Øvingsforelesning 12

TDT4120 - Algoritmer og datastrukturer

Øving 11

Korteste vei fra alle til alle

Oppgave 2: Hvordan kan du finne korteste vei fra alle til alle i en rettet graf med ikke-negative kantvekter, ved hjelp av Dijkstras algoritme?

Korteste vei fra alle til alle

Oppgave 2: Hvordan kan du finne korteste vei fra alle til alle i en rettet graf med ikke-negative kantvekter, ved hjelp av Dijkstras algoritme?

Bruk algoritmen til å finne korteste vei fra én til alle fra hver node.

Korteste vei fra alle til alle

Oppgave 2: Hvordan kan du finne korteste vei fra alle til alle i en rettet graf med ikke-negative kantvekter, ved hjelp av Dijkstras algoritme?

Bruk algoritmen til å finne korteste vei fra én til alle fra hver node.

Oppgave 3: Hvordan finner man raskest korteste vei mellom alle par med noder i en rettet graf uten negative sykler, gitt muligheten til å bruke DIJKSTRA eller BELLMAN-FORD? Hva blir kjøretiden?

Korteste vei fra alle til alle

Oppgave 2: Hvordan kan du finne korteste vei fra alle til alle i en rettet graf med ikke-negative kantvekter, ved hjelp av Dijkstras algoritme?

Bruk algoritmen til å finne korteste vei fra én til alle fra hver node.

Oppgave 3: Hvordan finner man raskest korteste vei mellom alle par med noder i en rettet graf uten negative sykler, gitt muligheten til å bruke DIJKSTRA eller BELLMAN-FORD? Hva blir kjøretiden?

DIJKSTRA håndterer ikke negative kanter. Må bruke BELLMAN-FORD.

Korteste vei fra alle til alle

Oppgave 2: Hvordan kan du finne korteste vei fra alle til alle i en rettet graf med ikke-negative kantvekter, ved hjelp av Dijkstras algoritme?

Bruk algoritmen til å finne korteste vei fra én til alle fra hver node.

Oppgave 3: Hvordan finner man raskest korteste vei mellom alle par med noder i en rettet graf uten negative sykler, gitt muligheten til å bruke DIJKSTRA eller BELLMAN-FORD? Hva blir kjøretiden?

DIJKSTRA håndterer ikke negative kanter. Må bruke BELLMAN-FORD.

BELLMAN-FORD har en kjøretid på $O(VE)$.

Korteste vei fra alle til alle

Oppgave 2: Hvordan kan du finne korteste vei fra alle til alle i en rettet graf med ikke-negative kantvekter, ved hjelp av Dijkstras algoritme?

Bruk algoritmen til å finne korteste vei fra én til alle fra hver node.

Oppgave 3: Hvordan finner man raskest korteste vei mellom alle par med noder i en rettet graf uten negative sykler, gitt muligheten til å bruke DIJKSTRA eller BELLMAN-FORD? Hva blir kjøretiden?

DIJKSTRA håndterer ikke negative kanter. Må bruke BELLMAN-FORD.

BELLMAN-FORD har en kjøretid på $\mathrm{O}(\mathrm{VE})$.

Total kjøretid $\mathrm{O}(\mathrm{V}^2\mathrm{E})$.

Forgjengermatriser

Oppgave 4: Hva forteller π_{ij} oss i en forgjengermatrise?
Oppgave 5: Hva betyr det hvis $\pi_{ij} = \textsc{nil}$?

Forgjengermatriser

Oppgave 4: Hva forteller π_{ij} oss i en forgjengermatrise?
Oppgave 5: Hva betyr det hvis $\pi_{ij} = \text{NIL}$?

Som $v.\pi$ i én-til-alle tilfellet, foregående node på en av de korteste stiene fra node i til j.

Floyd-Warshall

Oppgave 6: Hva er kjøretiden til Floyd-Warshall?

