poly}(n) \cdot \left(\frac{n}{k}\right)^k\right)$

Eingabelänge: $O(n^2 + \log k)$

Cliquenproblem

Eingabe: $G = (V, E)$ mit $n = |V|$ und $k \in \mathbb{N}$

Algorithmus: Teste alle Teilmengen von V der Größe k darauf, ob sie eine Clique bilden.

Laufzeit: $\Theta\left(\text{poly}(n) \cdot \binom{n}{k}\right) = \Theta\left(\text{poly}(n) \cdot \left(\frac{n}{k}\right)^k\right)$

Eingabelänge: $O(n^2 + \log k)$

Laufzeit nur polynomiell, wenn k eine Konstante ist.

Rucksackproblem

Eingabe: Nutzenwerte $p_1, \ldots, p_N \in \mathbb{N}$, Gewichte $w_1, \ldots, w_N \in \mathbb{N}$, Kapazität $t \in \mathbb{N}$, Schranke $z \in \mathbb{N}$.

Frage: Gibt es Teilmenge $I \subseteq \{1, \ldots, N\}$ der Objekte mit $\sum_{i \in I} w_i \leq t$ und $\sum_{i \in I} p_i \geq z$.

Rucksackproblem

Eingabe: Nutzenwerte $p_1, \ldots, p_N \in \mathbb{N}$, Gewichte $w_1, \ldots, w_N \in \mathbb{N}$, Kapazität $t \in \mathbb{N}$, Schranke $z \in \mathbb{N}$.

Frage: Gibt es Teilmenge $I \subseteq \{1, \ldots, N\}$ der Objekte mit $\sum_{i \in I} w_i \leq t$ und $\sum_{i \in I} p_i \geq z$.

Algorithmus: Dynamische Programmierung mit Laufzeit $O(N^2 W \log P)$ für $W = \max_i w_i$ und $P = \sum_i p_i$.

Rucksackproblem

Eingabe: Nutzenwerte $p_1, \ldots, p_N \in \mathbb{N}$, Gewichte $w_1, \ldots, w_N \in \mathbb{N}$, Kapazität $t \in \mathbb{N}$, Schranke $z \in \mathbb{N}$.

Frage: Gibt es Teilmenge $I \subseteq \{1, \ldots, N\}$ der Objekte mit $\sum_{i \in I} w_i \leq t$ und $\sum_{i \in I} p_i \geq z$.

Algorithmus: Dynamische Programmierung mit Laufzeit $O(N^2 W \log P)$ für $W = \max_i w_i$ und $P = \sum_i p_i$.

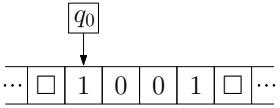
Dies ist i. A. nicht polynomiell, da die Eingabegröße mit $\log W$ wächst und nicht mit W.

4.1.2 Die Klasse NP

Definition 2.1

Eine **Turingmaschine (TM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ ist die **Zustandsüberführungsfunktion**.

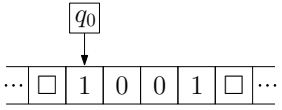


$\delta(q_0, 1) = (q, 1, R)$

Definition 4.3

Eine **nichtdeterministische Turingmaschine (NTM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta \subseteq ((Q \setminus \{\bar{q}\}) \times \Gamma) \times (Q \times \Gamma \times \{L, N, R\})$ ist die **Zustandsüberführungsrelation**.

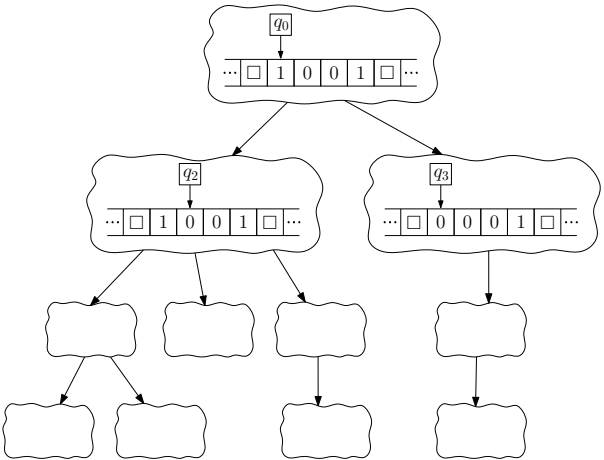


$((q_0, 1), (q, 1, R)) \in \delta$

oder

$((q_0, 1), (q, 0, L)) \in \delta$

Rechenbaum einer Turingmaschine:



Konfiguration = Zustand, Bandinhalt, Kopfposition

Wurzel

= Startkonfiguration

Kante

= erlaubter Übergang

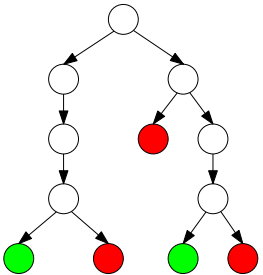
Blatt

= Konfiguration ohne erlaubten Übergang in δ

Rechenweg = Weg von der Wurzel zu einem Blatt

Definition 4.4

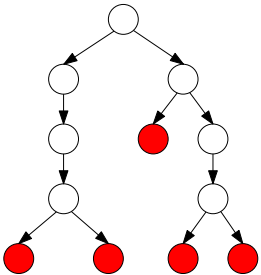
Eine NTM M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn es Rechenweg von M gibt, der bei Eingabe w zu einer akzeptierenden Endkonfiguration führt.



Eingabe wird akzeptiert

Definition 4.4

Eine NTM M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn es Rechenweg von M gibt, der bei Eingabe w zu einer akzeptierenden Endkonfiguration führt.

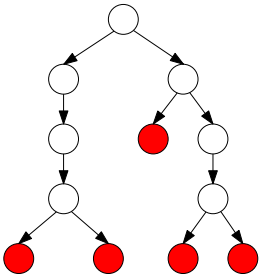


Eingabe wird nicht akzeptiert

Definition 4.4

Eine NTM M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn es Rechenweg von M gibt, der bei Eingabe w zu einer akzeptierenden Endkonfiguration führt.

Sei $L(M) \subseteq \Sigma^*$ die Menge der von M **akzeptierten Eingaben**. M **entscheidet** die Sprache $L(M)$, wenn sie für jede Eingabe auf jedem Rechenweg hält.

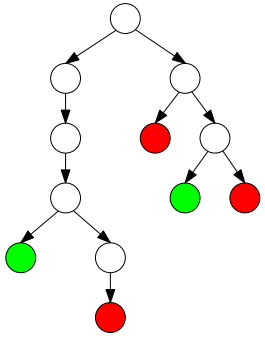


Eingabe wird nicht akzeptiert

4.1.2 Die Klasse NP

Definition 4.5

Die **Laufzeit** $t_M(w)$ einer nichtdeterministischen Turingmaschine M auf einer Eingabe $w \in \Sigma^*$ ist definiert als die **Länge des längsten Rechenweges** von M bei Eingabe w.

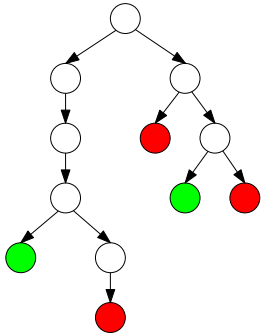


Laufzeit = 5

Definition 4.5

Die **Laufzeit** $t_M(w)$ einer nichtdeterministischen Turingmaschine M auf einer Eingabe $w \in \Sigma^*$ ist definiert als die **Länge des längsten Rechenweges** von M bei Eingabe w.

Gibt es bei Eingabe w einen Rechenweg, auf dem M nicht terminiert, so ist die Laufzeit unendlich.



