I. Grundläggande metoder för algoritmkonstruktion.

II. NP-fullständighet.

III. Inexakta metoder.

Fö. 2: Giriga algoritmer.

Fö. 3: Rekursiva algoritmer.

Fö. 4: Dynamisk programmering.

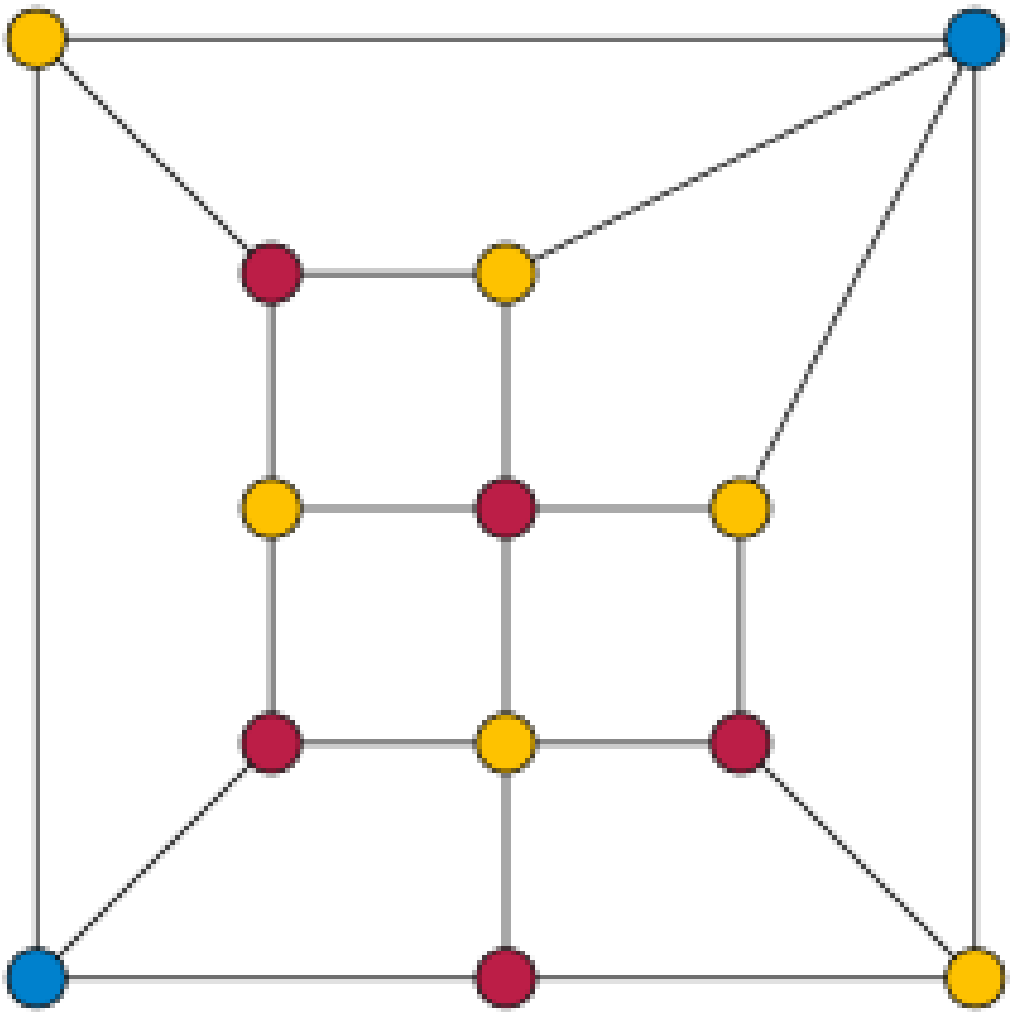
Giriga algoritmer

Exempel 1: 2-Färgning av Oriktad Graf

2-Färgning

En *k-färgning* av en oriktad graf $G = (V, E)$ är en funktion $f : V \to \{1, \ldots, k\}$ sådan att för varje båge $\{v, w\} \in E$ så gäller $f(v) \neq f(w)$.

2-Färgning



2-Färgning

Problem. 2-Färgning.
Indata. Oriktad graf $G = (V, E)$.
Fråga. Är G 2-färgbar?

2-Färgning

Antag att grafen $G = (V, E)$ är sammanhängande.

Välj en nod $v \in V$ godtyckligt.

Ge v färgen 0.

Ge alla dess grannar färgen 1.

Uppstår konflikt så svara "nej".

1. Om alla noder färgade så svara "ja".

Välj godtyckligt en nod $v \in V$ som har en färg a och en icke-färgad granne..

Ge de icke-färgade grannarna färgen $1 - a$..

Uppstår konflikt så svara "nej".

Gå till **1.**

2-Färgning

Antag att grafen $G = (V, E)$ är sammanhängande.

Välj en nod $v \in V$ godtyckligt.

Ge v färgen 0.