Floyd-Warshall(W)

$n = \mathrm{W}.\mathit{rows}$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $\mathrm{D}^{(n)}$

Floyd-Warshall

Oppgave 6: Hva er kjøretiden til Floyd-Warshall?

Tre nøstede iterasjoner til $n = |V|$, $\mathrm{O}(\mathrm{V}^3)$.

Floyd-Warshall(W)

$n = \mathrm{W}.rows$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $\mathrm{D}^{(n)}$

Floyd-Warshall

Oppgave 6: Hva er kjøretiden til Floyd-Warshall?

Tre nøstede iterasjoner til $n = |V|$, $\mathrm{O}(\mathrm{V}^3)$.

Oppgave 7: Hvordan kan vi modifisere Floyd-Warshall til å bruke mindre minne?

Floyd-Warshall(W)

$n = \mathrm{W}.rows$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $\mathrm{D}^{(n)}$

Floyd-Warshall

Oppgave 6: Hva er kjøretiden til Floyd-Warshall?

Tre nøstede iterasjoner til $n = |V|$, $O(V^3)$.

Oppgave 7: Hvordan kan vi modifisere Floyd-Warshall til å bruke mindre minne?

Reduser antall matriser D som anvendes.

```
Floyd-Warshall(W)
n = W.rows
D^(0) = W
for k = 1 to n
    let D^(k) = (d_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

Floyd-Warshall

Oppgave 8: Hvilke utsagn stemmer om Floyd-Warshall?

- Etter at Floyd-Warshall har kjørt, kan diagonalen i avstandsmatrisen D inneholde positive tall.
- Etter at Floyd-Warshall har kjørt, kan diagonalen i avstandsmatrisen D inneholde negative tall.

Floyd-Warshall(W)

$n = \mathrm{W}.rows$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $\mathrm{D}^{(n)}$

Floyd-Warshall

Oppgave 8: Hvilke utsagn stemmer om Floyd-Warshall?

- Etter at Floyd-Warshall har kjørt, kan diagonalen i avstandsmatrisen D inneholde positive tall.
- Etter at Floyd-Warshall har kjørt, kan diagonalen i avstandsmatrisen D inneholde negative tall.

Diagonalen i W består av nuller.

Floyd-Warshall(W)

$n = \mathrm{W}.rows$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $\mathrm{D}^{(n)}$

Floyd-Warshall

Oppgave 8: Hvilke utsagn stemmer om Floyd-Warshall?

- ~~Etter at Floyd-Warshall har kjørt, kan diagonalen i avstandsmatrisen D inneholde positive tall~~.
- Etter at Floyd-Warshall har kjørt, kan diagonalen i avstandsmatrisen D inneholde negative tall.

Diagonalen i W består av nuller.

