Øvingsforelesning 7

TDT4120 - Algoritmer og datastrukturer

Øving 6

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

- Problemet må kun ha én riktig løsning.
- Input må kun bestå av heltall.
- Man må bruke rekursive funksjoner for å løse problemet.
- Problemet må ha optimal delstruktur.
- Det må finnes overlappende delproblemer.

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

- ~~Problemet må kun ha én riktig løsning.~~
- Input må kun bestå av heltall.
- Man må bruke rekursive funksjoner for å løse problemet.
- Problemet må ha optimal delstruktur.
- Det må finnes overlappende delproblemer.

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

- ~~Problemet må kun ha én riktig løsning.~~
- ~~Input må kun bestå av heltall.~~
- Man må bruke rekursive funksjoner for å løse problemet.
- Problemet må ha optimal delstruktur.
- Det må finnes overlappende delproblemer.

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

- ~~Problemet må kun ha én riktig løsning.~~
- ~~Input må kun bestå av heltall.~~
- ~~Man må bruke rekursive funksjoner for å løse problemet.~~
- Problemet må ha optimal delstruktur.
- Det må finnes overlappende delproblemer.

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

Oppgave 3: Hva vil det si at et problem har optimal delstruktur?

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

Oppgave 3: Hva vil det si at et problem har optimal delstruktur?

En optimal løsning på en instans kan konstureres fra optimal løsninger på delinstansene.

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

Oppgave 3: Hva vil det si at et problem har optimal delstruktur?

En optimal løsning på en instans kan konstureres fra optimal løsninger på delinstansene.

Oppgave 4: Hva vil det si at et problem har overlappende delproblemer?

Dynamisk programmering - egenskaper og definisjoner

Oppgave 2: Hvilke egenskaper må et problem ha for at det skal gi mening å bruke dynamisk programmering til å løse problemet?

Oppgave 3: Hva vil det si at et problem har optimal delstruktur?

En optimal løsning på en instans kan konstureres fra optimal løsninger på delinstansene.

Oppgave 4: Hva vil det si at et problem har overlappende delproblemer?

Flere oppdelinger av problemet inneholder noen av de samme delproblemene.

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Kan regne ut iterativt.

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Kan regne ut iterativt.

$$r_1 = \max(p_1) = 1$$

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Kan regne ut iterativt.

$$r_1 = \max(p_1) = 1$$
$$r_2 = \max(r_1 + r_1, p_2) = 4$$

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Kan regne ut iterativt.

$$r_1 = \max(p_1) = 1$$
$$r_2 = \max(r_1 + r_1, p_2) = 4$$
$$r_3 = \max(r_1 + r_2, p_3) = 5$$

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Kan regne ut iterativt.

$$
\begin{aligned}
r_1 &= \max(p_1) = 1 \\
r_2 &= \max(r_1 + r_1, p_2) = 4 \\
r_3 &= \max(r_1 + r_2, p_3) = 5 \\
r_4 &= \max(r_1 + r_3, r_2 + r_2, p_4) = 8
\end{aligned}
$$

Stavkutting

Oppgave 5: La $n = 7$ og $p = \langle 1, 4, 3, 6, 8, 5, 9 \rangle$ være en instans av stavkuttingsproblemet. Hva blir da den maksimale inntekten, r_7?

Kan regne ut iterativt.

$$
\begin{aligned}
r_1 &= \max(p_1) = 1 \\
r_2 &= \max(r_1 + r_1, p_2) = 4 \\
r_3 &= \max(r_1 + r_2, p_3) = 5 \\
r_4 &= \max(r_1 + r_3, r_2 + r_2, p_4) = 8 \\
r_5 &= \max(r_1 + r_4, r_2 + r_3, p_5) = 9, \\
r_6 &= \max(r_1 + r_5, r_2 + r_4, r_3 + r_3, p_6) = 12 \\
r_7 &= \max(r_1 + r_6, r_2 + r_5, r_3 + r_4, p_7) = 13
\end{aligned}
$$

Stavkutting - Unik løsning

Oppgave 6: Oppdeling i lengde 2 og 4 er den eneste optimale løsningen for $n = 6$. Hvilke alternativer er da garantert **ikke** løsninger for $n = 8$?