Ge alla dess grannar färgen 1.

Uppstår konflikt så svara "nej".

1. Om alla noder färgade så svara "ja".

Välj godtyckligt en nod $v \in V$ som har en färg a och en icke-färgad granne..

Ge de icke-färgade grannarna färgen $1 - a$..

Uppstår konflikt så svara "nej".

Gå till **1.**

2-Färgning

Korrekthetsbevis

Vad händer om algoritmen svara "ja"?

Alla noder har fått en färg och ingen konflikt har uppstått.

Slutsats: Grafen är 2-färgbar.

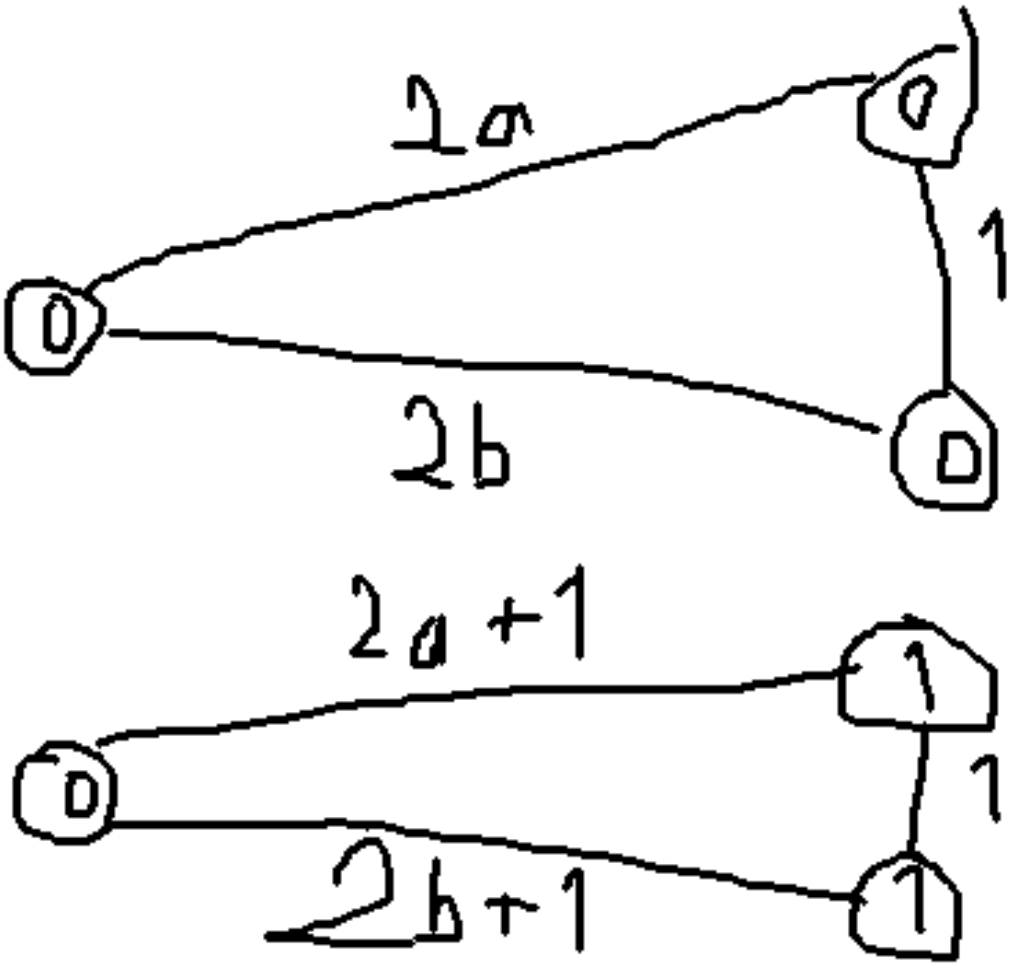
2-Färgning

Korrekthetsbevis

Vad händer om algoritmen svara "nej"?

En konflikt har uppstått.