```
Floyd-Warshall(W)
  n = W.rows
  D^(0) = W
  for k = 1 to n
        let D^(k) = (d_ij^(k)) be a new n × n matrix
        for i = 1 to n
              for j = 1 to n
                    d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
  return D^(n)
```

Generalisert Floyd-Warshall - Programmering

Oppgave 9: Implementer en generalisert versjon av Floyd-Warshall, som bruker to arbitrære funksjoner f og g i stedet for min og $+$.

Generalisert FLOYD-WARSHALL - Programmering

Oppgave 9: Implementer en generalisert versjon av FLOYD-WARSHALL, som bruker to arbitrære funksjoner f og g i stedet for min og $+$.

GENERAL-FLOYD-WARSHALL(W, f, g)

$n = \mathrm{W}.rows$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = f(d_{ij}^{(k-1)}, g(d_{ik}^{(k-1)}, d_{kj}^{(k-1)}))$
return $\mathrm{D}^{(n)}$

Transitiv tillukning

Oppgave 10: Hva er utrykket for $t_{ij}^{(k)}$ i $\textsc{Transitive-Closure}$ når $k > 0$?

Transitiv tillukning

Oppgave 10: Hva er utrykket for $t_{ij}^{(k)}$ i $\textsc{Transitive-Closure}$ når $k > 0$?

$t_{ij}^{(k)}$ indikerer om det går en sti mellom noder i og j som kun går igjennom de første k nodene.

Transitiv tillukning

Oppgave 10: Hva er utrykket for $t_{ij}^{(k)}$ i $\textsc{Transitive-Closure}$ når $k > 0$?

$t_{ij}^{(k)}$ indikerer om det går en sti mellom noder i og j som kun går igjennom de første k nodene.

For en gitt $k > 0$ må dette gjelde for enten $k - 1$ eller så må vi ha at vi kan gå igjennom k og oppnå dette.

Transitiv tillukning

Oppgave 10: Hva er utrykket for $t_{ij}^{(k)}$ i $\textsc{Transitive-Closure}$ når $k > 0$?

$t_{ij}^{(k)}$ indikerer om det går en sti mellom noder i og j som kun går igjennom de første k nodene.

For en gitt $k > 0$ må dette gjelde for enten $k - 1$ eller så må vi ha at vi kan gå igjennom k og oppnå dette.

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$$

Transitiv tillukning

Oppgave 10: Hva er utrykket for $t_{ij}^{(k)}$ i TRANSITIVE-CLOSURE når $k > 0$?

$t_{ij}^{(k)}$ indikerer om det går en sti mellom noder i og j som kun går igjennom de første k nodene.

For en gitt $k > 0$ må dette gjelde for enten $k - 1$ eller så må vi ha at vi kan gå igjennom k og oppnå dette.

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$$

Oppgave 11: Hva betyr $t_{ij}^{(k)} = 0$ i TRANSITIVE-CLOSURE?

Transitiv tillukning

Oppgave 10: Hva er utrykket for $t_{ij}^{(k)}$ i TRANSITIVE-CLOSURE når $k > 0$?

$t_{ij}^{(k)}$ indikerer om det går en sti mellom noder i og j som kun går igjennom de første k nodene.

For en gitt $k > 0$ må dette gjelde for enten $k - 1$ eller så må vi ha at vi kan gå igjennom k og oppnå dette.

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$$

Oppgave 11: Hva betyr $t_{ij}^{(k)} = 0$ i TRANSITIVE-CLOSURE?

Det finnes ingen sti fra i til j som kun går igjennom noen av de k første nodene.

Transitiv tillukning - Programmering

Oppgave 12: Gitt nabomatrisen, T, bruk den generelle implementasjonen av Floyd-Warshall til å implementere Transitive-Closure.

Transitiv tillukning - Programmering

Oppgave 12: Gitt nabomatrisen, T, bruk den generelle implementasjonen av FLOYD-WARSHALL til å implementere TRANSITIVE-CLOSURE.

Både FLOYD-WARSHALL og TRANSITIVE-CLOSURE bruker $|V|$ iterasjoner og oppdaterer hver verdi i matrisen i hver iterasjon.

Transitiv tillukning - Programmering

Oppgave 12: Gitt nabomatrisen, T, bruk den generelle implementasjonen av Floyd-Warshall til å implementere Transitive-Closure.

Både Floyd-Warshall og Transitive-Closure bruker $|V|$ iterasjoner og oppdaterer hver verdi i matrisen i hver iterasjon.

Oppdateringer:
General-Floyd-Warshall $d_{ij}^{(k)} = f(d_{ij}^{(k-1)}, g(d_{ik}^{(k-1)}, d_{kj}^{(k-1)}))$
Transitive-Closure $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$

Transitiv tillukning - Programmering

Oppgave 12: Gitt nabomatrisen, T, bruk den generelle implementasjonen av $\textsc{Floyd-Warshall}$ til å implementere $\textsc{Transitive-Closure}$.

Både $\textsc{Floyd-Warshall}$ og $\textsc{Transitive-Closure}$ bruker $|V|$ iterasjoner og oppdaterer hver verdi i matrisen i hver iterasjon.

Oppdateringer:
$\textsc{General-Floyd-Warshall}$ $d_{ij}^{(k)} = f(d_{ij}^{(k-1)}, g(d_{ik}^{(k-1)}, d_{kj}^{(k-1)}))$
$\textsc{Transitive-Closure}$ $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$

$\textsc{Transitive-Closure}(\mathrm{T})$
1 return $\textsc{Floyd-Warshall}(\mathrm{T}, \vee, \wedge)$

Johnsons algoritme

Oppgave 13: Hvilke algoritmer brukes som subrutiner i Johnsons algoritme?

