Øvingsforelesning 11

TDT4120 - Algoritmer og datastrukturer

Øving 10

Korteste vei fra én til alle

Oppgave 2: Hva går korteste-vei-problemet (fra én til alle) ut på?

Korteste vei fra én til alle

Oppgave 2: Hva går korteste-vei-problemet (fra én til alle) ut på?

Finne stier som minimerer summen av kantvektene.

Korteste vei fra én til alle

Oppgave 2: Hva går korteste-vei-problemet (fra én til alle) ut på?

Oppgave 3: Hvilke påstander er korrekte?

- Selv om man har korteste vei fra node A til B og korteste vei fra node B til C vil man ikke kunne garantere at det finnes en korteste vei mellom A og C som går via B.
- Korteste vei har ikke optimal substruktur om vi har positive kantvekter.
- Ved å snu alle kantene i en graf kan man finne korteste veier fra alle til én, gitt at man vet hvordan man finner korteste veier fra én til alle.
- Det finnes aldri mer enn én korteste vei mellom to noder.
- DFS vil finne korteste vei til alle noder i en graf.

Korteste vei fra én til alle

Oppgave 2: Hva går korteste-vei-problemet (fra én til alle) ut på?

Oppgave 3: Hvilke påstander er korrekte?

- Selv om man har korteste vei fra node A til B og korteste vei fra node B til C vil man ikke kunne garantere at det finnes en korteste vei mellom A og C som går via B.
- ~~Korteste vei har ikke optimal substruktur om vi har positive kantvekter.~~
- Ved å snu alle kantene i en graf kan man finne korteste veier fra alle til én, gitt at man vet hvordan man finner korteste veier fra én til alle.
- Det finnes aldri mer enn én korteste vei mellom to noder.
- DFS vil finne korteste vei til alle noder i en graf.

Korteste vei fra én til alle

Oppgave 2: Hva går korteste-vei-problemet (fra én til alle) ut på?

Oppgave 3: Hvilke påstander er korrekte?

- Selv om man har korteste vei fra node A til B og korteste vei fra node B til C vil man ikke kunne garantere at det finnes en korteste vei mellom A og C som går via B.
- ~~Korteste vei har ikke optimal substruktur om vi har positive kantvekter.~~
- Ved å snu alle kantene i en graf kan man finne korteste veier fra alle til én, gitt at man vet hvordan man finner korteste veier fra én til alle.
- ~~Det finnes aldri mer enn én korteste vei mellom to noder.~~
- DFS vil finne korteste vei til alle noder i en graf.

Korteste vei fra én til alle

Oppgave 2: Hva går korteste-vei-problemet (fra én til alle) ut på?

Oppgave 3: Hvilke påstander er korrekte?

- Selv om man har korteste vei fra node A til B og korteste vei fra node B til C vil man ikke kunne garantere at det finnes en korteste vei mellom A og C som går via B.
- ~~Korteste vei har ikke optimal substruktur om vi har positive kantvekter.~~
- Ved å snu alle kantene i en graf kan man finne korteste veier fra alle til én, gitt at man vet hvordan man finner korteste veier fra én til alle.
- ~~Det finnes aldri mer enn én korteste vei mellom to noder.~~
- ~~DFS vil finne korteste vei til alle noder i en graf.~~

Korteste vei

Oppgave 4: Hva har vi hvis en av de korteste veiene mellom A og C går igjennom B?

Korteste vei

Oppgave 4: Hva har vi hvis en av de korteste veiene mellom A og C går igjennom B?

Den korteste veien mellom A og B og mellom B og C.

Korteste vei

Oppgave 4: Hva har vi hvis en av de korteste veiene mellom A og C går igjennom B?

Den korteste veien mellom A og B og mellom B og C.

Oppgave 5: Hvorfor ser vi ikke på korteste vei fra én til én i pensum?

Korteste vei

Oppgave 4: Hva har vi hvis en av de korteste veiene mellom A og C går igjennom B?

Den korteste veien mellom A og B og mellom B og C.

Oppgave 5: Hvorfor ser vi ikke på korteste vei fra én til én i pensum?

Vi kjenner ikke til algoritmer som er bedre i verste tilfellet enn for én til alle.

Korteste vei fra én til alle

Oppgave 6: Hvilke påstander er korrekte?