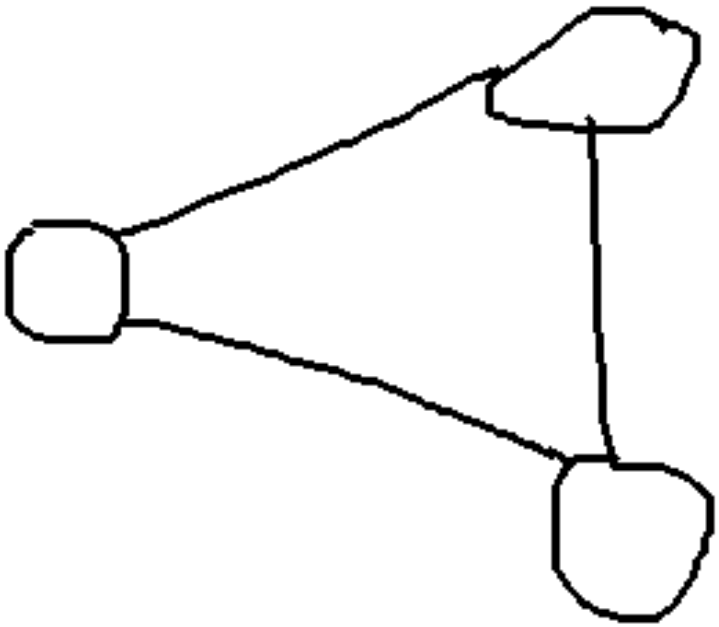
2-Färgning



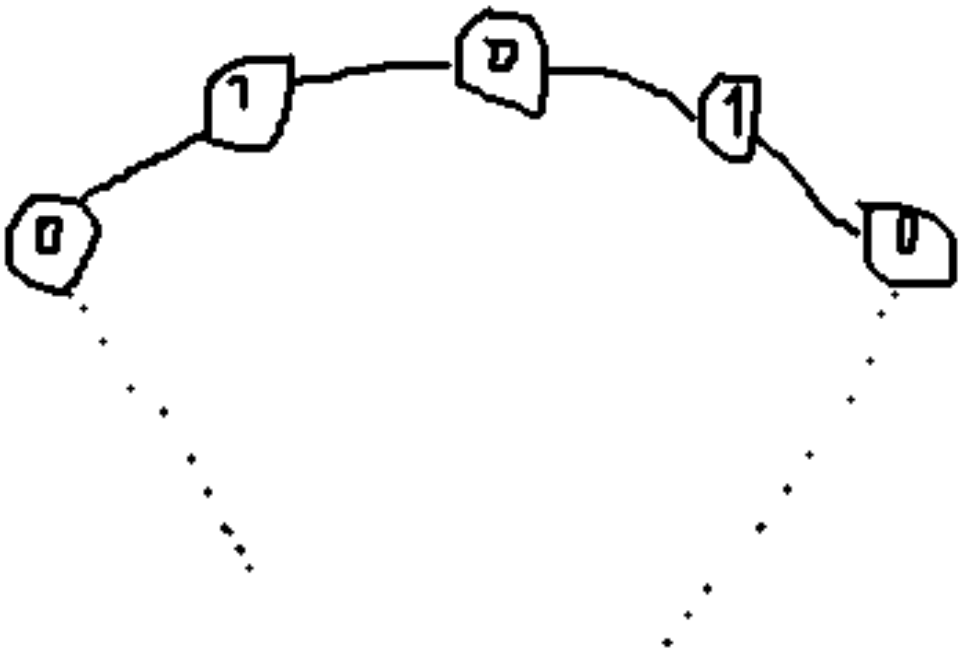
2-Färgning

Slutsats: Grafen innehåller en cykel av udda längd.

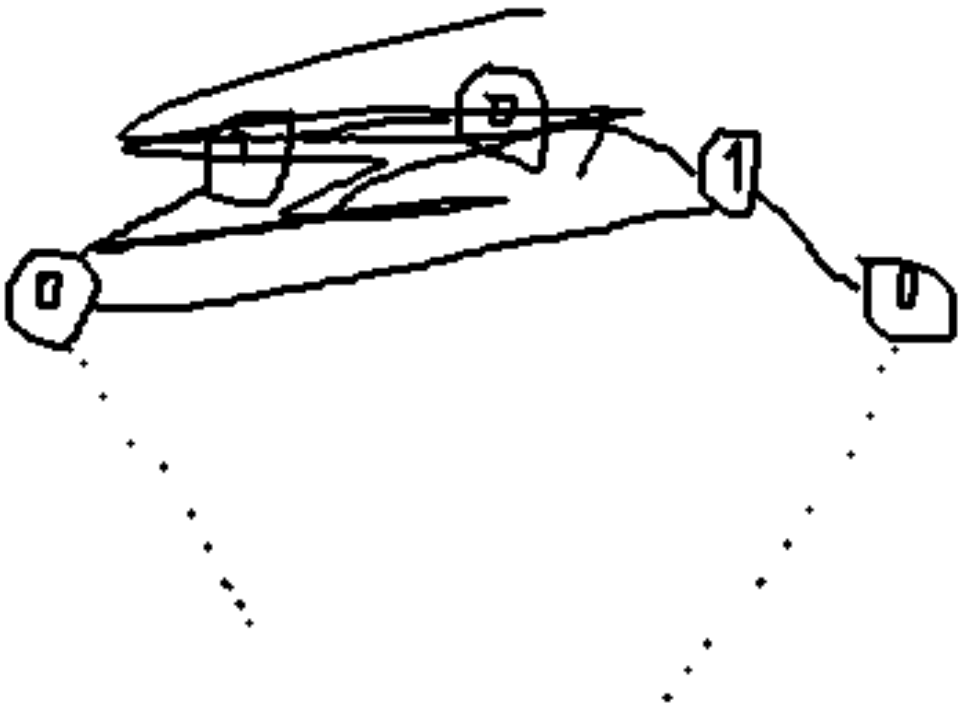
2-Färgning



2-Färgning



2-Färgning



2-Färgning

De grundläggande grafoperationerna som behövs går i polynomisk tid.

