Fö. 2: Giriga algoritmer.

Fö. 3: Rekursiva algoritmer.

Fö. 4: Dynamisk programmering.

Exempel 1: Majoritetsproblemet.

Exempel 2: Heltalsmultiplikation

Exempel 3: Max SAT

Rekursiva algoritmer

Rekursiva algoritmer

Målet är (oftast) att dela data i ungefär lika stora delar.

Rekursiva algoritmer

Målet är (oftast) att dela data i ungefär lika stora delar.

Rekursiva ekvationer av typen

$$T(n) = aT(n/b) + \mathrm{poly}(n)$$

har alltid polynomiska lösningar då $b > 1$.

Exempel 1: Majoritetsproblemet

Majoritetsproblemet

Problem. Majoritet.
Indata. Sekvens av objekt $[o_1, \ldots, o_n]$.
Fråga. Förekommer något objekt i majoritet, dvs. $\lceil \frac{n}{2} \rceil$ eller fler gånger? I så fall vilket?

Majoritetsproblemet

Objekt kan endast jämföras med *likhetsrelationen*.

Majoritetsproblemet

Objekt kan endast jämföras med *likhetsrelationen*.

Antag att jämförelse kan göras i $O(1)$ tid.

Majoritetsproblemet

Objekt kan endast jämföras med *likhetsrelationen*.

Antag att jämförelse kan göras i $O(1)$ tid.

Finns en linjär ordning på objekten kan problemet lösas i $O(n \log n)$ med hjälp av mergesort.

Majoritetsproblemet

Algoritm MAJ.

Majoritetsproblemet

Algoritm MAJ.

Indata $[o_1, \ldots, o_n]$

Majoritetsproblemet

Algoritm MAJ.

Indata $[o_1, \ldots, o_n]$

(basfall) $[o_1]$, $[o_1, o_2]$, $[o_1, o_2, o_3]$

Majoritetsproblemet

Algoritm MAJ.

Indata $[o_1, \ldots, o_n]$

(basfall) $[o_1]$, $[o_1, o_2]$, $[o_1, o_2, o_3]$

(rekursonsfall)

Majoritetsproblemet

Algoritm MAJ.

Indata $[o_1, \ldots, o_n]$

(basfall) $[o_1]$, $[o_1, o_2]$, $[o_1, o_2, o_3]$

(rekursonsfall)

$a := \mathrm{MAJ}[o_1, \ldots, o_{n/2}]$
$b := \mathrm{MAJ}[o_{n/2+1}, \ldots, o_n]$

Majoritetsproblemet

Algoritm MAJ.

Indata $[o_1, \ldots, o_n]$

(basfall) $[o_1]$, $[o_1, o_2]$, $[o_1, o_2, o_3]$

(rekursonsfall)

$a := \mathrm{MAJ}[o_1, \ldots, o_{n/2}]$
$b := \mathrm{MAJ}[o_{n/2+1}, \ldots, o_n]$

om $a \neq$ "nej" är i majoritet i $[o_1, \ldots, o_n]$ så returnera a
om $b \neq$ "nej" är i majoritet i $[o_1, \ldots, o_n]$ så returnera b
annars returnera "nej"

Majoritetsproblemet

Induktion över n.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Ind. ant. Antag att algoritmen är korrekt då $n \leq k$, $k \geq 1$.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Ind. ant. Antag att algoritmen är korrekt då $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Beräkningen av a och b är korrekt pga. ind. ant.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Ind. ant. Antag att algoritmen är korrekt då $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Beräkningen av a och b är korrekt pga. ind. ant.
Om $a =$ "nej" och $b =$ "nej" så kan inget majoritetselement existera. Algoritmen svarar "nej" vilket är korrekt.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Ind. ant. Antag att algoritmen är korrekt då $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Beräkningen av a och b är korrekt pga. ind. ant.
Om $a =$ "nej" och $b =$ "nej" så kan inget majoritetselement existera. Algoritmen svarar "nej" vilket är korrekt.
Om $a \neq$ "nej" så kontrolleras explicit om a är i majoritet. Svaret är sålunda korrekt.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Ind. ant. Antag att algoritmen är korrekt då $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Beräkningen av a och b är korrekt pga. ind. ant.
Om $a =$ "nej" och $b =$ "nej" så kan inget majoritetselement existera. Algoritmen svarar "nej" vilket är korrekt.
Om $a \neq$ "nej" så kontrolleras explicit om a är i majoritet. Svaret är sålunda korrekt.
Om $b \neq$ "nej" så kontrolleras explicit om b är i majoritet. Svaret är sålunda korrekt.

Majoritetsproblemet

Induktion över n.