- Dijkstra finner alltid korteste vei med både negative kanter og negative sykler.
- Dijkstra finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Bellman-Ford finner alltid korteste vei med både negative kanter og negative sykler.
- Bellman-Ford finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Vi kan fjerne 0-vektede sykler fra en sti for å lage en ny sti med samme vekt.
- Det er umulig å detektere om en graf har negative sykler.

Korteste vei fra én til alle

Oppgave 6: Hvilke påstander er korrekte?

- ~~Dijkstra finner alltid korteste vei med både negative kanter og negative sykler.~~
- Dijkstra finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Bellman-Ford finner alltid korteste vei med både negative kanter og negative sykler.
- Bellman-Ford finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Vi kan fjerne 0-vektede sykler fra en sti for å lage en ny sti med samme vekt.
- Det er umulig å detektere om en graf har negative sykler.

Korteste vei fra én til alle

Oppgave 6: Hvilke påstander er korrekte?

- ~~Dijkstra finner alltid korteste vei med både negative kanter og negative sykler.~~
- ~~Dijkstra finner alltid korteste vei med negative kanter, men ikke negative sykler.~~
- Bellman-Ford finner alltid korteste vei med både negative kanter og negative sykler.
- Bellman-Ford finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Vi kan fjerne 0-vektede sykler fra en sti for å lage en ny sti med samme vekt.
- Det er umulig å detektere om en graf har negative sykler.

Korteste vei fra én til alle

Oppgave 6: Hvilke påstander er korrekte?

- ~~Dijkstra finner alltid korteste vei med både negative kanter og negative sykler.~~
- ~~Dijkstra finner alltid korteste vei med negative kanter, men ikke negative sykler.~~
- ~~Bellman-Ford finner alltid korteste vei med både negative kanter og negative sykler.~~
- Bellman-Ford finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Vi kan fjerne 0-vektede sykler fra en sti for å lage en ny sti med samme vekt.
- Det er umulig å detektere om en graf har negative sykler.

Korteste vei fra én til alle

Oppgave 6: Hvilke påstander er korrekte?

- ~~Dijkstra finner alltid korteste vei med både negative kanter og negative sykler.~~
- ~~Dijkstra finner alltid korteste vei med negative kanter, men ikke negative sykler.~~
- ~~Bellman-Ford finner alltid korteste vei med både negative kanter og negative sykler.~~
- Bellman-Ford finner alltid korteste vei med negative kanter, men ikke negative sykler.
- Vi kan fjerne 0-vektede sykler fra en sti for å lage en ny sti med samme vekt.
- ~~Det er umulig å detektere om en graf har negative sykler.~~

Relax

Oppgave 7: $u.d = 3, v.d = 7, w(u, v) = 3$. Hva blir $u.d$ og $v.d$ etter $\textsc{Relax}(u, v, w)$?

Oppgave 8: Hva blir $u.\pi$ og $v.\pi$ i denne situasjonen?

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2        v.d = u.d + w(u, v)
3        v.π = u
```

Relax

Oppgave 7: $u.d = 3, v.d = 7, w(u, v) = 3$. Hva blir $u.d$ og $v.d$ etter $\textsc{Relax}(u, v, w)$?

Oppgave 8: Hva blir $u.\pi$ og $v.\pi$ i denne situasjonen?