Stavkutting - Unik løsning

Oppgave 6: Oppdeling i lengde 2 og 4 er den eneste optimale løsningen for $n = 6$. Hvilke alternativer er da garantert **ikke** løsninger for $n = 8$?

Ingen løsninger for $n = 8$ kan være optimale hvis de inneholder et subsett av staver som summerer til $2, 4$ eller 6 uten at dette subsettet er $\langle 2 \rangle, \langle 4 \rangle$ eller $\langle 2, 4 \rangle$.

Stavkutting - Unik løsning

Oppgave 6: Oppdeling i lengde 2 og 4 er den eneste optimale løsningen for $n = 6$. Hvilke alternativer er da garantert **ikke** løsninger for $n = 8$?

Ingen løsninger for $n = 8$ kan være optimale hvis de inneholder et subsett av staver som summerer til $2, 4$ eller 6 uten at dette subsettet er $\langle 2 \rangle, \langle 4 \rangle$ eller $\langle 2, 4 \rangle$.

- 8 staver av lengde 1.
- 2 staver av lengde 4.
- 2 staver av lengde 2 og 1 stav av lengde 4.
- 1 stav av lengde 3 og 1 stav av lengde 5.
- 2 staver av lengde 3 og 1 stav av lengde 2.
- 4 staver av lengde 2.

Stavkutting - Unik løsning

Oppgave 6: Oppdeling i lengde 2 og 4 er den eneste optimale løsningen for $n = 6$. Hvilke alternativer er da garantert **ikke** løsninger for $n = 8$?

Ingen løsninger for $n = 8$ kan være optimale hvis de inneholder et subsett av staver som summerer til $2, 4$ eller 6 uten at dette subsettet er $\langle 2 \rangle, \langle 4 \rangle$ eller $\langle 2, 4 \rangle$.

- 8 staver av lengde 1.
- ~~2 staver av lengde 4.~~
- 2 staver av lengde 2 og 1 stav av lengde 4.
- 1 stav av lengde 3 og 1 stav av lengde 5.
- 2 staver av lengde 3 og 1 stav av lengde 2.
- 4 staver av lengde 2.

Stavkutting - Unik løsning

Oppgave 6: Oppdeling i lengde 2 og 4 er den eneste optimale løsningen for $n = 6$. Hvilke alternativer er da garantert **ikke** løsninger for $n = 8$?

Ingen løsninger for $n = 8$ kan være optimale hvis de inneholder et subsett av staver som summerer til $2, 4$ eller 6 uten at dette subsettet er $\langle 2 \rangle, \langle 4 \rangle$ eller $\langle 2, 4 \rangle$.

- 8 staver av lengde 1.
- ~~2 staver av lengde 4.~~
- 2 staver av lengde 2 og 1 stav av lengde 4.
- ~~1 stav av lengde 3 og 1 stav av lengde 5.~~
- 2 staver av lengde 3 og 1 stav av lengde 2.
- 4 staver av lengde 2.

Deling av metallstykker - Programmering

Oppgave 7: Ønsker å dele et metallstykke i rektangulære stykker for å maksimere prisen. (2D-stavkutting).

Deling av metallstykker - Programmering

Oppgave 7: Ønsker å dele et metallstykke i rektangulære stykker for å maksimere prisen. (2D-stavkutting).

```
SHEET-CUTTING(w, h, p)
1  for i = 1 to w
2        for j = 1 to h
3              for k = 1 to ⌊j/2⌋
4                    p[i, j] = max(p[i, j − k] + p[i, k], p[i, j])
5              for k = 1 to ⌊i/2⌋
6                    p[i, j] = max(p[i − k, j] + p[k, j], p[i, j])
7  return p[w, h]
```

Memoisering eller iterasjon - Fordeler

Oppgave 8: Hvilke fordeler har memoisering over iterasjon?
Oppgave 9: Hvilke fordeler har iterasjon over memoisering?

- Man har generelt mindre overhead.
- Man kan potensielt ende opp med å løse færre delinstanser.
- Man vil alltid oppnå en bedre tidskompleksitet.
- Man slipper å løse samme delinstans flere ganger.

Memoisering eller iterasjon - Fordeler

Oppgave 8: Hvilke fordeler har memoisering over iterasjon?
Oppgave 9: Hvilke fordeler har iterasjon over memoisering?

