

Föreläsning 19 i ADK20

Probabilistiska algoritmer

Stefan Nilsson

KTH

Probabilistiska algoritmer

Algoritmer som använder slump.

Exempel: quicksort med slump

```
function RANDOMQUICKSORT( $v[i..j]$ )  
  if  $i < j$  then  
     $x \leftarrow v[\text{random}(i,j)]$   
     $m \leftarrow \text{Partition}(v[i..j], x)$   
    RANDOMQUICKSORT( $v[i..m-1]$ )  
    RANDOMQUICKSORT( $v[m+1..j]$ )
```

Analys: (Beviset skippat)

- Med sannolikhet $> 1 - \frac{1}{n^k}$ tar RANDOMQUICKSORT tid $\mathcal{O}(n \log n)$, för varje konstant k .

Las Vegas-algorithm - Probabilistisk algoritm som alltid svarar rätt, men kan ta olika lång tid på sig

Exempel: MAX3CNFSAT

Givet en mängd $\{c_i\}_{i=1}^n$ med klausuler över booleska variabler $x_1, \dots, x_m$, hitta en variabeltilldelning som satisfierar så många klausuler som möjligt.

```
function RANDOMMAX3CNFSAT( $\{c_i\}$ )  
  for  $i \leftarrow$  to  $m$  do  
     $x_i \leftarrow$  RANDOM(0, 1)  
  return  $\{x_i\}$ 
```

Analys:

- Tid: $\mathcal{O}(n)$
- $P[c_i \text{ inte satisfierad}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$
- Y = antal satisfierade platser
- $E[Y] = n \cdot P[c_i \text{ satisfierad}] = n(1 - \frac{1}{8}) = \frac{7}{8}n$
- $\Rightarrow$ approximerar MAX3CNFSAT inom $\frac{8}{7}$

Monte Carlo-algoritm - Probabilistisk algoritm som alltid går snabbt, men som bara ibland ger bra resultat

Gör en probabilistisk algoritm deterministisk genom att arbeta bort slumpen.

Metod 1: Pröva alla utfall

- Om k slumpbitar behövs i en algoritm finns det 2^k möjliga utfall.
- Testa alla dessa och plocka det bästa!
- Problem: Tar lång tid om k är stort

Metod 2: Betingade väntevärden

- Beräkna $E[Y|x_1 = 0]$ och $E[Y|x_1 = 1]$
- Välj det värde på x_1 som ger störst betingat väntevärde
- Fortsätt på samma sätt med resterande x_i
- Problem: Betingade väntevärdena kan vara svåra att beräkna

Fler metoder finns!

Går det alltid att slumpeliminera en probabilistisk polynomisk algoritm till en deterministisk polynomisk algoritm?