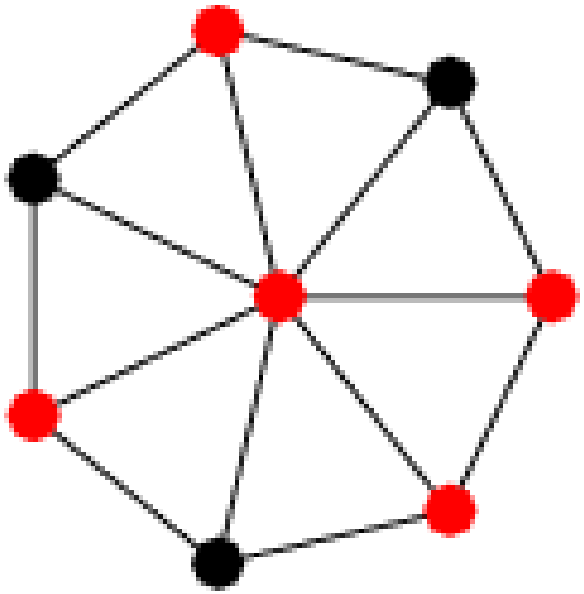
Algoritmen går i polynomisk tid.

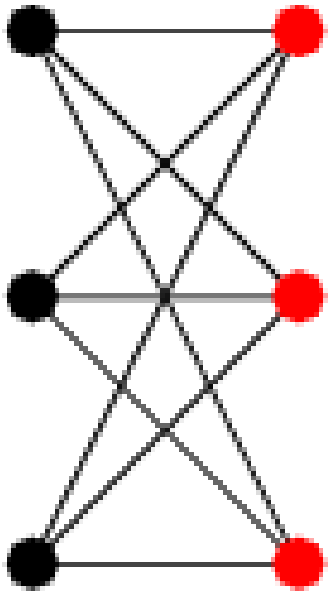
Exempel 2: Nodhölje

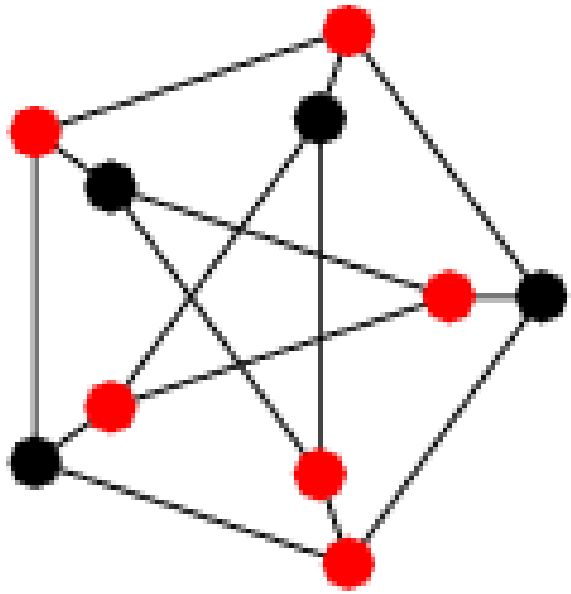
Nodhölje

Ett *nodhölje* i en oriktad graf $G = (V, E)$ är en delmängd $V' \subseteq V$ sådan att varje båge i E är ansluten till minst en nod i V'.

Nodhölje







Nodhölje

Problem. Nodhölje.
Indata. Oriktad graf $G = (V, E)$.
Utdata. Nodhölje $V' \subseteq V$ med minimal storlek.

Polynomisk algoritm finns (troligen) inte.

Konstruera polynomisk algoritm för *träd*.

Ett träd är en oriktad graf utan cykler.

Nodhölje

Problem. Nodhölje.
Indata. Oriktad graf $G = (V, E)$.
Utdata. Nodhölje $V' \subseteq V$ med minimal storlek.

Polynomisk algoritm finns (troligen) inte.

Konstruera polynomisk algoritm för *träd*.

Ett träd är en oriktad graf utan cykler.

Nodhölje

Problem. Nodhölje.
Indata. Oriktad graf $G = (V, E)$.
Utdata. Nodhölje $V' \subseteq V$ med minimal storlek.

Polynomisk algoritm finns (troligen) inte.

Konstruera polynomisk algoritm för *träd*.

Ett träd är en oriktad graf utan cykler.