- Man har generelt mindre overhead. **[Iterasjon]**
- Man kan potensielt ende opp med å løse færre delinstanser.
- Man vil alltid oppnå en bedre tidskompleksitet.
- Man slipper å løse samme delinstans flere ganger.

Memoisering eller iterasjon - Fordeler

Oppgave 8: Hvilke fordeler har memoisering over iterasjon?
Oppgave 9: Hvilke fordeler har iterasjon over memoisering?

- Man har generelt mindre overhead. **[Iterasjon]**
- Man kan potensielt ende opp med å løse færre delinstanser. **[Memoisering]**
- Man vil alltid oppnå en bedre tidskompleksitet.
- Man slipper å løse samme delinstans flere ganger.

Memoisering eller iterasjon - Fordeler

Oppgave 8: Hvilke fordeler har memoisering over iterasjon?
Oppgave 9: Hvilke fordeler har iterasjon over memoisering?

- Man har generelt mindre overhead. **[Iterasjon]**
- Man kan potensielt ende opp med å løse færre delinstanser. **[Memoisering]**
- ~~Man vil alltid oppnå en bedre tidskompleksitet.~~
- Man slipper å løse samme delinstans flere ganger.

Memoisering eller iterasjon - Fordeler

Oppgave 8: Hvilke fordeler har memoisering over iterasjon?
Oppgave 9: Hvilke fordeler har iterasjon over memoisering?

- Man har generelt mindre overhead. **[Iterasjon]**
- Man kan potensielt ende opp med å løse færre delinstanser. **[Memoisering]**
- ~~Man vil alltid oppnå en bedre tidskompleksitet.~~
- ~~Man slipper å løse samme delinstans flere ganger.~~

Quicksort og dynamisk programmering

Oppgave 10: Kan vi bruke memoisering i quicksort til å oppnå en bedre tidskompleksitet?

Quicksort og dynamisk programmering

Oppgave 10: Kan vi bruke memoisering i quicksort til å oppnå en bedre tidskompleksitet?

Quicksort deler rekursivt opp i to og to delinstanser ved hjelp av Partition.

Quicksort og dynamisk programmering

Oppgave 10: Kan vi bruke memoisering i quicksort til å oppnå en bedre tidskompleksitet?

Quicksort deler rekursivt opp i to og to delinstanser ved hjelp av Partition.

Partition garanterer at delinstansene er unike.

Quicksort og dynamisk programmering

Oppgave 10: Kan vi bruke memoisering i quicksort til å oppnå en bedre tidskompleksitet?

Quicksort deler rekursivt opp i to og to delinstanser ved hjelp av Partition.

Partition garanterer at delinstansene er unike.

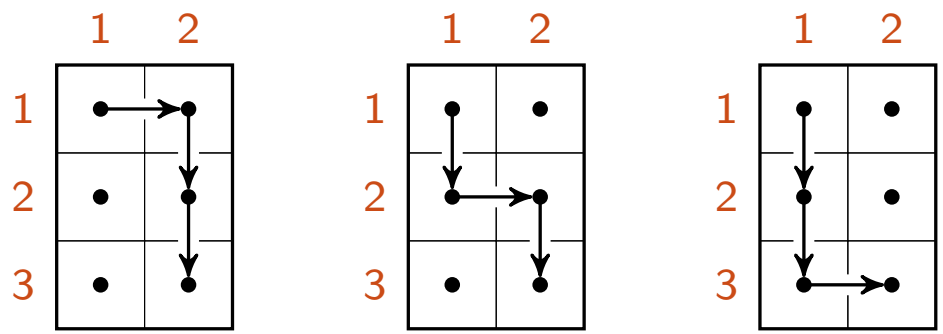
Dekomponeringen har ikke overlappende delproblemer.

Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?



Rutenettraversering

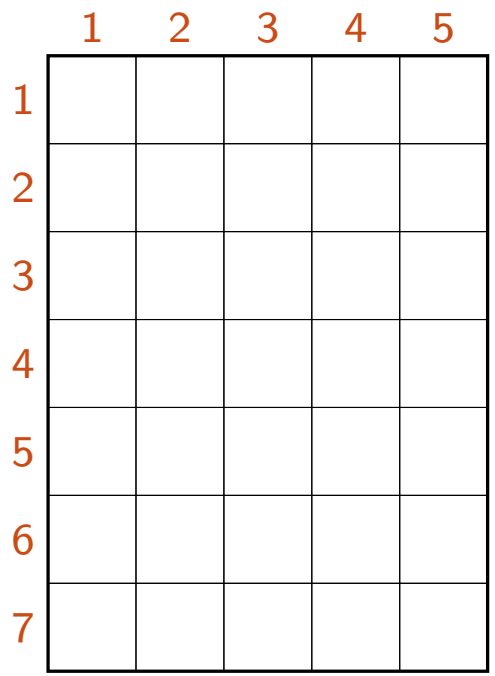
Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?

Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

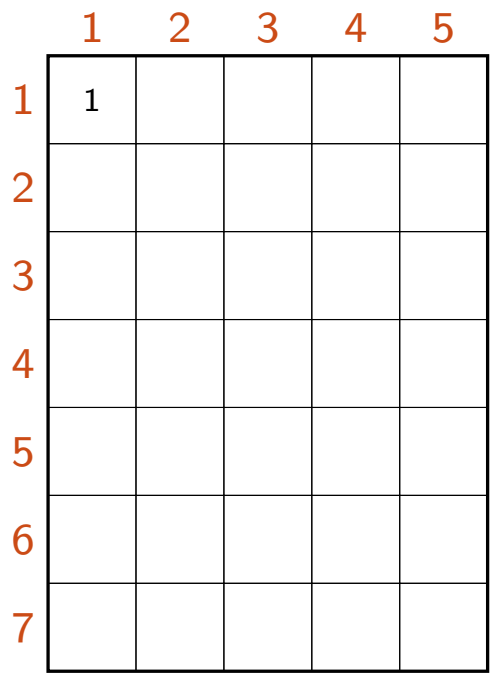
Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?



Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

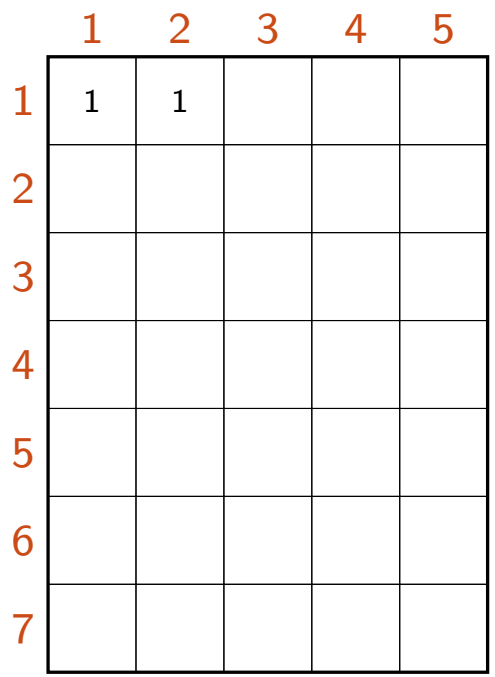
Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?



Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

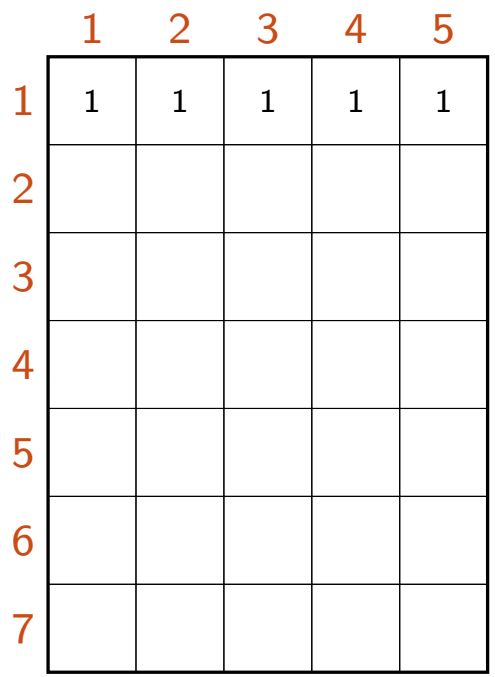
Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?



Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?



Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?

	1	2	3	4	5
1	1	1	1	1	1
2	1				
3	1				
4	1				
5	1				
6	1				
7	1				

Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?