```
Relax(u, v, w)
1  if v.d > u.d + w(u, v)
2        v.d = u.d + w(u, v)
3        v.π = u
```

Før: $u.d = 3, u.\pi = ?$ og $v.d = 7, v.\pi = ?$

Relax

Oppgave 7: $u.d = 3, v.d = 7, w(u, v) = 3$. Hva blir $u.d$ og $v.d$ etter $\textsc{Relax}(u, v, w)$?

Oppgave 8: Hva blir $u.\pi$ og $v.\pi$ i denne situasjonen?

$\textsc{Relax}(u, v, w)$
1 **if** $v.d > u.d + w(u, v)$
2 $\quad v.d = u.d + w(u, v)$
3 $\quad v.\pi = u$

Før: $u.d = 3, u.\pi = ?$ og $v.d = 7, v.\pi = ?$

Etter: $u.d = 3, u.\pi = ?$ og $v.d = 6, v.\pi = u$

Mulige verdier for distansen mellom to noder

Oppgave 9: Hvilke mulige verdier kan $a.d$ ha for en node a etter å ha kjørt en av pensum algoritmene som løser korteste-vei-problemet, fra én til alle?

Mulige verdier for distansen mellom to noder

Oppgave 9: Hvilke mulige verdier kan $a.d$ ha for en node a etter å ha kjørt en av pensum algoritmene som løser korteste-vei-problemet, fra én til alle?

INITIALIZE-SINGLE-SOURCE setter verdiene til 0 og ∞.

Mulige verdier for distansen mellom to noder

Oppgave 9: Hvilke mulige verdier kan $a.d$ ha for en node a etter å ha kjørt en av pensum algoritmene som løser korteste-vei-problemet, fra én til alle?

INITIALIZE-SINGLE-SOURCE setter verdiene til 0 og ∞.

RELAX endrer verdien til $u.d + w(u, a)$ for en annen node u.

Mulige verdier for distansen mellom to noder

Oppgave 9: Hvilke mulige verdier kan $a.d$ ha for en node a etter å ha kjørt en av pensum algoritmene som løser korteste-vei-problemet, fra én til alle?

INITIALIZE-SINGLE-SOURCE setter verdiene til 0 og ∞.

RELAX endrer verdien til $u.d + w(u, a)$ for en annen node u.

Kan få alle verdier i $\mathbb{R}$ og ∞ og lengden til den korteste stien mellom s og a.

Byggetid - Programmering

Oppgave 10: Du ønsker å finne korteste tid det tar å utføre et byggeprosjekt hvor hver oppgave tar en gitt tid og kan være avhengig av andre fullførelsen av andre oppgaver. Oppgaver kan fullføres i parallell hvis de ikke er avhengig av hverandre.

Byggetid - Programmering

Oppgave 10: Du ønsker å finne korteste tid det tar å utføre et byggeprosjekt hvor hver oppgave tar en gitt tid og kan være avhengig av andre fullførelsen av andre oppgaver. Oppgaver kan fullføres i parallell hvis de ikke er avhengig av hverandre.

BUILDING-TIMES(*tasks*)
1 create a dependency graph G
2 $\ell = $ TOPOLOGICAL-SORT(G)
3 iterate ℓ and let $v.f$ be the max of its parents + its own time
4 **return** *max*$(\{v.f | v \in \mathrm{G.V}\})$

DAG-Shortest-Path kontra Dijkstra

Oppgave 11: Hvilke utsagn om fordeler ved å bruke DAG-Shortest-Path over Dijkstra stemmer?

- Dijkstra fungerer ikke på rettede asykliske grafer, selv om de har positive kantvekter.
- DAG-Shortest-Path fungerer også med negative kantvekter, mens Dijkstra gjør ikke det.
- DAG-Shortest-Path har bedre asymptotisk tidskompleksitet enn Dijkstra i verste tilfelle om $|\mathrm{E}| = o(\mathrm{V} \lg \mathrm{V})$.
- DAG-Shortest-Path har bedre asymptotisk tidskompleksitet enn Dijkstra i verste tilfelle uansett hva $|E|$ er.

DAG-Shortest-Path kontra Dijkstra

Oppgave 11: Hvilke utsagn om fordeler ved å bruke DAG-Shortest-Path over Dijkstra stemmer?

- ~~Dijkstra fungerer ikke på rettede asykliske grafer, selv om de har positive kantvekter.~~
- DAG-Shortest-Path fungerer også med negative kantvekter, mens Dijkstra gjør ikke det.
- DAG-Shortest-Path har bedre asymptotisk tidskompleksitet enn Dijkstra i verste tilfelle om $|\mathrm{E}| = o(\mathrm{V} \lg \mathrm{V})$.
- DAG-Shortest-Path har bedre asymptotisk tidskompleksitet enn Dijkstra i verste tilfelle uansett hva $|E|$ er.

DAG-Shortest-Path kontra Dijkstra

Oppgave 11: Hvilke utsagn om fordeler ved å bruke DAG-Shortest-Path over Dijkstra stemmer?

- ~~Dijkstra fungerer ikke på rettede asykliske grafer, selv om de har positive kantvekter.~~
- DAG-Shortest-Path fungerer også med negative kantvekter, mens Dijkstra gjør ikke det.
- DAG-Shortest-Path har bedre asymptotisk tidskompleksitet enn Dijkstra i verste tilfelle om $|\mathrm{E}| = o(\mathrm{V} \lg \mathrm{V})$.
- ~~DAG-Shortest-Path har bedre asymptotisk tidskompleksitet enn Dijkstra i verste tilfelle uansett hva $|E|$ er.~~

Fibonacci-hauger

Oppgave 12: Hvilken operasjon tar $\Theta(\lg n)$ amortisert tid i verste tilfelle med en binærhaug, men $O(1)$ amortisert tid i verste tilfelle for en Fibonacci-haug?