Bassteg. Om $n \leq 3$ så ger algoritmen korrekt svar via basfallet.

Ind. ant. Antag att algoritmen är korrekt då $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Beräkningen av a och b är korrekt pga. ind. ant.
Om $a =$ "nej" och $b =$ "nej" så kan inget majoritetselement existera. Algoritmen svarar "nej" vilket är korrekt.
Om $a \neq$ "nej" så kontrolleras explicit om a är i majoritet. Svaret är sålunda korrekt.
Om $b \neq$ "nej" så kontrolleras explicit om b är i majoritet. Svaret är sålunda korrekt.
Not. Både $a \neq$ "nej" och $b \neq$ "nej" kan inte gälla samtidigt pga. definitionen av majoritetsproblemet.

Majoritetsproblemet

Algoritm MAJ.

Indata $[o_1, \ldots, o_n]$

(basfall) $[o_1]$, $[o_1, o_2]$, $[o_1, o_2, o_3]$ $O(1)$

(rekursonsfall)

$a := \mathrm{MAJ}[o_1, \ldots, o_{n/2}]$ $T(n/2)$
$b := \mathrm{MAJ}[o_{n/2+1}, \ldots, o_n]$ $T(n/2)$

om $a \neq$ "nej" är i majoritet i $[o_1, \ldots, o_n]$ så ret. a $O(n)$
om $b \neq$ "nej" är i majoritet i $[o_1, \ldots, o_n]$ så ret. b $O(n)$
annars returnera "nej" $O(1)$

Majoritetsproblemet

Summera.

$$T(1) = c$$
$$T(n) = 2T(n/2) + d \cdot n$$

Majoritetsproblemet

Summera.

$$T(1) = c$$
$$T(n) = 2T(n/2) + d \cdot n$$

Lös via Master Theorem.

$$T(n) \in O(n \log n)$$

Exempel 2: Heltalsmultiplikation

Heltalsmultiplikation

Problem. Heltalsmultiplikation.
Indata. Två heltal x och y.
Utdata. $x \cdot y$

Heltalsmultiplikation

Problem. Heltalsmultiplikation.
Indata. Två heltal x och y.
Utdata. $x \cdot y$

Förenklingar:

Antag att både x och y har längd n.

Vi räknar bara multiplikationer i tidskomplexiteten.

Heltalsmultiplikation

Standardmultiplikation.

$$
\begin{array}{r}
18 \\
\cdot\, 96 \\
\hline
10 \cdot 6 + 48 \\
100 \cdot 9 + 10 \cdot 72 \\
\hline
900 + 720 + 60 + 48 \\
\hline
1728
\end{array}
$$

Heltalsmultiplikation

Standardmultiplikation av två n-siffriga tal WX och YZ.

$$
\begin{array}{r}
\mathrm{WX} \\
\cdot\ \mathrm{YZ} \\
\hline
10^{n/2} \cdot \mathrm{ZW} + \mathrm{ZX} \\
10^{n} \cdot \mathrm{YW} + 10^{n/2} \cdot \mathrm{YX} \\
\hline
10^{n} \cdot \mathrm{YW} + 10^{n/2} \cdot \mathrm{YX} + 10^{n/2} \cdot \mathrm{ZW} + \mathrm{ZX}
\end{array}
$$

Heltalsmultiplikation

Förslag på rekursiv algoritm M.

Beräkna $M(\mathrm{Y}, \mathrm{W})$, $M(\mathrm{Y}, \mathrm{X})$, $M(\mathrm{Z}, \mathrm{W})$, $M(\mathrm{Z}, \mathrm{X})$.
Justera tio-potens-faktorer.
Addera.

Heltalsmultiplikation

Tidskomplexitet.

$$T(1) = 1$$
$$T(n) = 4\,T(n/2) + c \cdot n$$

Heltalsmultiplikation

Tidskomplexitet.

$$T(1) = 1$$
$$T(n) = 4\,T(n/2) + c \cdot n$$

$$T(n) \in O(n^2)$$

Heltalsmultiplikation

Standardmultiplikation av två n-siffriga tal WX och YZ.

$$
\begin{array}{r}
\mathrm{WX} \\
\cdot\,\mathrm{YZ} \\
\hline
10^{n/2} \cdot \mathrm{ZW} + \mathrm{ZX} \\
10^{n} \cdot \mathrm{YW} + 10^{n/2} \cdot \mathrm{YX} \\
\hline
10^{n} \cdot \mathrm{YW} + 10^{n/2} \cdot (\mathrm{YX} + \mathrm{ZW}) + \mathrm{ZX}
\end{array}
$$

Heltalsmultiplikation

$$p = \mathrm{Y} \cdot \mathrm{W}$$

$$q = \mathrm{Z} \cdot \mathrm{X}$$

$$r = (\mathrm{W} + \mathrm{X}) \cdot (\mathrm{Y} + \mathrm{Z})$$

Heltalsmultiplikation

$$p = \mathrm{Y} \cdot \mathrm{W}$$

$$q = \mathrm{Z} \cdot \mathrm{X}$$

$$r = (\mathrm{W} + \mathrm{X}) \cdot (\mathrm{Y} + \mathrm{Z})$$

$$r - p - q = \mathrm{YX} + \mathrm{ZW}$$

Heltalsmultiplikation

Förslag på ny rekursiv algoritm M.