```
JOHNSON(G, w)
compute G′, where G′.V = G.V ∪ {s},
      G′.E = G.E ∪ {(s, v) : v ∈ G.V}, and
      w(s, v) = 0 for all v ∈ G.V
if BELLMAN-FORD(G′, w, s) == FALSE
      print "the input graph contains a negative-weight cycle"
else for each vertex v ∈ G′.V
            set h(v) to the value of δ(s, v)
                  computed by the Bellman-Ford algorithm
      for each edge (u, v) ∈ G′.E
            ŵ(u, v) = w(u, v) + h(u) − h(v)
      let D = (d_uv) be a new n × n matrix
      for each vertex u ∈ G.V
            run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
            for each vertex v ∈ G.V
                  d_uv = δ̂(u, v) + h(v) − h(u)
      return D
```

Johnsons algoritme

Oppgave 14: Anta at vi bruker en binær min-heap. Hvilken kjøretid har da Johnsons algoritme?

```
JOHNSON(G, w)
compute G′, where G′.V = G.V ∪ {s},
      G′.E = G.E ∪ {(s, v) : v ∈ G.V}, and
      w(s, v) = 0 for all v ∈ G.V
if BELLMAN-FORD(G′, w, s) == FALSE
      print "the input graph contains a negative-weight cycle"
else for each vertex v ∈ G′.V
            set h(v) to the value of δ(s, v)
                  computed by the Bellman-Ford algorithm
      for each edge (u, v) ∈ G′.E
            ŵ(u, v) = w(u, v) + h(u) − h(v)
      let D = (d_uv) be a new n × n matrix
      for each vertex u ∈ G.V
            run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
            for each vertex v ∈ G.V
                  d_uv = δ̂(u, v) + h(v) − h(u)
      return D
```

Johnsons algoritme

Oppgave 14: Anta at vi bruker en binær min-heap. Hvilken kjøretid har da Johnsons algoritme? – $\mathrm{O}(\mathrm{VE} \lg \mathrm{V})$

$\textsc{Johnson}(\mathrm{G}, w)$
compute G', where $\mathrm{G}'.\mathrm{V} = \mathrm{G}.\mathrm{V} \cup \{s\}$,
    $\mathrm{G}'.\mathrm{E} = \mathrm{G}.\mathrm{E} \cup \{(s, v) : v \in \mathrm{G}.\mathrm{V}\}$, and
    $w(s, v) = 0$ for all $v \in \mathrm{G}.\mathrm{V}$
if $\textsc{Bellman-Ford}(\mathrm{G}', w, s) == \textsc{False}$
    print "the input graph contains a negative-weight cycle"
else for each vertex $v \in \mathrm{G}'.\mathrm{V}$
        set $h(v)$ to the value of $\delta(s, v)$
            computed by the Bellman-Ford algorithm
    for each edge $(u, v) \in \mathrm{G}'.\mathrm{E}$
        $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
    let $\mathrm{D} = (d_{uv})$ be a new $n \times n$ matrix
    for each vertex $u \in \mathrm{G}.\mathrm{V}$
        run $\textsc{Dijkstra}(\mathrm{G}, \hat{w}, u)$ to compute $\hat{\delta}(u, v)$ for all $v \in \mathrm{G}.\mathrm{V}$
        for each vertex $v \in \mathrm{G}.\mathrm{V}$
            $d_{uv} = \hat{\delta}(u, v) + h(v) - h(u)$
    return D

Johnsons algoritme

Oppgave 15: Hvilken teknikk er det som gjør Johnsons algoritme spesiell?