Fibonacci-hauger

Oppgave 12: Hvilken operasjon tar $\Theta(\lg n)$ amortisert tid i verste tilfelle med en binærhaug, men $O(1)$ amortisert tid i verste tilfelle for en Fibonacci-haug?

	Binærhaug	Fibonacci-haug
DECREASE-KEY	$\Theta(\lg n)$	$\mathrm{O}(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$\Theta(\lg n)$
MINIMUM	$\mathrm{O}(1)$	$\mathrm{O}(1)$

Minst energi - Programmering

Oppgave 13: Du har et sett med reaksjoner som enten krever energi eller kan gi fra seg energi. Finn den måten å komme seg fra et gitt stoff til et annet gitt stoff som krever minst ekstra energi. Det finnes ingen måter å komme seg tilbake til samme stoff og samtidig bruke netto negativ energi.

Minst energi - Programmering

Oppgave 13: Du har et sett med reaksjoner som enten krever energi eller kan gi fra seg energi. Finn den måten å komme seg fra et gitt stoff til et annet gitt stoff som krever minst ekstra energi. Det finnes ingen måter å komme seg tilbake til samme stoff og samtidig bruke netto negativ energi.

1. Lag en graf med stoff som noder og reaksjoner som kanter.
2. Utfør Bellman-Ford på grafen fra startstoffet.
3. Lag en sti fra startstoffet til målstoffet ved hjelp av $v.\pi$.

Korteste togrute - Programmering

Oppgave 14: Du ønsker å komme deg fra en by til en annen med ekspresstog og har et sett med rutetider for når det går tog mellom forskjellige byer. Finn det tidligste tidspunktet du kan ankomme reisemålet.

Korteste togrute - Programmering

Oppgave 14: Du ønsker å komme deg fra en by til en annen med ekspresstog og har et sett med rutetider for når det går tog mellom forskjellige byer. Finn det tidligste tidspunktet du kan ankomme reisemålet.

Utfør DIJKSTRA, hvor RELAX bruker den av de mulige togturene som kommer frem tidligst til hver av byene.

Dijkstra - færrest kanter

Oppgave 15: Hvordan kan du endre kantvektene i en graf hvor kantene har positive heltallsvekter slik at DIJKSTRA finner den av de korteste veiene som har færrest mulig kanter?

Dijkstra - færrest kanter

Oppgave 15: Hvordan kan du endre kantvektene i en graf hvor kantene har positive heltallsvekter slik at DIJKSTRA finner den av de korteste veiene som har færrest mulig kanter?

$$w'(u, v) = w(u, v) + \frac{1}{|\mathrm{V}|}$$

New-SSSP

Oppgave 16: Vil denne algoritmen løse korteste-vei-problemet?

Oppgave 17: Gi et bevis for at svaret er riktig.

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

New-SSSP - Kjøretid

Oppgave 18: Hva er kjøretiden til New-SSSP i beste tilfellet?

Oppgave 19: Hva er kjøretiden til New-SSSP i verste tilfellet?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

New-SSSP - Kjøretid

Oppgave 18: Hva er kjøretiden til New-SSSP i beste tilfellet?

Oppgave 19: Hva er kjøretiden til New-SSSP i verste tilfellet?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

Beste tilfelle: $\Omega(\mathrm{V})$

New-SSSP - Kjøretid

Oppgave 18: Hva er kjøretiden til New-SSSP i beste tilfellet?

Oppgave 19: Hva er kjøretiden til New-SSSP i verste tilfellet?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

Beste tilfelle: $\Omega(\mathrm{V})$ og $\mathrm{O}(\mathrm{V})$

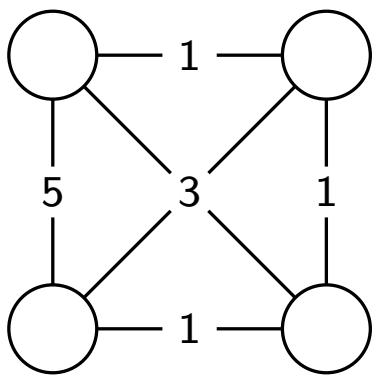
New-SSSP - Kjøretid

Oppgave 18: Hva er kjøretiden til New-SSSP i beste tilfellet?

Oppgave 19: Hva er kjøretiden til New-SSSP i verste tilfellet?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

Beste tilfelle: $\Omega(\mathrm{V})$ og $\mathrm{O}(\mathrm{V})$



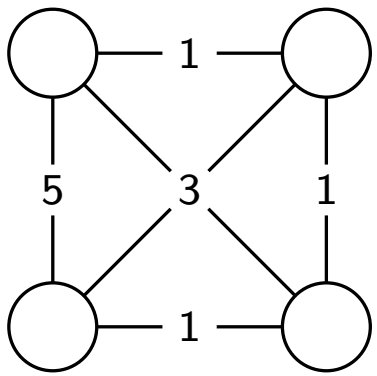
New-SSSP - Kjøretid

Oppgave 18: Hva er kjøretiden til New-SSSP i beste tilfellet?

Oppgave 19: Hva er kjøretiden til New-SSSP i verste tilfellet?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

Beste tilfelle: $\Omega(\mathrm{V})$ og $\mathrm{O}(\mathrm{V})$



Verste tilfelle: $\Omega(\mathrm{VE})$

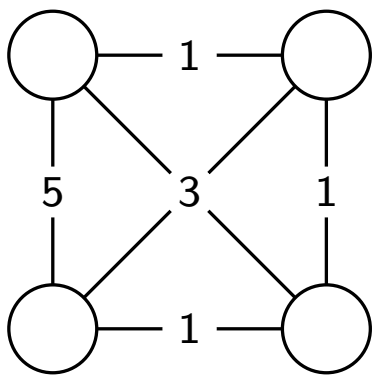
New-SSSP - Kjøretid

Oppgave 18: Hva er kjøretiden til New-SSSP i beste tilfellet?

Oppgave 19: Hva er kjøretiden til New-SSSP i verste tilfellet?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

Beste tilfelle: $\Omega(\mathrm{V})$ og $\mathrm{O}(\mathrm{V})$



Verste tilfelle: $\Omega(\mathrm{VE})$ og $\mathrm{O}(\mathrm{VE})$

New-SSSP - Negative sykler

Oppgave 20: Fungerer New-SSSP når det finnes negative sykler i grafen?