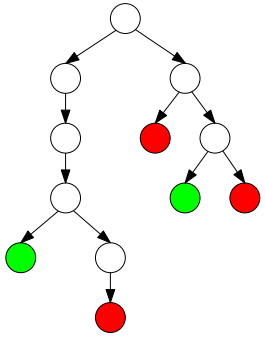
Laufzeit = 5

Definition 4.5

Die **Laufzeit** $t_M(w)$ einer nichtdeterministischen Turingmaschine M auf einer Eingabe $w \in \Sigma^*$ ist definiert als die **Länge des längsten Rechenweges** von M bei Eingabe w.

Gibt es bei Eingabe w einen Rechenweg, auf dem M nicht terminiert, so ist die Laufzeit unendlich.

Sei $t_M(n) = \max_{w \in \Sigma^n} t_M(w)$ die **Worst-Case-Laufzeit** für Eingaben der Länge $n \in \mathbb{N}$.



Laufzeit = 5

4.1.2 Die Klasse NP

Definition 4.6

Ein Entscheidungsproblem L gehört genau dann zu der **Komplexitätsklasse NP**, wenn es eine nichtdeterministische Turingmaschine M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

4.1.2 Die Klasse NP

Definition 4.6

Ein Entscheidungsproblem L gehört genau dann zu der **Komplexitätsklasse NP**, wenn es eine nichtdeterministische Turingmaschine M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

NP wurde nicht mit dem Ziel definiert, ein physikalisch realisierbares Rechnermodell zu finden, sondern als **theoretisches Hilfsmittel**.

4.1.2 Die Klasse NP

Definition 4.6

Ein Entscheidungsproblem L gehört genau dann zu der **Komplexitätsklasse NP**, wenn es eine nichtdeterministische Turingmaschine M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

NP wurde nicht mit dem Ziel definiert, ein physikalisch realisierbares Rechnermodell zu finden, sondern als **theoretisches Hilfsmittel**.

Theorem 4.7

Die Entscheidungsvarianten des Cliquenproblems und des Rucksackproblems gehören zu NP.

Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

4.1.2 Die Klasse NP

Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

4.1.2 Die Klasse NP

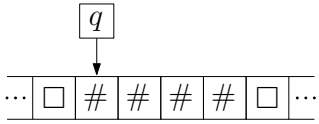
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

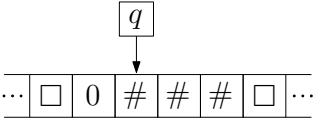
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

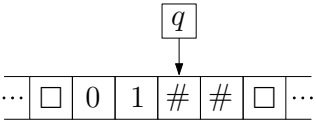
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

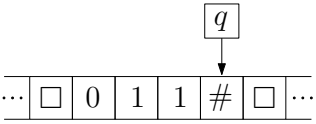
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

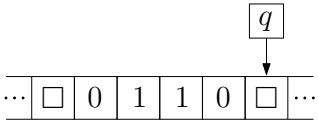
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

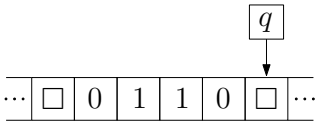
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

4.1.2 Die Klasse NP

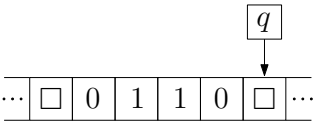
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

Phase 2: Akzeptiere genau dann, wenn V' die Größe k besitzt und eine Clique in G ist.

4.1.2 Die Klasse NP

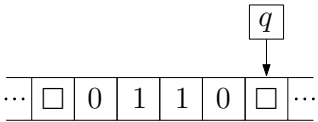
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

Phase 2: Akzeptiere genau dann, wenn V' die Größe k besitzt und eine Clique in G ist.

Laufzeit ist polynomiell.

4.1.2 Die Klasse NP

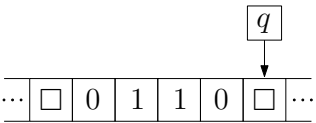
Beweis: Wir starten mit dem Cliquenproblem.

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

Phase 2: Akzeptiere genau dann, wenn V' die Größe k besitzt und eine Clique in G ist.

Laufzeit ist polynomiell.

Es gibt k-Clique in G. $\iff$ Es gibt akzeptierenden Rechenweg von M.