Nodhölje

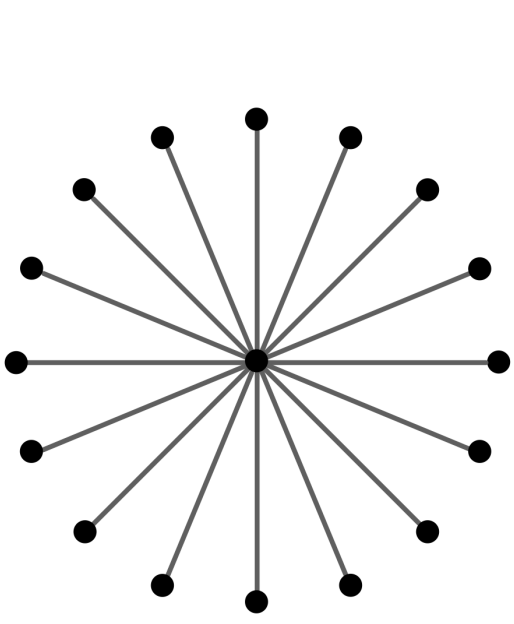
Problem. Nodhölje.
Indata. Oriktad graf $G = (V, E)$.
Utdata. Nodhölje $V' \subseteq V$ med minimal storlek.

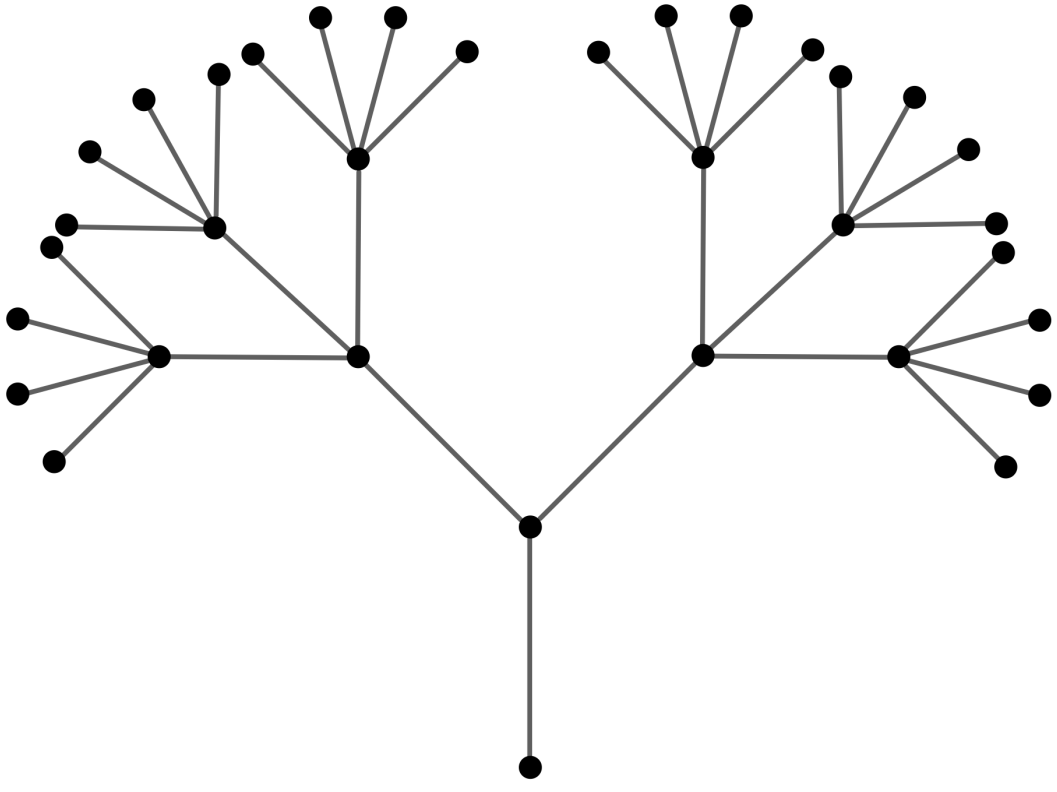
Polynomisk algoritm finns (troligen) inte.

Konstruera polynomisk algoritm för *träd*.