	1	2	3	4	5
1	1	1	1	1	1
2	1	2			
3	1				
4	1				
5	1				
6	1				
7	1				

Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3		
3	1				
4	1				
5	1				
6	1				
7	1				

Rutenettraversering

Oppgave 11: I et rutenett hvor vi kun kan gå til høyre eller ned i hvert steg. På hvor mange måter kan man komme fra $(1, 1)$ til $(2, 3)$?

Oppgave 12: Hvor mange måter kan man komme fra $(1, 1)$ til $(5, 7)$?

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	70
6	1	6	21	56	126
7	1	7	28	84	210

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.
Alle de lengste delfølgene som slutter på s_i er enten kun $\langle s_i \rangle$ eller er den lengste delfølgen som slutter i et tall $s_j > s_i$ hvor $j < i$.

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.
Alle de lengste delfølgene som slutter på s_i er enten kun $\langle s_i \rangle$ eller er den lengste delfølgen som slutter i et tall $s_j > s_i$ hvor $j < i$.

1. La løsningen på problemet for $i = 1$ være $\langle s_1 \rangle$.

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.
Alle de lengste delfølgene som slutter på s_i er enten kun $\langle s_i \rangle$ eller er den lengste delfølgen som slutter i et tall $s_j > s_i$ hvor $j < i$.

1. La løsningen på problemet for $i = 1$ være $\langle s_1 \rangle$.
2. Løs problemet for iterativt for $i > 1$ ved å:

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.
Alle de lengste delfølgene som slutter på s_i er enten kun $\langle s_i \rangle$ eller er den lengste delfølgen som slutter i et tall $s_j > s_i$ hvor $j < i$.

1. La løsningen på problemet for $i = 1$ være $\langle s_1 \rangle$.
2. Løs problemet for iterativt for $i > 1$ ved å:
    a. Finn den lengste delfølgen S^* blant løsningene for $j < i$, hvor $s_j > s_i$ og la løsningen for i være $\mathrm{S}^* + \langle s_i \rangle$.

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.
Alle de lengste delfølgene som slutter på s_i er enten kun $\langle s_i \rangle$ eller er den lengste delfølgen som slutter i et tall $s_j > s_i$ hvor $j < i$.

1. La løsningen på problemet for $i = 1$ være $\langle s_1 \rangle$.
2. Løs problemet for iterativt for $i > 1$ ved å:
    a. Finn den lengste delfølgen S^* blant løsningene for $j < i$, hvor $s_j > s_i$ og la løsningen for i være $\mathrm{S}^* + \langle s_i \rangle$.
    b. Hvis S^* ikke finnes sett løsningen for i til $\langle s_i \rangle$.

Lengste strengt synkende delfølge - Programmering

Oppgave 13: Implementer kode som finner lengste strengt synkende delfølge av en følge, $\mathrm{S} = \langle s_1, s_2, \ldots s_n \rangle$.

Den lengste delfølgen av S er den lengste delfølgen som slutter i en av s_i-ene.
Alle de lengste delfølgene som slutter på s_i er enten kun $\langle s_i \rangle$ eller er den lengste delfølgen som slutter i et tall $s_j > s_i$ hvor $j < i$.

1. La løsningen på problemet for $i = 1$ være $\langle s_1 \rangle$.
2. Løs problemet for iterativt for $i > 1$ ved å:
   a. Finn den lengste delfølgen S^* blant løsningene for $j < i$, hvor $s_j > s_i$ og la løsningen for i være $\mathrm{S}^* + \langle s_i \rangle$.
   b. Hvis S^* ikke finnes sett løsningen for i til $\langle s_i \rangle$.
3. Returner den lengste løsningen funnet i 1. og 2.

Segmentering av sekvenser

Oppgave 14: Har to sekvenser av heltall, A og B, og ønsker å finne ut om det er mulig å dele opp A slik at hver segment summerer til et tall i B. Gir det mening å anvende dynamisk programmering med sammenhengende segmenter av A som delinstanser?

Segmentering av sekvenser

Oppgave 14: Har to sekvenser av heltall, A og B, og ønsker å finne ut om det er mulig å dele opp A slik at hver segment summerer til et tall i B. Gir det mening å anvende dynamisk programmering med sammenhengende segmenter av A som delinstanser?

For en A er dette kun mulig hvis det finnes en i slik at dette er mulig for $\langle a_1, a_2, \ldots, a_i \rangle$ og $(\sum_{j=i+1}^{n} a_j) \in B$.