Beräkna $M(\mathrm{Y}, \mathrm{W})$, $M(\mathrm{W} + \mathrm{X}, \mathrm{Y} + \mathrm{Z})$, $M(\mathrm{Z}, \mathrm{X})$.
Beräkna $\mathrm{YX} + \mathrm{ZW}$
Justera tio-potens-faktorer.
Addera.

Heltalsmultiplikation

Tidskomplexitet.

$$T(1) = 1$$
$$T(n) = 3\,T(n/2) + c \cdot n$$

Heltalsmultiplikation

Tidskomplexitet.

$$T(1) = 1$$
$$T(n) = 3T(n/2) + c \cdot n$$

$$T(n) \in O(n^{\log_2 3}) \subseteq O(n^{1.585})$$

Exempel 1: 1/2-3SAT

1/2-3SAT

Problem. 1/2-3SAT.
Indata. Propositionslogisk formel $F = C_1 \wedge \cdots \wedge C_m$ på 3-CNF.
Utdata. En tilldelning som satisfierar minst hälften av klausulerna.

1/2-3SAT

Problem. 1/2-3SAT.
Indata. Propositionslogisk formel $F = C_1 \wedge \cdots \wedge C_m$ på 3-CNF.
Utdata. En tilldelning som satisfierar minst hälften av klausulerna.

$(p \vee \neg q \vee r) \wedge (s \vee t) \wedge (\neg s \vee \neg q \vee \neg r) \wedge (t)$

Tomma klausuler får inte förekomma.

1/2-3SAT

Bevisa att varje 3CNF-formel har en tilldelning som satisfierar minst hälften av klausulerna.

1/2-3SAT

Bevisa att varje 3CNF-formel har en tilldelning som satisfierar minst hälften av klausulerna.

Låt $F = C_1 \wedge \cdots \wedge C_m$ vara en 3-CNF-formel med variabler $x_1, \ldots, x_n$.

1/2-3SAT

Bevisa att varje 3CNF-formel har en tilldelning som satisfierar minst hälften av klausulerna.

Låt $F = C_1 \wedge \cdots \wedge C_m$ vara en 3-CNF-formel med variabler $x_1, \ldots, x_n$.

Induktion över n.

1/2-3SAT

Bevisa att varje 3CNF-formel har en tilldelning som satisfierar minst hälften av klausulerna.

Låt $F = C_1 \wedge \cdots \wedge C_m$ vara en 3-CNF-formel med variabler $x_1, \ldots, x_n$.

Induktion över n.

Bassteg. $n = 1$.

$(x_1) \vee (\neg x_1 \vee \neg x_1) \vee (x_1 \vee \neg x_1) \vee (x_1 \vee x_1 \vee x_1)$

1/2-3SAT

Bevisa att varje 3CNF-formel har en tilldelning som satisfierar minst hälften av klausulerna.

Låt $F = C_1 \wedge \cdots \wedge C_m$ vara en 3-CNF-formel med variabler $x_1, \ldots, x_n$.

Induktion över n.

Bassteg. $n = 1$.

$(x_1) \vee (\neg x_1 \vee \neg x_1) \vee (x_1 \vee \neg x_1) \vee (x_1 \vee x_1 \vee x_1)$

Varje klausul satisfieras av minst en av $f_0(x_1) = 0$ och $f_1(x_1) = 1$ eftersom inga klausuler får vara tomma.

1/2-3SAT

Bevisa att varje 3CNF-formel har en tilldelning som satisfierar minst hälften av klausulerna.

Låt $F = C_1 \wedge \cdots \wedge C_m$ vara en 3-CNF-formel med variabler $x_1, \ldots, x_n$.

Induktion över n.

Bassteg. $n = 1$.

$(x_1) \vee (\neg x_1 \vee \neg x_1) \vee (x_1 \vee \neg x_1) \vee (x_1 \vee x_1 \vee x_1)$

Varje klausul satisfieras av minst en av $f_0(x_1) = 0$ och $f_1(x_1) = 1$ eftersom inga klausuler får vara tomma.