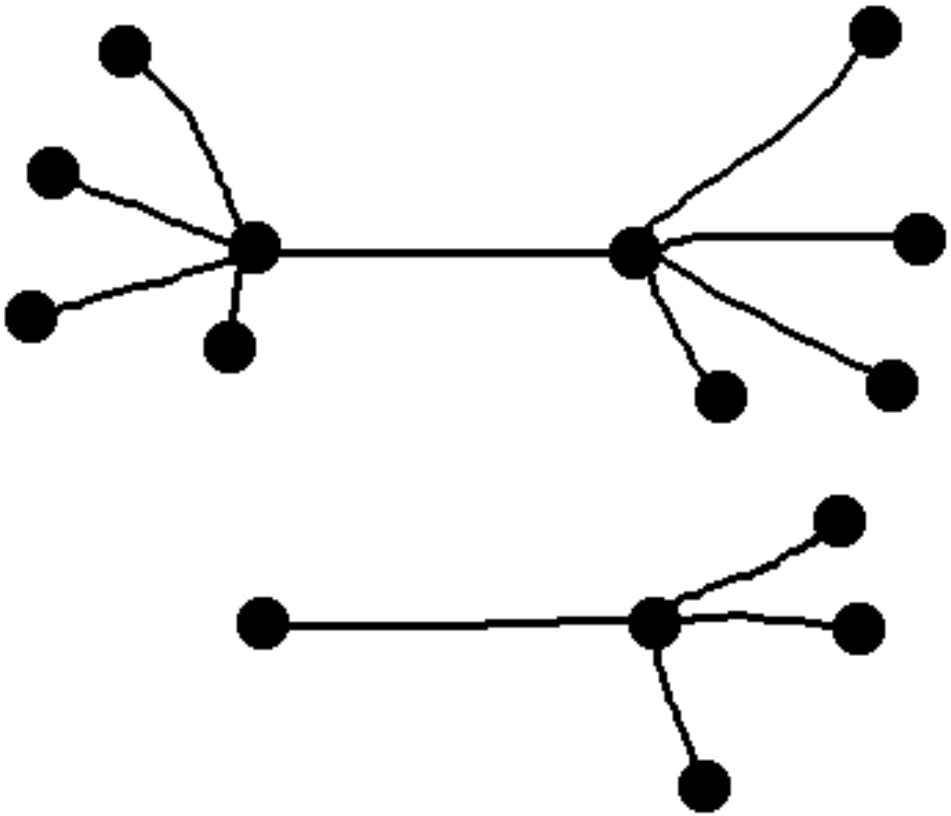
Ett träd är en oriktad graf utan cykler.

Nodhölje

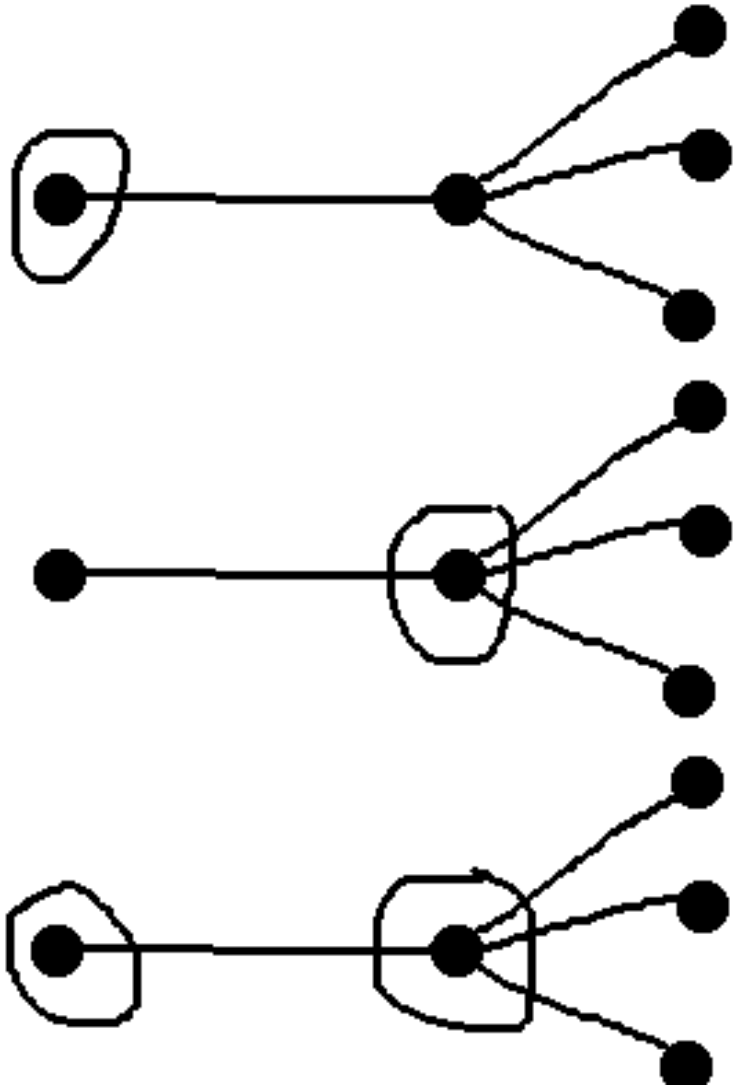




Nodhölje



Nodhölje



Nodhölje

Valet i mitten är *alltid* det bästa.

Mervärdesprincipen

Nodhölje

Låt $V' = \emptyset$

Välj ett godtyckligt löv L i trädet.

Lägg lövets unika granne L' i V'.

Tag bort L' från trädet.

Tag bort alla bågar som täcks av L'.

Tag bort alla isolerade noder i trädet.

Repetera tills inga bågar finns kvar.

Nodhölje

De grundläggande grafoperationerna som behövs går i polynomisk tid.

Algoritmen går i polynomisk tid.