Segmentering av sekvenser

Oppgave 14: Har to sekvenser av heltall, A og B, og ønsker å finne ut om det er mulig å dele opp A slik at hver segment summerer til et tall i B. Gir det mening å anvende dynamisk programmering med sammenhengende segmenter av A som delinstanser?

For en A er dette kun mulig hvis det finnes en i slik at dette er mulig for $\langle a_1, a_2, \ldots, a_i \rangle$ og $(\sum_{j=i+1}^{n} a_j) \in B$.

Kan først sjekke om dette er mulig for $\langle a_1 \rangle$. Så for $\langle a_1, a_2 \rangle$ ved å sjekke $i = 1$ og $i = 2$, og bruke resultatet for $\langle a_1 \rangle$.

Segmentering av sekvenser

Oppgave 14: Har to sekvenser av heltall, A og B, og ønsker å finne ut om det er mulig å dele opp A slik at hver segment summerer til et tall i B. Gir det mening å anvende dynamisk programmering med sammenhengende segmenter av A som delinstanser?

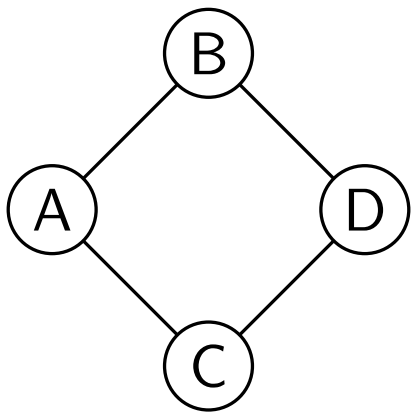
For en A er dette kun mulig hvis det finnes en i slik at dette er mulig for $\langle a_1, a_2, \ldots, a_i \rangle$ og $(\sum_{j=i+1}^{n} a_j) \in B$.

Kan først sjekke om dette er mulig for $\langle a_1 \rangle$. Så for $\langle a_1, a_2 \rangle$ ved å sjekke $i = 1$ og $i = 2$, og bruke resultatet for $\langle a_1 \rangle$.

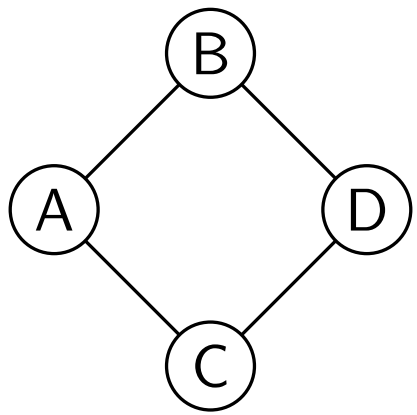
Kan fortsette å gjøre dette for alle segmenter av A på formen $\langle a_1, a_2, \ldots a_j \rangle$, med større og større verdier for j.

Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?

Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?

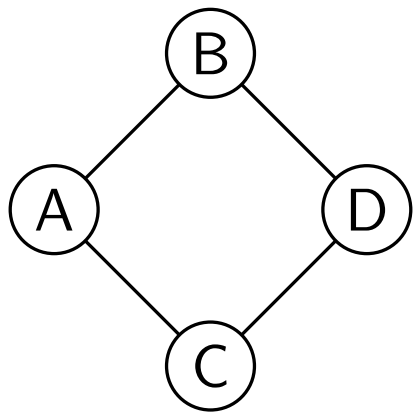


Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?



Den lengste veien fra A til B går igjennom C. Men, den lengste veien fra A til C er A - B - D - C og den lengste veien fra C til B er C - D - B eller C - A - B.

Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?



Den lengste veien fra A til B går igjennom C. Men, den lengste veien fra A til C er A - B - D - C og den lengste veien fra C til B er C - D - B eller C - A - B.

Har ikke optimal delstruktur.

Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?

Oppgave 16: Gir det mening å bruke dynamisk programmering hvis alle veiene er enveiskjørt og det ikke er mulig å kjøre i sirkel?

Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?