En av de två tilldelningarna f_0 och f_1 satisfierar minst hälften av klausulerna.

1/2-3SAT

Ind. ant. Påståendet gäller för $n \leq k$, $k \geq 1$.

1/2-3SAT

Ind. ant. Påståendet gäller för $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Välj godtyckligt en variabel x i F.

1/2-3SAT

Ind. ant. Påståendet gäller för $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Välj godtyckligt en variabel x i F.

Dela formeln i två delar $F = F^+ \wedge F^-$:

F^+ innehåller alla klausuler i F som innehåller x.
F^- innehåller alla klausuler i F som inte innehåller x.

1/2-3SAT

Ind. ant. Påståendet gäller för $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Välj godtyckligt en variabel x i F.

Dela formeln i två delar $F = F^+ \wedge F^-$:

F^+ innehåller alla klausuler i F som innehåller x.
F^- innehåller alla klausuler i F som inte innehåller x.

Minst hälften av klausulerna i F^+ är satisfierbara genom att ge x ett lämpligt värde (på samma sätt som i bassteget).

1/2-3SAT

Ind. ant. Påståendet gäller för $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Välj godtyckligt en variabel x i F.

Dela formeln i två delar $F = F^+ \wedge F^-$:

F^+ innehåller alla klausuler i F som innehåller x.
F^- innehåller alla klausuler i F som inte innehåller x.

Minst hälften av klausulerna i F^+ är satisfierbara genom att ge x ett lämpligt värde (på samma sätt som i bassteget).

Minst hälften av klausulerna i F^- är satisfierbara enligt induktionsantagandet.

1/2-3SAT

Ind. ant. Påståendet gäller för $n \leq k$, $k \geq 1$.

Ind. steg. Antag $n = k + 1$. Välj godtyckligt en variabel x i F.

Dela formeln i två delar $F = F^+ \wedge F^-$:

F^+ innehåller alla klausuler i F som innehåller x.
F^- innehåller alla klausuler i F som inte innehåller x.

Minst hälften av klausulerna i F^+ är satisfierbara genom att ge x ett lämpligt värde (på samma sätt som i bassteget).

Minst hälften av klausulerna i F^- är satisfierbara enligt induktionsantagandet.

Slutsats: Minst hälften av klausulerna i $F = F^+ \wedge F^-$ är satisfierbara.

1/2-3SAT

Transformera beviset till en algoritm.

1/2-3SAT

Algoritm $X(F)$

Antag F innehåller n variabler.

Välj godtyckligt en variabel x i F.

Om $n = 1$ så returnera antingen $f_0(x)$ eller $f_1(x)$ beroende på vilken som satisfierar flest klausuler.

1/2-3SAT

Algoritm $X(F)$

Antag F innehåller n variabler.

Välj godtyckligt en variabel x i F.

Om $n = 1$ så returnera antingen $f_0(x)$ eller $f_1(x)$ beroende på vilken som satisfierar flest klausuler.

Om $n > 1$ så dela F i F^+, F^-.

Låt $f = X(F^-)$.

Låt g vara lika med $f_0(x)$ eller $f_1(x)$ beroende på vilken som satisfierar flest klausuler i F^+.

Returnera "kombinationen" av f och g.

1/2-3SAT

Tidskomplexitet.

$$T(1) = \text{poly}(||F||)$$
$$T(n) = T(n-1) + \text{poly}(||F||)$$

1/2-3SAT

Tidskomplexitet.

$$T(1) = \text{poly}(||F||)$$
$$T(n) = T(n-1) + \text{poly}(||F||)$$

Formeln F är på 3-CNF så den kan innehålla högst $O(n^3)$ klausuler och sålunda maximalt ha längd $O(n^3)$.

1/2-3SAT

Tidskomplexitet.

$$T(1) = \text{poly}(||F||)$$
$$T(n) = T(n-1) + \text{poly}(||F||)$$

Formeln F är på 3-CNF så den kan innehålla högst $O(n^3)$ klausuler och sålunda maximalt ha längd $O(n^3)$.

$$T(1) = \text{poly}(n)$$
$$T(n) = T(n-1) + \text{poly}(n)$$

1/2-3SAT

Tidskomplexitet.

$$T(1) = \text{poly}(||F||)$$
$$T(n) = T(n-1) + \text{poly}(||F||)$$

Formeln F är på 3-CNF så den kan innehålla högst $O(n^3)$ klausuler och sålunda maximalt ha längd $O(n^3)$.

$$T(1) = \text{poly}(n)$$
$$T(n) = T(n-1) + \text{poly}(n)$$

$$T(n) \in \text{poly}(n).$$