Exempel 3: Punkttäckning

Punkttäckning

Låt $J = [a, b]$ vara ett intervall på tallinjen.

Intervallet J *täcker* en punkt p om $a \leq p \leq b$.

Punkttäckning

Problem. Punkttäckning.
Indata. En mängd $\mathbf{P}$ med punkter och en mängd $\mathbf{I}$ med intervall.
Utdata. En mängd $\mathbf{I}' \subseteq \mathbf{I}$ som täcker alla punkter i $\mathbf{P}$ och är så liten som möjligt.

Vi antar att de indata vi tittar på alltid har en lösning.

Punkttäckning

Problem. Punkttäckning.
Indata. En mängd $\mathbf{P}$ med punkter och en mängd $\mathbf{I}$ med intervall.
Utdata. En mängd $\mathbf{I}' \subseteq \mathbf{I}$ som täcker alla punkter i $\mathbf{P}$ och är så liten som möjligt.

Vi antar att de indata vi tittar på alltid har en lösning.

Punkttäckning

Algoritm A.

Låt $\mathbf{I}' = \emptyset$.

Välj $I \in \mathbf{I}$ som täcker den *minsta* punkten i $\mathbf{P}$ och så många ytterligare punkter i $\mathbf{P}$ som möjligt.

Lägg intervallet I i $\mathbf{I}'$.

Tag bort alla punkter som täcks av I.

Repetera tills inga punkter finns kvar att täcka.

Punkttäckning

Algoritm A kan (enkelt) implementeras så att den går i polynomisk tid.

Algoritm A producerar alltid en giltig lösning.

Visa att lösningen är optimal.

Punkttäckning

Antag att A inte alltid genererar en optimal lösning.

Då finns en instans $X = (\mathbf{P}, \mathbf{I})$ sådan att $|OPT(X)| < |A(X)|$.

Välj X så att $\mathbf{P}$ är så liten som möjligt.

Punkttäckning

Antag att A inte alltid genererar en optimal lösning.

Då finns en instans $X = (\mathbf{P}, \mathbf{I})$ sådan att $|OPT(X)| < |A(X)|$.

Välj X så att $\mathbf{P}$ är så liten som möjligt.

