

# Theoretische Informatik 3

Till Nierhoff

`nierhoff@informatik.hu-berlin.de`

Institut für Informatik

Humboldt-Universität zu Berlin

8. Juli 2003

## DEFINITION

Gegeben:

- Menge  $U$  mit  $n$  Elementen,
- $\mathcal{H} \subset 2^U$ ,      **OBdA**  $U = \bigcup \mathcal{H}$
- $c : \mathcal{H} \rightarrow \mathbb{Q}_+$

Gesucht:

- $S \subset \mathcal{H}$ , so dass
- $\bigcup S = U$
- $c(S) = \sum_{H \in S} c(H)$  minimal

## Algorithmus SCGREEDY

- Eingabe  $U, \mathcal{H}, c$
- $C \leftarrow \emptyset, S \leftarrow \emptyset$
- while  $C \neq U$ 
  - $H \leftarrow \operatorname{argmin} \left\{ \frac{c(H)}{|H \setminus C|} \mid H \in \mathcal{H} \right\}$  dabei sei  $\frac{c}{0} = \infty$
  - $C \leftarrow C \cup H$
  - $S \leftarrow S \cup \{H\}$
- Ausgabe  $S$

## Algorithmus SCGREEDY

- Eingabe  $U, \mathcal{H}, c$
- $C \leftarrow \emptyset, S \leftarrow \emptyset$
- while  $C \neq U$ 
  - $H \leftarrow \operatorname{argmin} \left\{ \frac{c(H)}{|H \setminus C|} \mid H \in \mathcal{H} \right\}$  dabei sei  $\frac{c}{0} = \infty$
  - $H' \leftarrow H \setminus C$
  - foreach  $e \in H'$ :  $p(e) \leftarrow \frac{c(H)}{|H \setminus C|}$
  - $C \leftarrow C \cup H$
  - $S \leftarrow S \cup \{H\}$
- Ausgabe  $S$

$p : U \rightarrow \mathbb{Q}_+$  und  $H'$  nur für Analyse

## Randomisierte Algorithmen

- können (faire) Münzen werfen
- und damit zB von allen Elementen einer Menge eines zufällig gleichverteilt auswählen

## Randomisierte Algorithmen

- können (faire) Münzen werfen
- und damit zB von allen Elementen einer Menge eines zufällig gleichverteilt auswählen

## Beispiele:

- QuickSort
- MinCut

## QuickSort sortiert

- eine gegebene Liste
- bezüglich einer gegebenen Vergleichsoperation
- Eingabe: Liste  $S$ , Vergleichsoperation  $\leq$

## QuickSort sortiert

- eine gegebene Liste
- bezüglich einer gegebenen Vergleichsoperation
  
- Eingabe: Liste  $S$ , Vergleichsoperation  $\leq$
- Wähle  $p \in S$  **gleichverteilt** (Falls  $S = \emptyset$ : Ausgabe  $\emptyset$ )
- $T \leftarrow \emptyset, U \leftarrow \emptyset$
- foreach  $s \in S \setminus \{p\}$ :
  - if  $s \leq p$ :  $U \leftarrow s$
  - else:  $T \leftarrow s$
- Ausgabe: QuickSort( $T, \leq$ ),  $s$ , QuickSort( $U, \leq$ )

## THEOREM

QuickSort benötigt im schlechtesten Fall (**Worst Case**)  $\binom{n}{2} = \Theta(n^2)$  Vergleiche. Die erwartete Anzahl Vergleiche (**Average Case**) ist  $2n \ln n$ .

## THEOREM

QuickSort benötigt im schlechtesten Fall (**Worst Case**)  $\binom{n}{2} = \Theta(n^2)$  Vergleiche. Die erwartete Anzahl Vergleiche (**Average Case**) ist  $2n \ln n$ .

- Ausgabe **unabhängig** vom Zufall korrekt
- Laufzeit **abhängig** vom Zufall

→ „Las Vegas“-Algorithmus