```
JOHNSON(G, w)
compute G′, where G′.V = G.V ∪ {s},
      G′.E = G.E ∪ {(s, v) : v ∈ G.V}, and
      w(s, v) = 0 for all v ∈ G.V
if BELLMAN-FORD(G′, w, s) == FALSE
      print "the input graph contains a negative-weight cycle"
else for each vertex v ∈ G′.V
            set h(v) to the value of δ(s, v)
                  computed by the Bellman-Ford algorithm
      for each edge (u, v) ∈ G′.E
            ŵ(u, v) = w(u, v) + h(u) − h(v)
      let D = (d_uv) be a new n × n matrix
      for each vertex u ∈ G.V
            run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
            for each vertex v ∈ G.V
                  d_uv = δ̂(u, v) + h(v) − h(u)
      return D
```

Johnsons algoritme

Oppgave 16: Hva blir kjøretiden til Johnsons algoritme i en rettet komplett digraf, dersom vi antar at vi bruker en Fibonacci-haug?

Oppgave 16: Hva blir kjøretiden til Johnsons algoritme i en rettet komplett digraf, dersom vi antar at vi bruker en Fibonacci-haug?

Generell kjøretid: $\mathrm{O}(\mathrm{V}^2 \lg \mathrm{V} + \mathrm{VE})$.

Johnsons algoritme

Oppgave 16: Hva blir kjøretiden til Johnsons algoritme i en rettet komplett digraf, dersom vi antar at vi bruker en Fibonacci-haug?

Generell kjøretid: $\mathrm{O}(\mathrm{V}^2 \lg \mathrm{V} + \mathrm{VE})$.

I en komplett digraf: $|\mathrm{E}| = \Theta(\mathrm{V}^2)$

Johnsons algoritme

Oppgave 16: Hva blir kjøretiden til Johnsons algoritme i en rettet komplett digraf, dersom vi antar at vi bruker en Fibonacci-haug?

Generell kjøretid: $\mathrm{O}(\mathrm{V}^2 \lg \mathrm{V} + \mathrm{VE})$.

I en komplett digraf: $|\mathrm{E}| = \Theta(\mathrm{V}^2)$

Kjøretid i en komplett digraf: $\mathrm{O}(\mathrm{V}^3)$

Schulzemetoden - Programmering

Oppgave 17: Schulzemetoden brukes til å bestemme vinneren i et valg hvor hver stemme rangerer kandidatene. Dette kan rangeres som en graf, hvor det går en kant fra kandidat i til j hvis i er rangert over j på over 50% av stemmene. Kandidatene rangeres basert på den sterkeste stien mellom dem, altså stien hvor den laveste kantvekten er størst mulig. Implementer denne algoritmen.

Schulzemetoden - Programmering

Oppgave 17: Schulzemetoden brukes til å bestemme vinneren i et valg hvor hver stemme rangerer kandidatene. Dette kan rangeres som en graf, hvor det går en kant fra kandidat *i* til *j* hvis *i* er rangert over *j* på over 50% av stemmene. Kandidatene rangeres basert på den sterkeste stien mellom dem, altså stien hvor den laveste kantvekten er størst mulig. Implementer denne algoritmen.

```
SCHULZE(A)
for i = 1 to A.length
    for j = 1 to A.length
        if A[i][j] ⩽ A[j][i]
            A[i][j] = 0
A = GENERAL-FLOYD-WARSHALL(A, max, min)
return the sorted numbers from 1 to n where i is earlier
    in the list than j if A[i][j] ⩾ A[j][i]
```