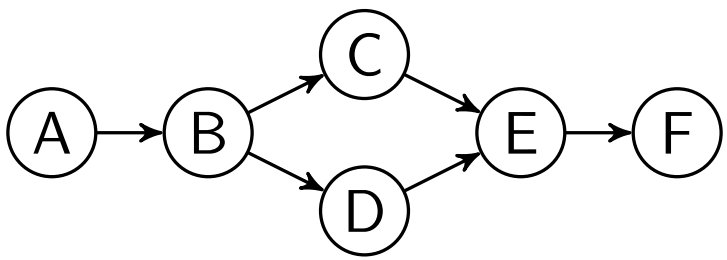
Oppgave 16: Gir det mening å bruke dynamisk programmering hvis alle veiene er enveiskjørt og det ikke er mulig å kjøre i sirkel?

For et sted C på veien mellom A og B kan ikke den lengste veien mellom A og C bruke noen av de samme veiene som mellom C og B.

Oppgave 15: Ønsker å finne lengste vei mellom A og B som kun bruker samme veistrekning en gang. Gir det mening å anvende dynamisk programmering for dette hvis hver delinstans er den lengste veien mellom to steder?

Oppgave 16: Gir det mening å bruke dynamisk programmering hvis alle veiene er enveiskjørt og det ikke er mulig å kjøre i sirkel?

For et sted C på veien mellom A og B kan ikke den lengste veien mellom A og C bruke noen av de samme veiene som mellom C og B.



Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

1. Sorter gjenstandene basert på verdi per vektenhet, v_i/w_i.
2. Inntil du er tom for plass, plukk så mye du kan av den gjenværende gjenstanden med høyest verdi per vektenhet.

Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

Oppgave 18: Finn en algoritme som løser det ubegrensede ryggsekkproblemet med tidskompleksitet $\mathrm{O}(nW)$.

Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

Oppgave 18: Finn en algoritme som løser det ubegrensede ryggsekkproblemet med tidskompleksitet $\mathrm{O}(nW)$.

```
KNAPSACK(n, W)
 1  let K[0..n, 0..W] be a new array
 2  for j = 0 to W
 3       K[0, j] = 0
 4  for i = 1 to n
 5       for j = 0 to W
 6            x = K[i − 1, j]
 7            if j < w_i
 8                 K[i, j] = x
 9            else y = K[i − 1, j − w_i] + v_i
10                 K[i, j] = max(x, y)
```

```
UNLIMITED-KNAPSACK(n, W)
 1  let K[0..n, 0..W] be a new array
 2  for j = 0 to W
 3       K[0, j] = 0
 4  for i = 1 to n
 5       for j = 0 to W
 6            x = K[i − 1, j]
 7            if j < w_i
 8                 K[i, j] = x
 9            else y = K[i, j − w_i] + v_i
10                 K[i, j] = max(x, y)
```

Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

Oppgave 18: Finn en algoritme som løser det ubegrensede ryggsekkproblemet med tidskompleksitet $\mathrm{O}(nW)$.

Oppgave 19: Er det mulig å løse det begrensede ryggsekkproblemet med lavere tidskompleksitet for all k enn hvor raskt man kan løse det binære ryggsekkproblemet?

Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

Oppgave 18: Finn en algoritme som løser det ubegrensede ryggsekkproblemet med tidskompleksitet $\mathrm{O}(nW)$.

Oppgave 19: Er det mulig å løse det begrensede ryggsekkproblemet med lavere tidskompleksitet for all k enn hvor raskt man kan løse det binære ryggsekkproblemet?

KNAPSACK(n, W)
1 **return** LIMITED-KNAPSACK$(n, \mathrm{W}, 1)$

Ryggsekkproblemet

Oppgave 17: Finn en algoritme som løser det kontinuerlige ryggsekkproblemet med tidskompleksitet $\mathrm{O}(n \lg n)$.

Oppgave 18: Finn en algoritme som løser det ubegrensede ryggsekkproblemet med tidskompleksitet $\mathrm{O}(nW)$.

Oppgave 19: Er det mulig å løse det begrensede ryggsekkproblemet med lavere tidskompleksitet for all k enn hvor raskt man kan løse det binære ryggsekkproblemet?

```
KNAPSACK(n, W)
1  return LIMITED-KNAPSACK(n, W, 1)
```

Det begrensede ryggsekkproblemet er minst like vanskelig som det binære ryggsekkproblemet.

Sømfjerning - Programmering

Oppgave 20: Implementer en funksjon som gitt en todimensjonall tabell av tall finner stien fra toppen til bunnen av tabellen.