```
New-SSSP(G, w, s)
 1  Initialize-Single-Source(G, s)
 2  for each vertex v ∈ V
 3       v.queued = FALSE
 4  let Q = ∅ be a FIFO queue
 5  Enqueue(Q, s)
 6  s.queued = TRUE
 7  while Q ≠ ∅
 8       u = Dequeue(Q)
 9       u.queued = FALSE
10       for each v ∈ G.adj[u]
11            if v.d > u.d + w(u, v)
12                 v.d = u.d + w(u, v)
13                 v.π = u
14                 if not v.queued
15                      Enqueue(Q, v)
16                      v.queued = TRUE
```

Hamiltonsti

Oppgave 21: Hvordan kan du finne ut om det finnes en Hamilton sti i en graf gitt at du har en metode for å finne den korteste enkle stien mellom to noder i arbitrære grafer?

Hamiltonsti

Oppgave 21: Hvordan kan du finne ut om det finnes en Hamilton sti i en graf gitt at du har en metode for å finne den korteste enkle stien mellom to noder i arbitrære grafer?

Sett kantvektene til -1 og sjekk alle nodepar.

Hamiltonsti

Oppgave 21: Hvordan kan du finne ut om det finnes en Hamilton sti i en graf gitt at du har en metode for å finne den korteste enkle stien mellom to noder i arbitrære grafer?

Sett kantvektene til -1 og sjekk alle nodepar.

Kan slippe å bruke metoden mange ganger ved å legge til to noder som har kanter til alle andre noder.