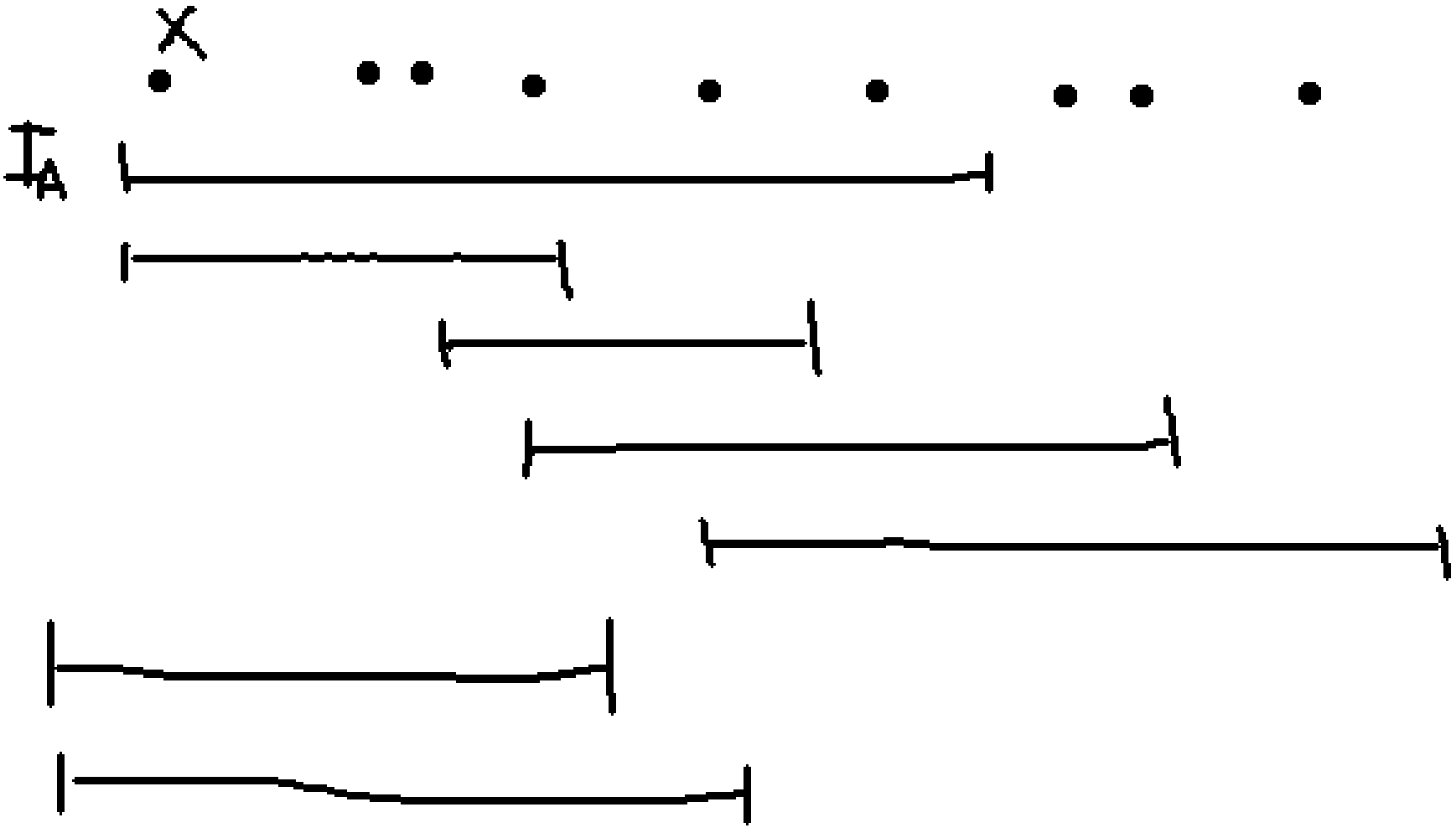
Punkttäckning

Antag att A inte alltid genererar en optimal lösning.

Då finns en instans $X = (\mathbf{P}, \mathbf{I})$ sådan att $|OPT(X)| < |A(X)|$.

Välj X så att $\mathbf{P}$ är så liten som möjligt.

Punkttäckning



Punkttäckning

Den minsta punkten är x.

Algoritm A täcker den genom att utnyttja intervallet I_A.

I den optimal lösningen täcks x med intervallen $I_B^1, \ldots, I_B^p$.

Punkttäckning

Bilda instansen $X' = (\mathbf{P}', \mathbf{I})$ där alla punkter som täcks av I_A har tagits bort.

$|A(X')| = |A(X)| - 1$

$|OPT(X')| < |OPT(X)| < |A(X)| = |A(X')| + 1$

Punkttäckning

Bilda instansen $X' = (\mathbf{P}', \mathbf{l})$ där alla punkter som täcks av l_A har tagits bort.

$|A(X')| = |A(X)| - 1$

$|OPT(X')| < |OPT(X)| < |A(X)| = |A(X')| + 1$

Punkttäckning

Bilda instansen $X' = (\mathbf{P}', \mathbf{I})$ där alla punkter som täcks av I_A har tagits bort.

$|A(X')| = |A(X)| - 1$

$|OPT(X')| < |OPT(X)| < |A(X)| = |A(X')| + 1$

Punkttäckning

$|A(X)| - |OPT(X')| \geq 2$ (vi jobbar med heltal).

$|A(X')| + 1 - |OPT(X')| \geq 2$ (omskrivning).

X' innehåller färre punkter än X så $|A(X')| = |OPT(X')|$.

$|A(X')| + 1 - |A(X')| \geq 2$

$1 \geq 2$

Motsägelse.

Punkttäckning

$|A(X)| - |OPT(X')| \geq 2$ (vi jobbar med heltal).

$|A(X')| + 1 - |OPT(X')| \geq 2$ (omskrivning).

X' innehåller färre punkter än X så $|A(X')| = |OPT(X')|$.

$|A(X')| + 1 - |A(X')| \geq 2$

$1 \geq 2$

Motsägelse.

Punkttäckning

$|A(X)| - |OPT(X')| \geq 2$ (vi jobbar med heltal).

$|A(X')| + 1 - |OPT(X')| \geq 2$ (omskrivning).

X' innehåller färre punkter än X så $|A(X')| = |OPT(X')|$.

$|A(X')| + 1 - |A(X')| \geq 2$

$1 \geq 2$

Motsägelse.

Punkttäckning

$|A(X)| - |OPT(X')| \geq 2$ (vi jobbar med heltal).

$|A(X')| + 1 - |OPT(X')| \geq 2$ (omskrivning).

X' innehåller färre punkter än X så $|A(X')| = |OPT(X')|$.

$|A(X')| + 1 - |A(X')| \geq 2$

$1 \geq 2$

Motsägelse.

Punkttäckning

$|A(X)| - |OPT(X')| \geq 2$ (vi jobbar med heltal).

$|A(X')| + 1 - |OPT(X')| \geq 2$ (omskrivning).

X' innehåller färre punkter än X så $|A(X')| = |OPT(X')|$.

$|A(X')| + 1 - |A(X')| \geq 2$

$1 \geq 2$

Motsägelse.

Punkttäckning

$|A(X)| - |OPT(X')| \geq 2$ (vi jobbar med heltal).

$|A(X')| + 1 - |OPT(X')| \geq 2$ (omskrivning).

X' innehåller färre punkter än X så $|A(X')| = |OPT(X')|$.

$|A(X')| + 1 - |A(X')| \geq 2$

$1 \geq 2$

Motsägelse.